SACM
Internet-Draft
Intended status: Informational
Expires: April 18, 2016

N. Cam-Winget, Ed.
Cisco Systems
L. Lorenzin
Pulse Secure
I. McDonald
High North Inc
A. Woland
Cisco Systems

October 16, 2015

Secure Automation and Continuous Monitoring (SACM) Architecture
draft-ietf-sacm-architecture-05

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Problem Statement 3
3. Architectural Overview 3
3.1. Component Roles 5
3.1.1. Provider 6
3.1.2. Consumer 7
3.1.3. Types of Providers and Consumers 8
3.1.3.1. Collector 8
3.1.3.2. Evaluator 9
3.1.3.3. Report Generator 9
3.1.3.4. Data Store 9
3.1.4. Controller 9
4. Interfaces between Consumers, Providers, and Controllers . . 11
5. Component Functions 12
5.1. Control Plane Functions 12
5.2. Data Plane Functions 14
6. Component Capabilities 15
7. Example Illustration of Functions and Workflow 15
8. Acknowledgements 18
9. IANA Considerations 18
10. Security Considerations 18
11. References 19
11.1. Normative References 19
11.2. Informative References 19
Authors' Addresses 19

1. Introduction

Several data models and protocols (including - but not limited to - NEA, TCG TNC, SCAP, SWIDs, XMPP, etc.) 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]. The architecture described enables 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].

1.1. Requirements Language

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

When the words appear in lower case, their natural language meaning is used.

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. SACM is standardizing an information model, data models, operations, and transports that will allow for administrators to share with others and to use data from others interoperably.

3. Architectural Overview

At a high level, the SACM architecture describes "Where" and "How" information and assessment of posture may be collected, processed (e.g. normalization, translation, aggregation, etc.), assessed, exchanged, and/or stored. This section provides an architectural overview of

- o the basic architectural building blocks, which - in combination - constitute SACM components (the entities, the "where"), and

- o the relationships and interaction between these building blocks on the data plane and control plane (communications and flows between entities, the "how").

The SACM architecture provides the basic means to describe and compose SACM components. Components enable the basic functionality in SACM, such as Endpoint Attribute Collection or Target Endpoint Posture Assessment.

The role(s) a component plays in the SACM architecture are determined by the function(s) that component instantiates. Three main component roles are defined: a Consumer (Cs), a Provider (Pr), and a Controller (Cr) used to facilitate some of the security functions such as authentication and authorization and other metadata functions. See Section 3.1 for details on roles.

In SACM, components are composed of functions, the modular building blocks in the SACM architecture. The SACM architecture defines the purpose of these functions. Attributes and operations used by component functions are described in other SACM documents. See Section 5 for details on component functions.

Functions use SACM interfaces for communications between components. Interfaces handle management and control functions (such as authentication, authorization, registration, and discovery), and enable SACM components to share information (via publication, query, and subscription). Three primary interfaces are defined: an interface for management and control (A), an interface for data communication between the controller and providers or consumers (B), and an interface for data communication directly between a provider and a consumer (C). See Section 4 for details on interfaces.

Figure 1 illustrates the relationships between component roles and interfaces:

```
                         +--------------------------------------+
                         | +--------------------------------------+
                         | | +--------------------------------------+
                         | | |                                      |
                         +-| |            Consumer (Cs)             |
                           +-|                                      |
                             +--------------------------------------+
                               /   \         /   \            /   \
                              /     \       /     \          /     \
                              -     -       -  d  -          -     -
                               || ||A        | a  |B          |   |C
                               || ||         | t  |           |   |
                              -     -       -  a  -           |   |
                              \     /       \     /           |   |
                               \   /         \   /            |   |
                            /|---------------------|\         |   |
                     /|----/                         \--------| d |--|\
                    /     /      Controller (Cr)      \ ctrl  | a |    \
                    \     \                           / plane | t |    /
                     \|----\                         /--------| a |--|/
                            \|---------------------|/         |   |
                               /   \         /   \            |   |
                              /     \       /     \           |   |
                              -     -       -  d  -           |   |
                               || ||A        | a |B           |   |C
                               || ||         | t |            |   |
                              -     -       -  a  -          -     -
                              \     /       \     /          \     /
                               \   /         \   /            \   /
                             +------------------------------------+
                             |                                    |-+
                             |            Provider (Pr            | |
                             |                                    | |-+
                             +------------------------------------+ | |
                               +------------------------------------+ |
                                 +------------------------------------+
```

Figure 1: Simple Architectural Model

3.1. Component Roles

An endpoint, as defined in [I-D.ietf-sacm-terminology], can operate 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 functions (see Section 5). In the SACM architecture, individual endpoints may be a target endpoint, 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 Provider, Consumer, and/or Controller.

3.1.1. Provider

The Provider (Pr) 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 3.1.3), 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.

One means by which a Provider shares information, is in response to a direct request from a Consumer.

A Provider may also share information spontaneously. Use cases such as the change in a posture state require that a Provider be able to provide such changes or updates especially to Consumers such as Security Information and Event Management (SIEM) systems; similarly, SIEM applications that are providing live information require any such updates or changes to posture information to be provided spontaneously. Authorization for the enabling for these unsolicited messages happens through the Controller at the time that both Provider and Consumers request authorization for (spontaneous) messages.

The information provided, 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. However, it must support either one or more standard data models, or one or more standard 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 3.1.4).

The Provider must be authorized to provide the Posture Assessment Information for specific consumers.

3.1.2. Consumer

The Consumer (Cs) 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 5.2), 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 enable 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 evaluation result 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 3.1.4) 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. Types of Providers and Consumers

SACM Providers and Consumers can perform a variety of SACM-related tasks. For example, a Collector can perform Collection tasks; an Evaluator can perform Evaluation tasks. A single Provider or Consumer may be able to perform only one task, or multiple tasks. SACM defines the following types of Providers/Consumers:

3.1.3.1. Collector

A collector consumes Guidance and/or other Posture Assessment Information; it provides Posture Assessment Information. Collectors may be internal or external. As a SACM component, a Collector may be a Consumer as it may consume guidance information and may also be a Provider as it may publish the collected information.

3.1.3.1.1. Internal Collector

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

3.1.3.1.2. 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

- o A management system that configures and installs software on the endpoint, which collects information based on its provisioning activities

3.1.3.1.3. 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)

3.1.3.2. 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]

3.1.3.3. Report Generator

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

- o Endpoint Attribute Assertions, including Evaluation Results
- o 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.

3.1.3.4. Data Store

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

3.1.4. Controller

The Controller (Cr or Controller) is a component defined to facilitate the overall SACM management and control system functions. This component is responsible for handling the secure communications establishment (such as the authentication and authorization) between Providers and Consumers. In addition, the Controller may also handle how the data may be routed. While the architecture defines the

Controller as a single component, implementations may implement this to suit the different deployment and scaling requirements. In particular, for the data handling, SACM defines three types of Controller:

Broker: Intermediary negotiating connection between Provider and Consumer. Implements only control plane functions. A Controller acting as a Broker:

* Receives a request for information from a Consumer and instructs the Consumer where and how 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. Implements both control and data plane functions. 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. Implements both control and data plane functions. 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.

A single instantiation of a Controller may be a Broker, Proxy, or Repository, or any combination thereof.

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 visibility 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) services in the data plane. 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 services 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 providing the data plane services for the actual data, e.g. Posture Assessment Information flow. Note that Controllers may be implemented to only provide control plane functions (broker), or both control plane functions and data plane services (proxy or repository).

4. Interfaces between Consumers, Providers, and Controllers

A SACM interface is a transport carrying operations (e.g. publication via a RESTful API). 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.

5. Component Functions

SACM components are composed of a variety of functions, which may be instantiated on a single endpoint or on separate standalone endpoints providing various roles. An endpoint MUST implement one or more of these functions to be considered a SACM component. A SACM solution offers a set of functions across a set of SACM components.

The functions described here are the minimum set that is mandatory to implement in a SACM solution. A SACM solution MAY implement additional functions.

5.1. Control Plane Functions

Control plane functions represent various services offered by the Controller to the Providers and Consumers to facilitate sharing of information. Control plane functions include, but are not limited to:

Authentication: The authentication of Consumers and Providers independent of the actual information-sharing communication channel. While authentication between peers (e.g. a Consumer and a Provider) can be achieved directly through peer to peer authentication (using TLS for instance), there are 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.

To address the above use cases, 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. Consumers and Providers that consume or publish information without requiring knowledge of the Providers and Consumers respectively would function in a SACM system where the Controller is a distinct entity. As a distinct SACM component, the Controller would authenticate Providers and Consumers.

Authorization: The restriction of Posture Assessment Information sharing between the Consumers and Providers. At minimum, a management function must define the necessary policies to control what Providers can publish and Consumers to accept. The Controller is the authority for the type of Posture Information that a Provider can publish and a Consumer can accept. If a Controller is a Broker, then it may only grant authorization to the capabilities requested by the Provider or Consumer. When acting as a Proxy, as part of its authorization, the Controller may further obscure or block information being shared by a Provider as it distributes it to a Consumer. Similarly, a Repository may block information as recieved by the Provider and pass to the Consumer and to its storage the resulting authorized information. A Provider may also enforce its own authorization based upon its connection to a Controller; though, in the case where an application includes both the Provider and Controller roles, it can choose to implement all authorization on the Controller. Similarly, a Consumer may enforce its own authorization of what data it can receive based on the Controller (or Provider) it is communicaticating with; in the case where an application includes both the Consumer and Controller roles, it can choose to implement all the authorization on the Controller.

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. Note that implementations of this function is abstractly centralized, but to address scalability and the need to manage different resources (e.g. users, processes and devices) a distributed system that is centrally coordinated may be used.

Registration/Discovery: A SACM ecosystem needs to provide the ability for devices to discover Providers, Consumers, Controllers and their respective capabilities. For a Consumer to be able to obtain the information of interest must either configure itself to know what Providers to communicate with directly (and their known capabilities, such as the supported data model and information provided) or can dynamically discover the information that is available. Similarly, Providers may need to either be configured to know who to publish the information to, or can dynamically discover its Consumers.

In the case where there is a Controller, the capabilities of the Controller must also be advertised so that Providers and Consumers may know how the data is being handled as well (e.g. if acting as a Broker or Repository). The Controller also provides the function of registering the Providers and Consumers; the registration function enables the Controller to also affect the authorization afforded to the Provider or Consumer.

5.2. Data Plane Functions

There are three basic functions to facilitate data flow:

Subscription: A Consumer that wants to recieve information from a specific Provider or from the Controller advertising the availability of specific information (that may come from more than one Provider) will effectively subscribe to recieve the information spontaneously and continuously as new information as subscribed to becomes available.

Publication A Provider being registered through the Controller to provide specific information, may publish the information either directly to the Consumers or to the Controller that is acting as the broker or respository.

Query/Response A Consumer may contact the Provider directly and request the information through a query operation; and in response, the Provider would send the information directly to the Consumer.

6. Component Capabilities

TODO: add a discussion of "capability" as being able to talk a specific data model, data operations, or SACM transport

TODO: data plane capabilities / control plane capabilities can be discovered via querying the controller

7. Example Illustration of Functions and Workflow

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

```
                      +-------------------------------+
                     | +-------------------------------+
                     | |                               |
                     +-|        Controller (Cr)        |
                       +-------------------------------+
                          //   /            \   \\
                         //   /              \   \\
                      A //   /                \   \\ A
                       //   /                  \   \\
                      //   /  B             B   \   \\
                     //   /                      \   \\
    +------------------------+           +------------------------+
    | +----------------------+     A     | +------------------------+
    | |                      |===========| |                        |
    | |    Consumer (C)      |-----------| |      Provider (P)      |
    +-|                      |     C     +-|                        |
       +---------------------+             +------------------------+
```

Figure 2: Communications Model

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:

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 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.

Consumers: a Consumer may request or receive one or more posture attributes or posture assessment information from a 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.

```
                                   +--------------+
                                    |Evaluation   |
                   +-------------+  |Guidance     +--+
                   |Endpoint     |  |Function     |  |
           +-------+             |  +-------------+  |
           |       |             |                   |
           |       +-------+-----+             +-----v-------+
           | Collection    |                   |Evaluation   |
         +-> Function   +--+--------+          |Function     |
         | |            |Collection |    +-----------+   +----------+
         | +------------+Provider   |    |           |---|          |
         |              |           |    |Collection |   |Evaluation|
         |              |           |    |Consumer   |   |Provider  |
         |              +----+------+    +----^------+   +---+------+
        ++---------+         |                |              |
        |Collection|   +-----v------+     +---+--------+     |
        |Guidance  |   |            |     |Collection  |     |
        |Function  |   |Collection  |     |Provider    |     |
        |          |   |Consumer    |-----|            |     |
        +----------+   +------------+     +------------+     |
                                  | Collection |             |
                                  | Data Store |             |
                                  +------------+             |
                                                             |
            +--------------+           +---------------+     |
            |Evaluation    |           |Evaluation     |     |
            |Results       |           |Consumer       <-----+
            |Provider      |-----------|               |
            +-----+--------+           +---------------+
                  |     |Results Reporting|
                  |     |Function         |
                  |     +------------^----+
                  |                  |
            +-----v--------+    +----+------+
            |Evaluation    |    |Reporting  |
            |Results       |    |Guidance   |
            |Consumer      |    |Data Store |
            +---+----------+    +-----------+ +-------------+
                |                             | Results     |
                +-----------------------------> Data Store  |
                                              |             |
                                              +-------------+
```

Figure 3: Example Posture Information Flow

TODO - add example of / more content around interactions with endpoint, possible communications patterns

8. 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.

9. IANA Considerations

This memo includes no request to IANA.

10. 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 disclosure (e.g. where privacy may be needed), 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.

11. References

11.1. Normative References

[I-D.ietf-sacm-requirements]
Cam-Winget, N. and L. Lorenzin, "Secure Automation and Continuous Monitoring (SACM) Requirements", draft-ietf-sacm-requirements-08 (work in progress), July 2015.

[I-D.ietf-sacm-terminology]
Birkholz, H., "Secure Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-07 (work in progress), July 2015.

[I-D.ietf-sacm-use-cases]
Waltermire, D. and D. Harrington, "Endpoint Security Posture Assessment - Enterprise Use Cases", draft-ietf-sacm-use-cases-10 (work in progress), July 2015.

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

11.2. Informative References

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <http://www.rfc-editor.org/info/rfc3444>.

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <http://www.rfc-editor.org/info/rfc5209>.

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 D. Waltermire, Ed.
Internet-Draft NIST
Intended status: Standards Track K. Watson
Expires: October 29, 2017 DHS
C. Kahn
L. Lorenzin
Pulse Secure, LLC
M. Cokus
D. Haynes
The MITRE Corporation
H. Birkholz
Fraunhofer SIT
April 27, 2017

SACM Information Model
draft-ietf-sacm-information-model-10

Abstract

This document defines the Information Elements that are transported between SACM components and their interconnected relationships. The primary purpose of the Secure Automation and Continuous Monitoring (SACM) Information Model is to ensure the interoperability of corresponding SACM data models and addresses the use cases defined by SACM. The Information Elements and corresponding types are maintained as the IANA "SACM Information Elements" registry.

Copyright Notice

Table of Contents

1. Introduction 12
2. Conventions used in this document 13
  2.1. Requirements Language 13
  2.2. Information Element Examples 13
3. Information Elements 13
  3.1. Context of Information Elements 14
  3.2. Extensibility of Information Elements 14
4. Structure of Information Elements 14
  4.1. Information Element Naming Convention 17
  4.2. SACM Content Elements 18
  4.3. SACM Statements 18
  4.4. Relationships 20
  4.5. Event 22
  4.6. Categories 23
5. Abstract Data Types 23
  5.1. Simple Datatypes 23
    5.1.1. IPFIX Datatypes 23
  5.2. Structured Datatypes 24
    5.2.1. List Datatypes 24
    5.2.2. Enumeration Datatype 25
    5.2.3. Category Datatype 26
6. Information Model Assets 26
  6.1. Asset 27
  6.2. Endpoint 28
  6.3. Hardware Component 28
  6.4. Software Component 29
    6.4.1. Software Instance 29
  6.5. Identity 29
  6.6. Guidance 29
    6.6.1. Collection Guidance 30
    6.6.2. Evaluation Guidance 30

```
       6.6.3.  Classification Guidance . . . . . . . . . . . . . . .  31
       6.6.4.  Storage Guidance  . . . . . . . . . . . . . . . . . .  31
       6.6.5.  Evaluation Results  . . . . . . . . . . . . . . . . .  31
   7.  Information Model Elements  . . . . . . . . . . . . . . . . .  32
     7.1.  sacmStatement . . . . . . . . . . . . . . . . . . . . . .  32
     7.2.  sacmStatementMetadata . . . . . . . . . . . . . . . . . .  32
     7.3.  sacmContentElement  . . . . . . . . . . . . . . . . . . .  32
     7.4.  sacmContentElementMetadata  . . . . . . . . . . . . . . .  33
     7.5.  targetEndpoint  . . . . . . . . . . . . . . . . . . . . .  33
     7.6.  targetEndpointIdentifier  . . . . . . . . . . . . . . . .  33
     7.7.  targetEndpointLabel . . . . . . . . . . . . . . . . . . .  33
     7.8.  anyIE . . . . . . . . . . . . . . . . . . . . . . . . . .  34
     7.9.  accessPrivilegeType . . . . . . . . . . . . . . . . . . .  34
     7.10. accountName . . . . . . . . . . . . . . . . . . . . . . .  34
     7.11. administrativeDomainType  . . . . . . . . . . . . . . . .  34
     7.12. addressAssociationType  . . . . . . . . . . . . . . . . .  34
     7.13. addressMaskValue  . . . . . . . . . . . . . . . . . . . .  35
     7.14. addressType . . . . . . . . . . . . . . . . . . . . . . .  35
     7.15. addressValue  . . . . . . . . . . . . . . . . . . . . . .  35
     7.16. applicationComponent  . . . . . . . . . . . . . . . . . .  35
     7.17. applicationLabel  . . . . . . . . . . . . . . . . . . . .  36
     7.18. applicationType . . . . . . . . . . . . . . . . . . . . .  36
     7.19. applicationManufacturer . . . . . . . . . . . . . . . . .  36
     7.20. authenticator . . . . . . . . . . . . . . . . . . . . . .  36
     7.21. authenticationType  . . . . . . . . . . . . . . . . . . .  36
     7.22. birthdate . . . . . . . . . . . . . . . . . . . . . . . .  37
     7.23. bytesReceived . . . . . . . . . . . . . . . . . . . . . .  37
     7.24. bytesReceived . . . . . . . . . . . . . . . . . . . . . .  37
     7.25. bytesSent . . . . . . . . . . . . . . . . . . . . . . . .  37
     7.26. certificate . . . . . . . . . . . . . . . . . . . . . . .  38
     7.27. collectionTaskType  . . . . . . . . . . . . . . . . . . .  38
     7.28. confidence  . . . . . . . . . . . . . . . . . . . . . . .  38
     7.29. contentAction . . . . . . . . . . . . . . . . . . . . . .  38
     7.30. countryCode . . . . . . . . . . . . . . . . . . . . . . .  38
     7.31. dataOrigin  . . . . . . . . . . . . . . . . . . . . . . .  39
     7.32. dataSource  . . . . . . . . . . . . . . . . . . . . . . .  39
     7.33. default-depth . . . . . . . . . . . . . . . . . . . . . .  39
     7.34. discoverer  . . . . . . . . . . . . . . . . . . . . . . .  39
     7.35. emailAddress  . . . . . . . . . . . . . . . . . . . . . .  40
     7.36. eventType . . . . . . . . . . . . . . . . . . . . . . . .  40
     7.37. eventThreshold  . . . . . . . . . . . . . . . . . . . . .  40
     7.38. eventThresholdName  . . . . . . . . . . . . . . . . . . .  40
     7.39. eventTrigger  . . . . . . . . . . . . . . . . . . . . . .  40
     7.40. firmwareId  . . . . . . . . . . . . . . . . . . . . . . .  41
     7.41. hostName  . . . . . . . . . . . . . . . . . . . . . . . .  41
     7.42. interfaceLabel  . . . . . . . . . . . . . . . . . . . . .  41
     7.43. ipv6AddressSubnetMask . . . . . . . . . . . . . . . . . .  41
     7.44. ipv6AddressSubnetMaskCidrNotation . . . . . . . . . . . .  41
```

7.45. ipv6AddressValue 42
7.46. ipv4AddressSubnetMask 42
7.47. ipv4AddressSubnetMaskCidrNotation 42
7.48. ipv4AddressValue 42
7.49. layer2InterfaceType 42
7.50. layer4PortAddress 42
7.51. layer4Protocol 43
7.52. locationName 43
7.53. networkZoneLocation 43
7.54. layer2NetworkLocation 43
7.55. layer3NetworkLocation 44
7.56. macAddressValue 44
7.57. methodLabel 44
7.58. methodRepository 44
7.59. networkAccessLevelType 44
7.60. networkId 45
7.61. networkInterfaceName 45
7.62. networkLayer 45
7.63. networkName 45
7.64. organizationId 45
7.65. patchId 46
7.66. patchName 46
7.67. personFirstName 46
7.68. personLastName 46
7.69. personMiddleName 46
7.70. phoneNumber 46
7.71. phoneNumberType 47
7.72. privilegeName 47
7.73. privilegeValue 47
7.74. protocol 47
7.75. publicKey 48
7.76. relationshipContentElementGuid 48
7.77. relationshipStatementElementGuid 48
7.78. relationshipObjectLabel 48
7.79. relationshipType 48
7.80. roleName 49
7.81. sessionStateType 49
7.82. statementGuid 49
7.83. statementType 49
7.84. status 50
7.85. subAdministrativeDomain 50
7.86. subInterfaceLabel 50
7.87. superAdministrativeDomain 50
7.88. superInterfaceLabel 51
7.89. teAssessmentState 51
7.90. teLabel 51
7.91. teId 51
7.92. timestampType 51

7.93. unitsReceived 52
7.94. unitsSent 52
7.95. userDirectory 52
7.96. sacmUserId 52
7.97. webSite 53
7.98. WGS84Longitude 53
7.99. WGS84Latitude 53
7.100. WGS84Altitude 53
7.101. hardwareSerialNumber 53
7.102. interfaceName 54
7.103. interfaceIndex 54
7.104. interfaceMacAddress 54
7.105. interfaceType 54
7.106. interfaceFlags 54
7.107. networkInterface 55
7.108. softwareIdentifier 55
7.109. softwareTitle 55
7.110. softwareCreator 56
7.111. simpleSoftwareVersion 56
7.112. rpmSoftwareVersion 56
7.113. ciscoTrainSoftwareVersion 56
7.114. softwareVersion 56
7.115. softwareLastUpdated 57
7.116. softwareClass 57
7.117. softwareInstance 58
7.118. globallyUniqueIdentifier 59
7.119. creationTimestamp 59
7.120. collectionTimestamp 59
7.121. publicationTimestamp 59
7.122. relayTimestamp 59
7.123. storageTimestamp 60
7.124. type 60
7.125. protocolIdentifier 60
7.126. sourceTransportPort 60
7.127. sourceIPv4PrefixLength 61
7.128. ingressInterface 61
7.129. destinationTransportPort 61
7.130. sourceIPv6PrefixLength 61
7.131. sourceIPv4Prefix 62
7.132. destinationIPv4Prefix 62
7.133. sourceMacAddress 62
7.134. ipVersion 62
7.135. interfaceDescription 62
7.136. applicationDescription 62
7.137. applicationId 63
7.138. applicationName 63
7.139. exporterIPv4Address 63
7.140. exporterIPv6Address 63

7.141. portId 63
7.142. templateId 64
7.143. collectorIPv4Address 64
7.144. collectorIPv6Address 64
7.145. informationElementIndex 65
7.146. informationElementId 65
7.147. informationElementDataType 65
7.148. informationElementDescription 65
7.149. informationElementName 66
7.150. informationElementRangeBegin 66
7.151. informationElementRangeEnd 66
7.152. informationElementSemantics 67
7.153. informationElementUnits 67
7.154. applicationCategoryName 68
7.155. mibObjectValueInteger 68
7.156. mibObjectValueOctetString 69
7.157. mibObjectValueOID 69
7.158. mibObjectValueBits 69
7.159. mibObjectValueIPAddress 70
7.160. mibObjectValueCounter 70
7.161. mibObjectValueGauge 71
7.162. mibObjectValueTimeTicks 71
7.163. mibObjectValueUnsigned 72
7.164. mibObjectValueTable 72
7.165. mibObjectValueRow 72
7.166. mibObjectIdentifier 73
7.167. mibSubIdentifier 73
7.168. mibIndexIndicator 73
7.169. mibCaptureTimeSemantics 74
7.170. mibContextEngineID 75
7.171. mibContextName 76
7.172. mibObjectName 76
7.173. mibObjectDescription 76
7.174. mibObjectSyntax 76
7.175. mibModuleName 76
7.176. interface 77
7.177. iflisteners 77
7.178. physicalProtocol 77
7.179. hwAddress 78
7.180. programName 79
7.181. userId 79
7.182. inetlisteningserver 79
7.183. transportProtocol 79
7.184. localAddress 79
7.185. localPort 80
7.186. localFullAddress 80
7.187. foreignAddress 80
7.188. foreignFullAddress 80

7.189. selinuxboolean 80
7.190. selinuxName 81
7.191. currentStatus 81
7.192. pendingStatus 81
7.193. selinuxsecuritycontext 81
7.194. filepath 82
7.195. path 82
7.196. filename 82
7.197. pid 82
7.198. role 82
7.199. domainType 83
7.200. lowSensitivity 83
7.201. lowCategory 83
7.202. highSensitivity 83
7.203. highCategory 83
7.204. rawlowSensitivity 84
7.205. rawlowCategory 84
7.206. rawhighSensitivity 84
7.207. rawhighCategory 84
7.208. systemdunitdependency 84
7.209. unit 85
7.210. dependency 85
7.211. systemdunitproperty 85
7.212. property 85
7.213. systemdunitValue 85
7.214. file 86
7.215. fileType 86
7.216. groupId 86
7.217. aTime 86
7.218. cTime 86
7.219. mTime 87
7.220. size 87
7.221. suid 87
7.222. sgid 87
7.223. sticky 87
7.224. hasExtendedAcl 88
7.225. inetd 88
7.226. serverProgram 88
7.227. inetdEndpointType 88
7.228. execAsUser 89
7.229. waitStatus 89
7.230. inetAddr 90
7.231. netmask 90
7.232. passwordInfo 90
7.233. username 91
7.234. password 91
7.235. gcos 91
7.236. homeDir 91

7.237. loginShell 91
7.238. lastLogin 92
7.239. process 92
7.240. commandLine 92
7.241. ppid 92
7.242. priority 93
7.243. startTime 93
7.244. routingtable 93
7.245. destination 93
7.246. gateway 93
7.247. runlevelInfo 94
7.248. runlevel 94
7.249. start 94
7.250. kill 94
7.251. shadowItem 94
7.252. chgLst 95
7.253. chgAllow 95
7.254. chgReq 95
7.255. expWarn 95
7.256. expInact 95
7.257. expDate 96
7.258. encryptMethod 96
7.259. symlink 96
7.260. symlinkFilepath 96
7.261. canonicalPath 97
7.262. sysctl 97
7.263. kernelParameterName 97
7.264. kernelParameterValue 97
7.265. uname 98
7.266. machineClass 98
7.267. nodeName 98
7.268. osName 98
7.269. osRelease 98
7.270. processorType 99
7.271. internetService 99
7.272. serviceProtocol 99
7.273. serviceName 99
7.274. flags 99
7.275. noAccess 100
7.276. onlyFrom 100
7.277. port 100
7.278. server 100
7.279. serverArguments 100
7.280. socketType 101
7.281. registeredServiceType 101
7.282. wait 101
7.283. disabled 102
7.284. windowsView 102

7.285. fileauditedpermissions 102
7.286. trusteeName 103
7.287. auditStandardDelete 103
7.288. auditStandardReadControl 103
7.289. auditStandardWriteDac 104
7.290. auditStandardWriteOwner 104
7.291. auditStandardSynchronize 105
7.292. auditAccessSystemSecurity 105
7.293. auditGenericRead 106
7.294. auditGenericWrite 106
7.295. auditGenericExecute 107
7.296. auditGenericAll 107
7.297. auditFileReadData 108
7.298. auditFileWriteData 108
7.299. auditFileAppendData 109
7.300. auditFileReadEa 109
7.301. auditFileWriteEa 110
7.302. auditFileExecute 110
7.303. auditFileDeleteChild 111
7.304. auditFileReadAttributes 111
7.305. auditFileWriteAttributes 112
7.306. fileeffectiverights 112
7.307. standardDelete 113
7.308. standardReadControl 113
7.309. standardWriteDac 113
7.310. standardWriteOwner 114
7.311. standardSynchronize 114
7.312. accessSystemSecurity 114
7.313. genericRead 114
7.314. genericWrite 114
7.315. genericExecute 115
7.316. genericAll 115
7.317. fileReadData 115
7.318. fileWriteData 115
7.319. fileAppendData 115
7.320. fileReadEa 116
7.321. fileWriteEa 116
7.322. fileExecute 116
7.323. fileDeleteChild 116
7.324. fileReadAttributes 116
7.325. fileWriteAttributes 117
7.326. groupInfo 117
7.327. group 117
7.328. subgroup 117
7.329. groupSidInfo 117
7.330. userSidInfo 118
7.331. userSid 118
7.332. subgroupSid 118

7.333. lockoutpolicy 118
7.334. forceLogoff 118
7.335. lockoutDuration 119
7.336. lockoutObservationWindow 119
7.337. lockoutThreshold 119
7.338. passwordpolicy 119
7.339. maxPasswdAge 120
7.340. minPasswdAge 120
7.341. minPasswdLen 120
7.342. passwordHistLen 121
7.343. passwordComplexity 121
7.344. reversibleEncryption 121
7.345. portInfo 121
7.346. foreignPort 121
7.347. printereffectiverights 122
7.348. printerName 122
7.349. printerAccessAdminister 122
7.350. printerAccessUse 122
7.351. jobAccessAdminister 122
7.352. jobAccessRead 123
7.353. registry 123
7.354. registryHive 123
7.355. registryKey 124
7.356. registryKeyName 124
7.357. lastWriteTime 124
7.358. registryKeyType 125
7.359. registryKeyValue 126
7.360. regkeyauditedpermissions 127
7.361. auditKeyQueryValue 128
7.362. auditKeySetValue 128
7.363. auditKeyCreateSubKey 129
7.364. auditKeyEnumerateSubKeys 129
7.365. auditKeyNotify 130
7.366. auditKeyCreateLink 130
7.367. auditKeyWow6464Key 131
7.368. auditKeyWow6432Key 131
7.369. auditKeyWow64Res 132
7.370. regkeyeffectiverights 132
7.371. keyQueryValue 133
7.372. keySetValue 133
7.373. keyCreateSubKey 133
7.374. keyEnumerateSubKeys 134
7.375. keyNotify 134
7.376. keyCreateLink 134
7.377. keyWow6464Key 134
7.378. keyWow6432Key 134
7.379. keyWow64Res 134
7.380. service 135

7.381. displayName 135
7.382. description 135
7.383. serviceType 135
7.384. startType 136
7.385. currentState 137
7.386. controlsAccepted 138
7.387. startName 140
7.388. serviceFlag 140
7.389. dependencies 140
7.390. serviceeffectiverights 140
7.391. trusteeSid 141
7.392. serviceQueryConf 141
7.393. serviceChangeConf 141
7.394. serviceQueryStat 141
7.395. serviceEnumDependents 141
7.396. serviceStart 142
7.397. serviceStop 142
7.398. servicePause 142
7.399. serviceInterrogate 142
7.400. serviceUserDefined 142
7.401. sharedresourceauditedpermissions 143
7.402. netname 143
7.403. sharedresourceeffectiverights 143
7.404. user 144
7.405. enabled 144
7.406. lastLogon 144
7.407. groupSid 144
7.408. endpointType 144
7.409. endpointPurpose 145
7.410. endpointCriticality 145
7.411. ingestTimestamp 145
7.412. vulnerabilityVersion 146
7.413. vulnerabilityExternalId 146
7.414. vulnerabilitySeverity 146
7.415. assessmentTimestamp 146
7.416. vulnerableSoftware 146
7.417. endpointVulnerabilityStatus 147
7.418. vulnerabilityDescription 147
8. Acknowledgements 147
9. IANA Considerations 148
10. Security Considerations 148
11. Operational Considerations 149
11.1. Endpoint Designation 149
11.2. Timestamp Accuracy 150
12. Privacy Considerations 151
13. References 151
13.1. Normative References 151
13.2. Informative References 151

Appendix A. Change Log 152
A.1. Changes in Revision 01 152
A.2. Changes in Revision 02 154
A.3. Changes in Revision 03 154
A.4. Changes in Revision 04 154
A.5. Changes in Revision 05 155
A.6. Changes in Revision 06 155
A.7. Changes in Revision 07 155
A.8. Changes in Revision 08 156
A.9. Changes in Revision 09 156
A.10. Changes in Revision 10 157
Authors' Addresses 157

1. Introduction

The SACM Information Model (IM) serves multiple purposes:

- to ensure interoperability between SACM data models that are used as transport encodings,

- to provide a standardized set of Information Elements - the SACM Vocabulary - to enable the exchange of content vital to automated security posture assessment, and

- to enable secure information sharing in a scalable and extensible fashion in order to support the tasks conducted by SACM components.

A complete set of requirements imposed on the IM can be found in [I-D.ietf-sacm-requirements]. The SACM IM is intended to be used for standardized data exchange between SACM components (data in motion). Nevertheless, the Information Elements (IE) and their relationships defined in this document can be leveraged to create and align corresponding data models for data at rest.

The information model expresses, for example, target endpoint (TE) attributes, guidance, and evaluation results. The corresponding Information Elements are consumed and produced by SACM components as they carry out tasks.

The primary tasks that this information model supports (on data, control, and management plane) are:

- TE Discovery

- TE Characterization

- TE Classification

- Collection
- Evaluation
- Information Sharing
- SACM Component Discovery
- SACM Component Authentication
- SACM Component Authorization
- SACM Component Registration

These tasks are defined in [I-D.ietf-sacm-terminology].

2. Conventions used in this document

2.1. Requirements Language

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

2.2. Information Element Examples

The notation used to define the SACM Information Elements (IEs) is based on a customized version of the IPFIX information model syntax [RFC7012] which is described in Figure 2. However, there are several examples presented throughout the document that use a simplified pseudo-code to illustrate the basic structure. It should be noted that while they include actual names of subjects and attributes as well as values, they are not intended to influence how corresponding SACM IEs should be defined in Section 7. The examples are provided for demonstration purposes only.

3. Information Elements

The IEs defined in this document comprise the building blocks by which all SACM content is composed. They are consumed and provided by SACM components on the data plane. Every Information Element has a unique label: its name. Every type of IE defined by the SACM IM is registered as a type at the IANA registry. The Integer Index of the IANA SMI number tables can be used by SACM data models.

3.1. Context of Information Elements

The IEs in this information model represent information related to assets in the following areas (based on the use cases described in [RFC7632]):

- Endpoint Management

- Software Inventory Management

- Hardware Inventory Management

- Configuration Management

- Vulnerability Management

3.2. Extensibility of Information Elements

A SACM data model based on this information model MAY include additional information elements that are not defined here. The labels of additional Information Elements included in different SACM data models MUST NOT conflict with the labels of the Information Elements defined by this information model, and the names of additional Information Elements MUST NOT conflict with each other or across multiple data models. In order to avoid naming conflicts, the labels of additional IEs SHOULD be prefixed to avoid collisions across extensions. The prefix MUST include an organizational identifier and therefore, for example, MAY be an IANA enterprise number, a (partial) name space URI, or an organization name abbreviation.

4. Structure of Information Elements

There are two basic types of IEs:

- Attributes: Atomic information elements that are equivalent to name-value-pairs and can be components of Subjects.

- Subjects: Composite information elements that have a name and are made up of Attributes and/or other Subjects. Every IE that is part of a Subject can have a quantity associated with it (e.g. zero-one, none-unbounded). The content IEs of a Subject can be ordered or unordered.

```
Example Instance of an Attribute:
hostname = "arbutus"

Example Instance of a Subject:
coordinates = (
latitude = N27.99619,
longitude = E86.92761
)
```

Figure 1: Example instance of an attribute and subject.

In general, every piece of information that enables security posture assessment or further enriches the quality of the assessment process can be associated with metadata. In the SACM IM, metadata is represented by specific subjects and is bundled with other attributes or subjects to provide additional information about them. The IM explicitly defines two kinds of metadata:

- o Metadata focusing on the data origin (the SACM component that provides the information to the SACM domain)

- o Metadata focusing on the data source (the target endpoint that is assessed)

Metadata can also include relationships that refer to other associated IEs (or SACM content in general) by using referencing labels that have to be included in the metadata of the associated IE.

Subjects can be nested and the SACM IM allows for circular or recursive nesting. The association of IEs via nesting results in a tree-like structure wherein subjects compose the root and intermediary nodes and attributes the leaves of the tree. This semantic structure does not impose a specific structure on SACM data models regarding data in motion or data repository schemata for data at rest.

The SACM IM provides two conceptual top-level subjects that are used to ensure a homogeneous structure for SACM content and its associated metadata: SACM statements and SACM content-elements. Every set of IEs that is provided by a SACM component must provide the information contained in these two subjects although it is up to the implementer whether or not the subjects are explicitly defined in a data model.

The notation the SACM IM is defined in is based on a modified version of the IP Information Flow Export (IPFIX) Information Model syntax described in Section 2.1 of [RFC7012]. The customized syntax used by the SACM IM is defined below in Figure 2.

elementId (required): The numeric identifier of the Information Element. It is used for the compact identification of an Information Element. If this identifier is used without an enterpriseID, then the elementId must be unique, and the description of allowed values is administrated by IANA. The value "TBD" may be used during development of the information model until an elementId is assigned by IANA and filled in at publication time.

enterpriseId (optional): Enterprises may wish to define Information Elements without registering them with IANA, for example, for enterprise-internal purposes. For such Information Elements, the elementId is not sufficient when used outside the enterprise. If specifications of enterprise-specific Information Elements are made public and/or if enterprise-specific identifiers are used by SACM components outside the enterprise, then the enterprise-specific identifier MUST be made globally unique by combining it with an enterprise identifier. Valid values for the enterpriseId are defined by IANA as Structure of Management Information (SMI) network management private enterprise numbers.

name (required): A unique and meaningful name for the Information Element.

dataType (required): There are two kinds of datatypes: simple and structured. Attributes are defined using simple datatypes and subjects are defined using structured datatypes. The contents of the datatype field will be either a reference to one of the simple datatypes listed in Section 5.1, or the specification of structured datatype as defined in Section 5.2.

status (required): The status of the specification of the Information Element. Allowed values are "current" and "deprecated". All newly defined Information Elements have "current" status. The process for moving Information Elements to the "deprecated" status is TBD.

description (required): Describes the meaning of the Information Element, how it is derived, conditions for its use, etc.

structure (optional): A parsable property that provides details about the definition of structured Information Elements as described in Section 5.2.

references (optional): Identifies other RFCs or documents outside the IETF which provide additional information or context about the Information Element.

Figure 2: Information Element Specification Template

4.1. Information Element Naming Convention

SACM Information Elements must adhere to the following naming conventions.

- o Names SHOULD be descriptive
- o Names MUST be unique within the SACM registry. Enterprise-specific names SHOULD be prefixed with a Private Enterprise Number [PEN].
- o Names MUST start with lowercase letters unless it begins with a Private Enterprise Number
- o Composed names MUST use capital letters for the first letter of each part

4.2. SACM Content Elements

Every piece of information that is provided by a SACM Component is always associated with a set of data source metadata (e.g. the timestamp when the information was collected, the target endpoint from which the this set of information is about, etc.) which is provided in the SACM Content Element Metadata. The SACM Content Element is the subject information element that associates the information with the SACM Content Element Metadata. The SACM Content Element Metadata may also include relationships that express associations with other SACM Content Elements.

```
content-element = (
  content-metadata = (
    collection-timestamp = 146193322,
    data-source = fb02e551-7101-4e68-8dec-1fde6bd10981
  ),
  hostname = "arbutus",
  coordinates = (
  latitude = N27.99619,
  longitude = E86.92761
  )
)
```

Figure 3: Example set of IEs associated with a timestamp and a target endpoint label.

4.3. SACM Statements

One or more SACM Content Elements are bundled in a SACM Statement. In contrast to SACM Content Element Metadata, SACM Statement Metadata focuses on the providing information about the SACM Component that provided it rather than the target endpoint that the content is about. The only content-specific metadata included in the SACM Statement is the statement-type IE. Therefore, multiple SACM Content Elements that share the same SACM Statement Metadata and are of the same statement-type can be included in a single SACM Statement. A SACM Statement functions similar to an envelope or a header and is the subject information element that associates SACM Statement Metadata with security automation information provided in its SACM Content Element(s). Its purpose is to enable the tracking of the origin of data inside a SACM domain and more importantly to enable the mitigation of conflicting information that may originate from different SACM Components. How a consuming SACM Component actually deals with conflicting information is out-of-scope of the SACM IM. Semantically, the term statement implies that the SACM content provided by a SACM Component might not be correct in every context, but, rather is the result of a best-effort to produce correct information.

```
sacm-statement = (
  statement-metadata = (
    publish-timestamp = 1461934031,
    data-origin = 24e67957-3d31-4878-8892-da2b35e121c2,
    statement-type = observation
  ),
  content-element = (
    content-metadata = (
      collection-timestamp = 146193322,
      data-source = fb02e551-7101-4e68-8dec-1fde6bd10981
    ),
    hostname = "arbutus"
  )
)
```

Figure 4: Example of a simple SACM statement including a single content-element.

```
sacm-statement = (
  statement-metadata = (
    publish-timestamp = 1461934031,
    data-origin = 24e67957-3d31-4878-8892-da2b35e121c2
    statement-type = observation
  ),
  content-element = (
    content-metadata = (
      collection-timestamp = 146193322,
      data-source = fb02e551-7101-4e68-8dec-1fde6bd10981
    ),
    coordinates = (
      latitude = N27.99619,
      longitude = E86.92761
    )
  )
)

sacm-statement = (
  statement-metadata = (
    publish-timestamp = 1461934744,
    data-origin = e42885a1-0270-44e9-bb5c-865cf6bd4800,
    statement-type = observation
  ),
  content-element = (
    content-metadata = (
      collection-timestamp = 146193821,
      te-label = fb02e551-7101-4e68-8dec-1fde6bd10981
    ),
    coordinates = (
      latitude = N16.67622,
      longitude = E141.55321
    )
  )
)
```

Figure 5: Example of conflicting information originating from different SACM components.

4.4. Relationships

An IE can be associated with another IE, e.g. a user-name attribute can be associated with a content-authorization subject. These references are expressed via the relationships subject, which can be included in a corresponding content-metadata subject. The relationships subject includes a list of one or more references. The SACM IM does not enforce a SACM domain to use unique identifiers as references. Therefore, there are at least two ways to reference another

- o The value of a reference represents a specific content-label that is unique in a SACM domain (and has to be included in the corresponding content-element metadata in order to be referenced), or

- o The reference is a subject that includes an appropriate number of IEs in order to identify the referenced content-element by its actual content.

It is recommended to provide unique identifiers in a SACM domain and the SACM IM provides a corresponding naming-convention as a reference in Section 4.1. The alternative highlighted above summarizes a valid approach that does not require unique identifiers and is similar to the approach of referencing target endpoints via identifying attributes included in a characterization record.

```
content-element = (
  content-metadata = (
    collection-timestamp = 1461934031,
    te-label =
    fb02e551-7101-4e68-8dec-1fde6bd10981
    relationships = (
      associated-with-user-account =
      f3d70ef4-7e18-42af-a894-8955ba87c95d
    )
  ),
  hostname = "arbutus"
)

content-element = (
  content-metadata = (
    content-label = f3d70ef4-7e18-42af-a894-8955ba87c95d
  ),
  user-account = (
    username = romeo
    authentication = local
  )
)
```

Figure 6: Example instance of a content-element subject associated with another subject via its content metadata.

4.5. Event

Event subjects provide a structure to represent the change of IE values that was detected by a collection task at a specific point of time. It is mandatory to include the new values and the collection timestamp in an event subject and it is recommended to include the past values and a collection timestamp that were replaced by the new IE values. Every event can also be associated with a subject-specific event-timestamp and a lastseen-timestamp that might differ from the corresponding collection-timestamps. If these are omitted the collection-timestamp that is included in the content-metadata subject is used instead.

```
sacm-statement = (
  statement-metadata = (
    publish-timestamp = 1461934031,
    data-origin = 24e67957-3d31-4878-8892-da2b35e121c2,
    statement-type = event
  ),
  event = (
    event-attributes = (
      event-name = "host-name change",
      content-element = (
        content-metadata = (
        collection-timestamp = 146193322,
        data-source =
          fb02e551-7101-4e68-8dec-1fde6bd10981,
          event-component = past-state
       ),
       hostname = "arbutus"
      ),
      content-element = (
        content-metadata = (
          collection-timestamp = 146195723,
          data-source =
          fb02e551-7101-4e68-8dec-1fde6bd10981,
          event-component = current-state
        ),
        hostname = "lilac"
      )
    )
  )
```

Figure 7: Example of a SACM statement containing an event.

4.6. Categories

Categories are special IEs that refer to multiple types of IEs via just one name. Therefore, they are similar to a type-choice. A prominent example of a category is when identifying a target endpoint. In some cases, a target endpoint will be identified by a set of identifying attributes and in other cases a target endpoint will be identified by a target endpoint label which is unique within a SACM domain. If a subject includes the targetEndpoint information element as one of its components, any of the category members (targetEndpointIdentifier or targetEndpointLabel) are valid to be used in its place.

5. Abstract Data Types

This section describes the set of valid abstract data types that can be used for the specification of the SACM Information Elements in Section 7. SACM currently supports two classes of datatypes that can be used to define Information Elements.

- o Simple: Datatypes that are atomic and are used to define the type of data represented by an attribute Information Element.

- o Structured: Datatypes that can be used to define the type of data represented by a subject Information Element.

Note that further abstract data types may be specified by future extensions of the SACM information model.

5.1. Simple Datatypes

5.1.1. IPFIX Datatypes

To facilitate the use of existing work, SACM supports the following abstract data types defined in Section 3 of [RFC7012].

- o unsigned8, unsigned16, unsigned32, unsigned64

- o signed8, signed16, signed32, signed64

- o float32, float64

- o boolean

- o macAddress

- o octetArray

- string

- dateTimeSeconds, dateTimeMilliseconds, dateTimeMicroseconds, dateTimeNanoSeconds

- ipv4Address, ipv6Address

5.2. Structured Datatypes

5.2.1. List Datatypes

SACM defines the following abstract list data types that are used to represent the structured data associated with subjects.

- list: indicates that the Information Element order is not significant but MAY be preserved.

- orderedList: indicates that Information Element order is significant and MUST be preserved.

The notation for defining a SACM structured datatype is based on regular expressions, which are composed of the keywords "list" or "orderedList" and an Information Element expression. IE expressions use some of the regular expression syntax and operators, but the terms in the expression are the names of defined Information Elements instead of character classes. The syntax for defining list and orderedList datatypes is described below, using BNF:

```
<list-def> -> ("list"|"orderedList") "(" <ie-expression> ")"

<ie-expression> -> <ie-name> <cardinality>?
                   ( ("," | "|") <ie-name> <cardinality>?)*

<cardinality> -> "*" | "+" | "?" |
                 ( "(" <non-neg-int> ("," <non-neg-int>)? ")" )
```

Figure 8: Syntax for Defining List Datatypes

As seen above, multiple occurrences of an Information Element may be present in a structured datatype. The cardinality of an Information Element within a structured Information Element definition is defined by the following operators:

```
       * - zero or more occurrences

       + - one or more occurrences

       ? - zero or one occurrence

      (m,n) - between m and n occurrences
```

Figure 9: Specifying Cardinality for Structured Datatypes

The absence of a cardinality operator implies one mandatory occurrence of the Information Element.

Below is an example of a structured Information Element definition.

```
personInfo = list(firstName, middleNames?, lastName)
firstName = string
middleNames = orderedList(middleName+)
middleName = string
lastName = string
```

As an example, consider the name "John Ronald Reuel Tolkien". Below are instances of this name, structured according to the personInfo definition.

```
personInfo = (firstName="John", middleNames(middleName="Ronald",
              middleName="Reuel"), lastName="Tolkien")

personInfo = (middleNames(middleName="Ronald", middleName=" Reuel"),
              lastName="Tolkien", firstName="John")
```

The instance below is not legal with respect to the definition of personInfo because the order in middleNames is not preserved.

```
personInfo = (firstName="John", middleNames(middleName=" Reuel",
              middleName="Ronald"), lastName="Tolkien")
```

Figure 10: Example of Defining a Structured List Datatype

5.2.2. Enumeration Datatype

SACM defines the following abstract enumeration datatype that is used to represent the restriction of an attribute value to a set of values.

```
name, hex-value, description
<enumeration-def> -> -> <name> ";" <hex-value> ";" <description>
<name> -> [0-9a-zA-Z]+
<hex-value> -> 0x[0-9a-fA-F]+
<description> -> [0-9a-zA-Z\.\,]+
```

Figure 11: Syntax for Defining an Enumeration Datatype

Below is an example of a structured Information Element definition for an enumeration.

```
Red    ; 0x1  ; The color is red.
Orange ; 0x2  ; The color is orange.
Yellow ; 0x3  ; The color is yellow.
Green  ; 0x4  ; The color is green.
...
```

Figure 12: Example of Defining a Structured Enumeration Datatype

5.2.3. Category Datatype

SACM defines the following abstract category datatype that is used to represent a type-choice between a set of information elements.

```
<category-def> -> "category(" <ie-expression> ")"
<ie-expression> -> <ie-name> ("|" <ie-name>)*
<name> -> [0-9a-zA-Z]+
```

Figure 13: Syntax for Defining an Category Datatype

Below is an example of a structured Information Element definition for a category.

```
targetEndpoint = category(targetEndpointIdentifier |
                          targetEndpointLabel)
```

Figure 14: Example of Defining a Structured Category Datatype

6. Information Model Assets

In order to represent the Information Elements related to the areas listed in Section 3.1, the information model defines the information needs (or metadata about those information needs) related to following types of assets which are defined in [I-D.ietf-sacm-terminology] (and included below for convenience) which are of interest to SACM. Specifically:

o Endpoint

o Software Component

o Hardware Component

o Identity

o Guidance

o Evaluation Results

The following figure shows the make up of an Endpoint asset which contains zero or more hardware components and zero or more software components each of which may have zero or more instances running an endpoint at any given time as well as zero or more identities that act on behalf of the endpoint when interfacing with other endpoints, tools, or services. An endpoint may also contain other endpoints in the case of a virtualized environment.

```
        +---------+*______in>_______*+-----+
        |Hardware |                  |!   !|
        |Component|   +---------+    |!   !|
        +---------+   |Software |in> |!   !|
                      |Component|____|!   !|
                      +---------+*  *|!   !|
                          1|         |!   !|
                          *|         |     |       +----------+
                      +---------+    |End- |*_____*| Identity |
                      |Software |in> |point| acts  +----------+
                      |Instance |____|     | for>
                      +---------+*  1|!   !|
                                     |!   !|
                                     |!   !|
                                     |!   !|
                                     |!   !|____
                                     |!   !|0..1|
                                     +-----+    |
                                        |*      |
                                        |_______|
                                           in>
```

Figure 15: Model of an Endpoint

6.1. Asset

As defined in [RFC4949], an asset is 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.

In the scope of SACM, an asset can be composed of other assets. Examples of Assets include: Endpoints, Software, Guidance, or Identity. Furthermore, an asset is not necessarily owned by an organization.

6.2. Endpoint

From [RFC5209], an endpoint is 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, 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.

Physical endpoints are always composites that are composed of hardware components and software components. Virtual endpoints are composed entirely of software components and rely on software components that provide functions equivalent to hardware components.

The SACM architecture differentiates two essential categories of endpoints: Endpoints whose security posture is intended to be assessed (target endpoints) and endpoints that are specifically excluded from endpoint posture assessment (excluded endpoints).

6.3. Hardware Component

Hardware components are the distinguishable physical components that compose an endpoint. The composition of an endpoint can be changed over time by adding or removing hardware components. In essence, every physical endpoint is potentially a composite of multiple hardware components, typically resulting in a hierarchical composition of hardware components. The composition of hardware components is based on interconnects provided by specific hardware types (e.g. mainboard is a hardware type that provides local busses as an interconnect). In general, a hardware component can be distinguished by its serial number.

Examples of a hardware components include: motherboards, network interfaces, graphics cards, hard drives, etc.

6.4. Software Component

A software package installed on an endpoint (including the operating system) as well as a unique serial number if present (e.g. a text editor associated with a unique license key).

It should be noted that this includes both benign and harmful software packages. Examples of benign software components include: applications, patches, operating system kernel, boot loader, firmware, code embedded on a webpage, etc. Examples of malicious software components include: malware, trojans, viruses, etc.

6.4.1. Software Instance

A running instance of the software component (e.g. on a multi-user system, one logged-in user has one instance of a text editor running and another logged-in user has another instance of the same text editor running, or on a single-user system, a user could have multiple independent instances of the same text editor running).

6.5. Identity

Any mechanism that can be used to identify an asset during an authentication process. Examples include usernames, user and device certificates, etc. Note, that this is different than the identity of assets in the context of designation as described in Section 11.1.

6.6. Guidance

Guidance is input instructions to processes and tasks, such as collection or evaluation. Guidance influences the behavior of a SACM component and is considered content of the management plane. Guidance can be manually or automatically generated or provided. Typically, the tasks that provide guidance to SACM components have a low-frequency and tend to be sporadic. A prominent example of guidance are target endpoint profiles,but guidance can have many forms, including:

Configuration, e.g. a SACM component’s name, or a CMDB’s IPv6 address.

Profiles, e.g. a set of expected states for network behavior associated with target endpoints employed by specific users.

Policies, e.g. an interval to refresh the registration of a SACM component, or a list of required capabilities for SACM components in a specific location.

6.6.1. Collection Guidance

A collector may need guidance to govern what it collects and when. Collection Guidance provides instructions for a Collector that specifies which endpoint attributes to collect, when to collect them, and how to collect them. Collection Guidance is composed of Target Endpoint Attribute Guidance, Frequency Guidance, and Method Guidance.

- o Target Endpoint Attribute Guidance: Set of endpoint attributes that are supposed to be collected from a target endpoint. The definition of the set of endpoint attributes is typically based on an endpoint characterization record.

- o Frequency Guidance: Specifies when endpoint attributes are to be collected.

- o Method Guidance: Indicates how endpoint attributes are to be collected.

6.6.2. Evaluation Guidance

An evaluator typically needs guidance to govern what it considers to be a good or bad security posture. Evaluation Guidance provides instructions for an Evaluator that specifies which endpoint attributes to evaluate, the desired state of those endpoint attributes, and any special requirements that enable an Evaluator to determine if the endpoint attributes can be used in the evaluation (e.g. freshness of data, how it was collected, etc.). Evaluation Guidance is composed of Target Endpoint Attribute Guidance, Expected Endpoint Attribute Value Guidance, and Frequency Guidance.

- o Target Endpoint Attribute Guidance: Set of target endpoint attributes that are supposed to be used in an evaluation as well as any requirements on the endpoint attributes. The definition of the set of endpoint attributes is typically based on an endpoint characterization record.

- o Expected Endpoint Attribute Value Guidance: The expected values of the endpoint attributes described in the Target Endpoint Attribute Guidance.

- o Frequency Guidance: Specifies when endpoint attributes are to be evaluated.

- o Method Guidance: Indicates how endpoint attributes are to be collected.

6.6.3. Classification Guidance

A SACM Component carrying out the Target Endpoint Classification Task may need guidance on how to classify an endpoint. Specifically, how to associate endpoint classes with a specific target endpoint characterization record. Target Endpoint Classes function as guidance for collection, evaluation, remediation and security posture assessment in general. Classification Guidance is composed of Target Endpoint Attribute Guidance and Class Guidance.

- o Target Endpoint Attribute Guidance: Set of target endpoint attributes that are supposed to be used to identify the endpoint characterization record.

- o Class Guidance: A list of target endpoint classes that are to be associated with the identified target endpoint characterization record.

6.6.4. Storage Guidance

An SACM Component typically needs guidance to govern what information it should store and where. Storage Guidance provides instructions for a SACM Component that specifies which security automation information should be stored, for how long, and on which endpoint. Storage Guidance is composed of Target Endpoint Attribute Guidance, Expected Security Automation Information Guidance, and Retention Guidance.

- o Target Endpoint Attribute Guidance: Set of target endpoint attributes that are supposed to be used to identify the endpoint where the security automation information is to be stored.

- o Expected Security Automation Information Guidance: The security automation information that is expected to be stored (guidance, collected posture attributes, results, etc.).

- o Retention Guidance: Specifies how long the security automation information should be stored.

6.6.5. Evaluation Results

Evaluation Results are the output of comparing the actual state of an endpoint against the expected state of an endpoint. In addition to the actual results of the comparison, Evaluation Results should include the Evaluation Guidance and actual target endpoint attributes values used to perform the evaluation.

7. Information Model Elements

This section defines the specific Information Elements and relationships that will be implemented by data models and transported between SACM Components.

7.1. sacmStatement

```
elementId: TBD
name: sacmStatement
dataType: orderedList
status: current
description: Associates SACM Statement Metadata
which provides data origin information about
the providing SACM Component with one or more
SACM Content Elements that contain security
automation information.
structure: orderedList(sacmStatementMetadata,
                       sacmContentElement+)
```

7.2. sacmStatementMetadata

```
elementId: TBD
name: sacmStatementMetadata
dataType: orderedList
status: current
description: Contains IEs that provide
information about the data origin of the
providing SACM Component as well as the
information necessary for other SACM
Components to understand the type of
security automation information in the
SACM Statement's SACM Content Element(s).
structure: orderedList(publicationTimestamp,
                       dataOrigin, anyIE*)
```

7.3. sacmContentElement

```
elementId: TBD
name: sacmContentElement
dataType: list
status: current
description: Associates SACM Content Element
Metadata which provides information about the
data source and type of security automation
information with the actual security automation
information.
structure: TODO
```

7.4. sacmContentElementMetadata

```
elementId: TBD
name: sacmContentElementMetadata
dataType: orderedList
status: current
description: Contains IEs that provide
information about the data source and type of
security automation information such that other
SACM Components are able to parse and understand
the security automation information contained
within the SACM Statement's SACM Content Element(s).
structure: orderedList(collectionTimestamp,
                       targetEndpoint, anyIE*)
```

7.5. targetEndpoint

```
elementId: TBD
name: targetEndpoint
dataType: category
status: current
description: Information that identifies a target
        endpoint on the network. This may be a set of
attributes that can be used to identify an endpoint
on the network or a label that is unique to a SACM
domain.
        structure: category(targetEndpointIdentifier |
                            targetEndpointLabel)
```

7.6. targetEndpointIdentifier

```
elementId: TBD
name: targetEndpointIdentifier
dataType: list
status: current
description: A set of attributes that uniquely
            identify a target endpoint on the network.
structure: list(anyIE+)
```

7.7. targetEndpointLabel

```
elementId: TBD
name: targetEndpointLabel
dataType: string
status: current
description: A label that uniquely identifies
            a target endpoint on SACM domain.
```

7.8. anyIE

```
elementId: TBD
name: anyIE
dataType: category
status: current
description: This category is a placeholder
            for any information element defined within
            the SACM Information Model. Its purpose is
to provide an extension point in other
            information elements that enable them to
            support the specific needs of an enterprise,
            user, product, or service.
```

7.9. accessPrivilegeType

```
elementId: TBD
name: accessPrivilegeType
dataType: string
status: current
description: A set of types that represent access
privileges (read, write, none, etc.).
```

7.10. accountName

```
elementId: TBD
name: accountName
dataType: string
status: current
description: A label that uniquely identifies an account
that can require some form of (user) authentication to
access.
```

7.11. administrativeDomainType

```
elementId: TBD
name: administrativeDomainType
dataType: string
status: current
description: A label the is supposed to uniquely
identify an administrative domain.
```

7.12. addressAssociationType

elementId: TBD
name: addressAssociationType
dataType: string
status: current
description: A label the is supposed to uniquely identify an administrative domain.

7.13. addressMaskValue

elementId: TBD
name: addressMaskValue
dataType: string
status: current
description: A value that expresses a generic address subnetting bitmask.

7.14. addressType

elementId: TBD
name: addressType
dataType: string
status: current
description: A set of types that specifies the type of address that is expressed in an address subject (e.g. ethernet, modbus, zigbee).

7.15. addressValue

elementId: TBD
name: addressValue
dataType: string
status: current
description: A value that expresses a generic network address.

7.16. applicationComponent

elementId: TBD
name: applicationComponent
dataType: string
status: current
description: A label that references a "sub"-application that is part of the application (e.g. an add-on, a cipher-suite, a library).

7.17. applicationLabel

elementId: TBD
name: applicationLabel
dataType: string
status: current
description: A label that is supposed to uniquely reference an application.

7.18. applicationType

elementId: TBD
name: applicationType
dataType: string
status: current
description: A set of types (FIXME maybe a finite set is not realistic here - value not enumerator?) that identifies the type of (user-space) application (e.g. text-editor, policy-editor, service-client, service-server, calendar, rouge-like RPG).

7.19. applicationManufacturer

elementId: TBD
name: applicationManufacturer
dataType: string
status: current
description: The name of the vendor that created the application.

7.20. authenticator

elementId: TBD
name: authenticator
dataType: string
status: current
description: A label that references a SACM component that can authenticate target endpoints (can be used in a target-endpoint subject to express that the target endpoint was authenticated by that SACM component.

7.21. authenticationType

elementId: TBD
name: authenticationType
dataType: string
status: current
description: A set of types that express which type of authentication was used to enable a network interaction/connection.

7.22. birthdate

elementId: TBD
name: birthdate
dataType: string
status: current
description: A label for the registered day of birth of a natural person (e.g. the date of birth of a person as an ISO date string).
references: http://rs.tdwg.org/ontology/voc/Person#birthdate

7.23. bytesReceived

elementId: TBD
name: bytesReceived
dataType: string
status: current
description: A value that represents a number of octets received on a network interface.

7.24. bytesReceived

elementId: TBD
name: bytesReceived
dataType: string
status: current
description: A value that represents the number of octets received on a network interface.

7.25. bytesSent

elementId: TBD
name: bytesSent
dataType: string
status: current
description: A value that represents the number of octets sent on a network interface.

7.26. certificate

elementId: TBD
name: certificate
dataType: string
status: current
description: A value that expresses a certificate that can be collected from a target endpoint.

7.27. collectionTaskType

elementId: TBD
name: collectionTaskType
dataType: string
status: current
description: A set of types that defines how collected SACM content was acquired (e.g. network-observation, remote-acquisition, self-reported, derived, authority, verified).

7.28. confidence

elementId: TBD
name: confidence
dataType: string
status: current
description: A representation of the subjective probability that the assessed value is correct. If no confidence value is given, it is assumed that the confidence is 1. Acceptable values are between 0 and 1.

7.29. contentAction

elementId: TBD
name: contentAction
dataType: string
status: current
description: A set of types that express a type of action (e.g. add, delete, update). It can be associated, for instance, with an event subject or with a network observation.

7.30. countryCode

elementId: TBD
name: countryCode
dataType: string
status: current
description: A set of types according to ISO 3166-1.

7.31. dataOrigin

elementId: TBD
name: dataOrigin
dataType: string
status: current
description: A label that uniquely identifies a SACM component in and across SACM domains.

7.32. dataSource

elementId: TBD
name: dataSource
dataType: string
status: current
description: A label that is supposed to uniquely identify the data source (e.g. a target endpoint or sensor) that provided an initial endpoint attribute record.

7.33. default-depth

elementId: TBD
name: default-depth
dataType: string
status: current
description: A value that expresses how often a circular reference of subject is allowed to repeat, or how deep a recursive nesting may occur, respectively.

7.34. discoverer

elementId: TBD
name: discoverer
dataType: string
status: current
description: A label that refers to the SACM component that discovered a target endpoint (can be used in a target-endpoint subject to express, for example, that the target endpoint was authenticated by that SACM component).

7.35. emailAddress

elementId: TBD
name: emailAddress
dataType: string
status: current
description: A value that expresses an email-address.

7.36. eventType

elementId: TBD
name: eventType
dataType: string
status: current
description: a set of types that define the categories of an event (e.g. access-level-change, change-of-privilege, change-of-authorization, environmental-event, or provisioning-event).

7.37. eventThreshold

elementId: TBD
name: eventThreshold
dataType: string
status: current
description: If applicable, a value that can be included in an event subject to indicate what numeric threshold value was crossed to trigger that event.

7.38. eventThresholdName

elementId: TBD
name: eventThresholdName
dataType: string
status: current
description: If an event is created due to a crossed threshold, the threshold might have a name associated with it that can be expressed via this value.

7.39. eventTrigger

elementId: TBD
name: eventTrigger
dataType: string
status: current
description: This value is used to express more complex trigger conditions that may cause the creation of an event.

7.40. firmwareId

elementId: TBD
name: firmwareId
dataType: string
status: current
description: A label that represents the BIOS or firmware ID of a specific target endpoint.

7.41. hostName

elementId: TBD
name: hostName
dataType: string
status: current
description: A label typically associated with an endpoint, but, not always intended to be unique given scope.

7.42. interfaceLabel

elementId: TBD
name: interfaceLabel
dataType: string
status: current
description: A unique label that can be used to reference a network interface.

7.43. ipv6AddressSubnetMask

elementId: TBD
name: ipv6AddressSubnetMask
dataType: string
status: current
description: An IPv6 subnet bitmask.

7.44. ipv6AddressSubnetMaskCidrNotation

elementId: TBD
name: ipv6AddressSubnetMaskCidrNotation
dataType: string
status: current
description: An IPv6 subnet bitmask in CIDR notation.

7.45. ipv6AddressValue

elementId: TBD
name: ipv6AddressValue
dataType: ipv6Address
status: current
description: An IPv6 subnet bitmask in CIDR notation.

7.46. ipv4AddressSubnetMask

elementId: TBD
name: ipv4AddressSubnetMask
dataType: string
status: current
description: An IPv4 subnet bitmask.

7.47. ipv4AddressSubnetMaskCidrNotation

elementId: TBD
name: ipv4AddressSubnetMaskCidrNotation
dataType: string
status: current
description: An IPv4 subnet bitmask in CIDR notation.

7.48. ipv4AddressValue

elementId: TBD
name: ipv4AddressValue
dataType: ipv4Address
status: current
description: An IPv4 address value.

7.49. layer2InterfaceType

elementId: TBD
name: layer2InterfaceType
dataType: string
status: current
description: A set of types referenced by IANA ifType.

7.50. layer4PortAddress

elementId: TBD
name: layer4PortAddress
dataType: unsigned32
status: current
description: A layer 4 port address
typically associated with TCP and UDP
protocols.

7.51. layer4Protocol

elementId: TBD
name: layer4Protocol
dataType: string
status: current
description: A set of types that express a layer 4
protocol (e.g. UDP or TCP).

7.52. locationName

elementId: TBD
name: locationName
dataType: string
status: current
description: A value that represents a named region of
physical space.

7.53. networkZoneLocation

elementId: TBD
name: networkZoneLocation
dataType: string
status: current
description: The zone location of an endpoint on the
network (e.g. internet, enterprise DMZ,
enterprise WAN, enclave DMZ, enclave).

7.54. layer2NetworkLocation

elementId: TBD
name: layer2NetworkLocation
dataType: string
status: current
description: The location of a layer-2 interface on
the network (e.g. link-layer neighborhood,
shared broadcast domain).

7.55. layer3NetworkLocation

elementId: TBD
name: layer3NetworkLocation
dataType: string
status: current
description: The location of a layer-3 interface on the network (e.g. next-hop routing neighbor).

7.56. macAddressValue

elementId: TBD
name: macAddressValue
dataType: string
status: current
description: A value that expresses an Ethernet address.

7.57. methodLabel

elementId: TBD
name: methodLabel
dataType: string
status: current
description: A label that references a specific method registered and used in a SACM domain (e.g. method to match and re-identify target endpoints via identifying attributes).

7.58. methodRepository

elementId: TBD
name: methodRepository
dataType: string
status: current
description: A label that references a SACM component methods can be registered at and that can provide guidance in the form of registered methods to other SACM components.

7.59. networkAccessLevelType

elementId: TBD
name: networkAccessLevelType
dataType: string
status: current
description: A set of types that express categories of network access-levels (e.g. block, quarantine, etc.).

7.60. networkId

elementId: TBD
name: networkId
dataType: string
status: current
description: Most networks such as AS, OSBF domains, or VLANs can have an ID.

7.61. networkInterfaceName

elementId: TBD
name: networkInterfaceName
dataType: string
status: current
description: A label that uniquely identifies an interface associated with a distinguishable endpoint.

7.62. networkLayer

elementId: TBD
name: networkLayer
dataType: string
status: current
description: A set of layers that expresses the specific network layer an interface operates on.

7.63. networkName

elementId: TBD
name: networkName
dataType: string
status: current
description: A label that is associated with a network. Some networks, for example, effective layer2-broadcast-domains are difficult to "grasp" and therefore quite difficult to name.

7.64. organizationId

elementId: TBD
name: organizationId
dataType: string
status: current
description: A label that uniquely identifies an organization via a PEN.

7.65. patchId

```
elementId: TBD
name: patchId
dataType: string
status: current
description: A label that uniquely identifies a specific
             software patch.
```

7.66. patchName

```
elementId: TBD
name: patchName
dataType: string
status: current
description: The vendor’s name of a software patch.
```

7.67. personFirstName

```
elementId: TBD
name: personFirstName
dataType: string
status: current
description: The first name of a natural person.
```

7.68. personLastName

```
elementId: TBD
name: personLastName
dataType: string
status: current
description: The last name of a natural person.
```

7.69. personMiddleName

```
elementId: TBD
name: personMiddleName
dataType: string
status: current
description: The middle name of a natural person.
```

7.70. phoneNumber

elementId: TBD
name: phoneNumber
dataType: string
status: current
description: A label that expresses the U.S. national
phone number (e.g. pattern value="((\d{3}))?\d{3}-\d{4}").

7.71. phoneNumberType

elementId: TBD
name: phoneNumberType
dataType: string
status: current
description: A set of types that express the type of
a phone number (e.g. DSN, Fax, Home, Mobile, Pager,
Secure, Unsecure, Work, Other).

7.72. privilegeName

elementId: TBD
name: privilegeName
dataType: string
status: current
description: The attribute name of the privilege
represented as an AVP.

7.73. privilegeValue

elementId: TBD
name: privilegeValue
dataType: string
status: current
description: The value content of the privilege
represented as an AVP.

7.74. protocol

elementId: TBD
name: protocol
dataType: string
status: current
description: A set of types that defines specific
protocols above layer 4 (e.g. http, https, dns, ipp,
or unknown).

7.75. publicKey

elementId: TBD
name: publicKey
dataType: string
status: current
description: The value of a public key (regardless of its method of creation, crypto-system, or signature scheme) that can be collected from a target endpoint.

7.76. relationshipContentElementGuid

elementId: TBD
name: relationshipContentElementGuid
dataType: string
status: current
description: A reference to a specific content element used in a relationship subject.

7.77. relationshipStatementElementGuid

elementId: TBD
name: relationshipStatementElementGuid
dataType: string
status: current
description: A reference to a specific SACM statement used in a relationship subject.

7.78. relationshipObjectLabel

elementId: TBD
name: relationshipObjectLabel
dataType: string
status: current
description: A reference to a specific label used in content (e.g. a te-label or a user-id). This reference is typically used if matching content attribute can be done efficiantly and can also be included in addition to a relationship-content-element-guid reference.

7.79. relationshipType

elementId: TBD
name: relationshipType
dataType: string
status: current
description: A set of types that is in every instance of a relationship subject to highlight what kind of relationship exists between the subject the relationship is included in (e.g. associated_with_user, applies_to_session, seen_on_interface, associated_with_flow, contains_virtual_device).

7.80. roleName

elementId: TBD
name: roleName
dataType: string
status: current
description: A label that references a collection of privileges assigned to a specific entity.

7.81. sessionStateType

elementId: TBD
name: sessionStateType
dataType: string
status: current
description: A set of types a discernible session (an ongoing network interaction) can be in (e.g. Authenticating, Authenticated, Postured, Started, Disconnected).

7.82. statementGuid

elementId: TBD
name: statementGuid
dataType: string
status: current
description: A label that expresses a global unique ID referencing a specific SACM statement that was produced by a SACM component.

7.83. statementType

elementId: TBD
name: statementType
dataType: string
status: current
description: A set of types that define the type of content that is included in a SACM statement (e.g. Observation, DirectoryContent, Correlation, Assessment, Guidance, Event).

7.84. status

elementId: TBD
name: status
dataType: string
status: current
description: A set of types that defines possible result values for a finding in general (e.g. true, false, error, unknown, not applicable, not evaluated).

7.85. subAdministrativeDomain

elementId: TBD
name: subAdministrativeDomain
dataType: string
status: current
description: A label for related child domains an administrative domain can be composed of (used in the subject administrativeDomain).

7.86. subInterfaceLabel

elementId: TBD
name: subInterfaceLabel
dataType: string
status: current
description: A unique label a sub network interface (e.g. a tagged vlan on a trunk) can be referenced with.

7.87. superAdministrativeDomain

elementId: TBD
name: superAdministrativeDomain
dataType: string
status: current
description: a label for related parent domains an administrative domain is part of (used in the subject administrativeDomain).

7.88. superInterfaceLabel

```
elementId: TBD
name: superInterfaceLabel
dataType: string
status: current
description: a unique label a super network interface
             (e.g. a physical interface a tunnel
             interface terminates on) can be referenced
             with.
```

7.89. teAssessmentState

```
elementId: TBD
name: teAssessmentState
dataType: string
status: current
description: a set of types that defines the state of
             assessment of a target-endpoint (e.g.
             in-discovery, discovered, in-classification,
             classified, in-assessment, assessed).
```

7.90. teLabel

```
elementId: TBD
name: teLabel
dataType: string
status: current
description: an identifying label created from a set
             of identifying attributes used to reference
             a specific target endpoint.
```

7.91. teId

```
elementId: TBD
name: teId
dataType: string
status: current
description: an identifying label that is created
             randomly, is supposed to be unique, and
             used to reference a specific target
             endpoint.
```

7.92. timestampType

```
elementId: TBD
name: timestampType
dataType: string
status: current
description: a set of types that express what type of
             action or event happened at that point
             of time (e.g. discovered, classified,
             collected, published). Can be included in
             a generic timestamp subject.
```

7.93. unitsReceived

```
elementId: TBD
name: unitsReceived
dataType: string
status: current
description: a value that represents a number of units
             (e.g. frames, packets, cells or segments)
             received on a network interface.
```

7.94. unitsSent

```
elementId: TBD
name: unitsSent
dataType: string
status: current
description: a value that represents a number of units
             (e.g. frames, packets, cells or segments)
             sent on a network interface.
```

7.95. userDirectory

```
elementId: TBD
name: userDirectory
dataType: string
status: current
description: a label that identifies a specific type
of user-directory (e.g. ldap, active-directory,
local-user).
```

7.96. sacmUserId

```
elementId: TBD
name: sacmUserId
dataType: string
status: current
description: a label that references a specific user
known in a SACM domain.
```

7.97. webSite

elementId: TBD
name: webSite
dataType: string
status: current
description: a URI that references a web-site.

7.98. WGS84Longitude

elementId: TBD
name: WGS84Longitude
dataType: float64
status: current
description: a label that represents WGS 84 rev 2004 longitude.

7.99. WGS84Latitude

elementId: TBD
name: WGS84Latitude
dataType: float64
status: current
description: a label that represents WGS 84 rev 2004 latitude.

7.100. WGS84Altitude

elementId: TBD
name: WGS84Altitude
dataType: float64
status: current
description: a label that represents WGS 84 rev 2004 altitude.

7.101. hardwareSerialNumber

elementId: TBD
name: hardwareSerialNumber
dataType: string
status: current
description: A globally unique identifier for a particular piece of hardware assigned by the vendor.

7.102. interfaceName

```
elementId: TBD
name: interfaceName
dataType: string
status: current
description: A short name uniquely describing an
             interface, e.g. "Eth1/0". See [RFC2863]
             for the definition of the ifName object.
```

7.103. interfaceIndex

```
elementId: TBD
name: interfaceIndex
dataType: unsigned32
status: current
description: The index of an interface installed on an endpoint.
             The value matches the value of managed object
             'ifIndex' as defined in [RFC2863]. Note that ifIndex
             values are not assigned statically to an interface
             and that the interfaces may be renumbered every time
             the device's management system is re-initialized,
             as specified in [RFC2863].
```

7.104. interfaceMacAddress

```
elementId: TBD
name: interfaceMacAddress
dataType: macAddress
status: current
description: The IEEE 802 MAC address associated with a network
             interface on an endpoint.
```

7.105. interfaceType

```
elementId: TBD
name: interfaceType
dataType: unsigned32
status: current
description: The type of a network interface. The value matches
             the value of managed object 'ifType' as defined in
             [IANA registry ianaiftype-mib].
```

7.106. interfaceFlags

```
   elementId: TBD
   name: interfaceFlags
   dataType: unsigned16
   status: current
   description: This information element specifies the flags
                associated with a network interface. Possible
                values include:
   structure:
          Up                  ; 0x1   ; Interface is up.
          Broadcast           ; 0x2   ; Broadcast address valid.
          Debug               ; 0x4   ; Turn on debugging.
          Loopback            ; 0x8   ; Is a loopback net.
          Point-to-point      ; 0x10  ; Interface is point-to-point
                                        link.
          No trailers         ; 0x20  ; Avoid use of trailers.
          Resources allocated ; 0x40  ; Resources allocated.
          No ARP              ; 0x80  ; No address resolution protocol.
          Receive all         ; 0x100 ; Receive all packets.
```

7.107. networkInterface

```
   elementId: TBD
   name: networkInterface
   dataType: orderedList
   status: current
   description: Information about a network interface
                installed on an endpoint. The
                following high-level digram
                describes the structure of
                networkInterface information
                element.
   structure: orderedList(interfaceName, interfaceIndex, macAddress,
                          interfaceType, flags)
```

7.108. softwareIdentifier

```
   elementId: TBD
   name: softwareIdentifier
   dataType: string
   status: current
   description: A globally unique identifier for a particular
                software application.
```

7.109. softwareTitle

elementId: TBD
name: softwareTitle
dataType: string
status: current
description: The title of the software application.

7.110. softwareCreator

elementId: TBD
name: softwareCreator
dataType: string
status: current
description: The software developer (e.g., vendor or author).

7.111. simpleSoftwareVersion

elementId: TBD
name: simpleSoftwareVersion
dataType: string
status: current
description: The version string for a software application that conforms to the format of a list of hierarchical non-negative integers separated by a single character delimiter format.

7.112. rpmSoftwareVersion

elementId: TBD
name: rpmSoftwareVersion
dataType: string
status: current
description: The version string for a software application that conforms to the EPOCH:VERSION-RELEASE format.

7.113. ciscoTrainSoftwareVersion

elementId: TBD
name: ciscoTrainSoftwareVersion
dataType: string
status: current
description: The version string for a software application that conforms to the Cisco IOS Train string format.

7.114. softwareVersion

```
   elementId: TBD
   name: softwareVerison
   dataType: category
   status: current
   description: The version of the software application. Software
                applications may be versioned using a number of
                schemas. The following high-level digram describes
                the structure of the softwareVersion information
                element.
   structure: category(simpleSoftwareVersion | rpmSoftwareVersion |
                    ciscoTrainSoftwareVersion)
```

7.115. softwareLastUpdated

```
   elementId: TBD
   name: softwareLastUpdated
   dataType: dateTimeSeconds
   status: current
   description: The date and time when the software instance
                was last updated on the system (e.g., new
                version instlalled or patch applied)
```

7.116. softwareClass

```
        elementId: TBD
        name: softwareClass
        dataType: enumeration
        status: current
        description: The class of the software instance.
        structure:
    Unknown                ; 0x1 ; The class is not known.
    Other                  ; 0x2 ; The class is known, but,
                                   something other than a value
                                   listed in the enumeration.
    Driver                 ; 0x3 ; The class is a device driver.
    Configuration Software ; 0x4 ; The class is configuration
                                   software.
    Application Software   ; 0x5 ; The class is application
                                   software.
    Instrumentation        ; 0x6 ; The class is instrumentation.
    Diagnostic Software    ; 0x8 ; The class is diagnostic
                                   software.
    Operating System       ; 0x9 ; The class is operating
                                   system.
    Middleware             ; 0xA ; The class is middleware.
    Firmware               ; 0xB ; The class is firmware.
    BIOS/FCode             ; 0xC ; The class is BIOS or FCode.
    Support/Service Pack   ; 0xD ; The class is a support or
                                   service pack.
    Software Bundle        ; 0xE ; The class is a software
                                   bundle.
        References: See Classifications of the DMTF
                    CIM_SoftwareIdentity schema.
```

7.117. softwareInstance

```
elementId: TBD
name: softwareInstance
dataType: orderedList
status: current
description: Information about an instance of software
             installed on an endpoint. The following
             high-level digram describes the structure of
             the softwareInstance information element.
structure: orderedList(softwareIdentifier, softwareTitle,
                       softwareCreator, softwareVersion,
                       softwareLastUpdated, softwareClass)
```

7.118. globallyUniqueIdentifier

elementId: TBD
name: globallyUniqueIdentifier
dataType: unsigned8
status: current
description: TODO.

7.119. creationTimestamp

elementId: TBD
name: creationTimestamp
dataType: dateTimeSeconds
status: current
description: The date and time when the posture information was created by a SACM Component.

7.120. collectionTimestamp

elementId: TBD
name: collectionTimestamp
dataType: dateTimeSeconds
status: current
description: The date and time when the posture information was collected or observed by a SACM Component.

7.121. publicationTimestamp

elementId: TBD
name: publicationTimestamp
dataType: dateTimeSeconds
status: current
description: The date and time when the posture information was published.

7.122. relayTimestamp

elementId: TBD
name: relayTimestamp
dataType: dateTimeSeconds
status: current
description: The date and time when the posture information was relayed to another SACM Component.

7.123. storageTimestamp

```
elementId: TBD
name: storageTimestamp
dataType: dateTimeSeconds
status: current
description: The date and time when the posture
             information was stored in a Repository.
```

7.124. type

```
elementId: TBD
name: type
dataType: enumeration
status: current
description: The type of data model use to represent
             some set of endpoint information. The following
             table lists the set of data models supported by SACM.
structure: TBD
```

7.125. protocolIdentifier

```
elementId: TBD
name: protocolIdentifier
dataType: unsigned8
status: current
description: The value of the protocol number in the IP packet
             header. The protocol number identifies the IP packet
             payload type. Protocol numbers are defined in the
             IANA Protocol Numbers registry.

             In Internet Protocol version 4 (IPv4), this is
             carried in the Protocol field.  In Internet Protocol
             version 6 (IPv6), this is carried in the Next Header
             field in the last extension header of the packet.
```

7.126. sourceTransportPort

```
elementId: TBD
name: sourceTransportPort
dataType: unsigned16
status: current
description: The source port identifier in the transport header.
             For the transport protocols UDP, TCP, and SCTP, this
             is the source port number given in the respective
             header.  This field MAY also be used for future
             transport protocols that have 16-bit source port
             identifiers.
```

7.127. sourceIPv4PrefixLength

elementId: TBD
name: sourceIPv4PrefixLength
dataType: unsigned8
status: current
description: The number of contiguous bits that are relevant in the sourceIPv4Prefix Information Element.

7.128. ingressInterface

elementId: TBD
name: ingressInterface
dataType: unsigned32
status: current
description: The index of the IP interface where packets of this Flow are being received. The value matches the value of managed object 'ifIndex' as defined in [RFC2863]. Note that ifIndex values are not assigned statically to an interface and that the interfaces may be renumbered every time the device's management system is re-initialized, as specified in [RFC2863].

7.129. destinationTransportPort

elementId: TBD
name: destinationTransportPort
dataType: unsigned16
status: current
description: The destination port identifier in the transport header. For the transport protocols UDP, TCP, and SCTP, this is the destination port number given in the respective header. This field MAY also be used for future transport protocols that have 16-bit destination port identifiers.

7.130. sourceIPv6PrefixLength

elementId: TBD
name: sourceIPv6PrefixLength
dataType: unsigned8
status: current
description: The number of contiguous bits that are relevant in the sourceIPv6Prefix Information Element.

7.131. sourceIPv4Prefix

elementId: TBD
name: sourceIPv4Prefix
dataType: ipv4Address
status: current
description: IPv4 source address prefix.

7.132. destinationIPv4Prefix

elementId: TBD
name: destinationIPv4Prefix
dataType: ipv4Address
status: current
description: IPv4 destination address prefix.

7.133. sourceMacAddress

elementId: TBD
name: sourceMacAddress
dataType: macAddress
status: current
description: The IEEE 802 source MAC address field.

7.134. ipVersion

elementId: TBD
name: ipVersion
dataType: unsigned8
status: current
description: The IP version field in the IP packet header.

7.135. interfaceDescription

elementId: TBD
name: interfaceDescription
dataType: string
status: current
description: The description of an interface, e.g. "FastEthernet 1/0" or "ISP connection".

7.136. applicationDescription

elementId: TBD
name: applicationDescription
dataType: string
status: current
description: Specifies the description of an application.

7.137. applicationId

elementId: TBD
name: applicationId
dataType: octetArray
status: current
description: Specifies an Application ID per [RFC6759].

7.138. applicationName

elementId: TBD
name: applicationName
dataType: string
status: current
description: Specifies the name of an application.

7.139. exporterIPv4Address

elementId: TBD
name: exporterIPv4Address
dataType: ipv4Address
status: current
description: The IPv4 address used by the Exporting Process. This is used by the Collector to identify the Exporter in cases where the identity of the Exporter may have been obscured by the use of a proxy.

7.140. exporterIPv6Address

elementId: TBD
name: exporterIPv6Address
dataType: ipv6Address
status: current
description: The IPv6 address used by the Exporting Process. This is used by the Collector to identify the Exporter in cases where the identity of the Exporter may have been obscured by the use of a proxy.

7.141. portId

```
   elementId: TBD
   name: portId
   dataType: unsigned32
   status: current
   description: An identifier of a line port that is unique per
                IPFIX Device hosting an Observation Point.
                Typically, this Information Element is used for
                limiting the scope of other Information Elements.
```

7.142. templateId

```
   elementId: TBD
   name: templateId
   dataType: unsigned16
   status: current
   description: An identifier of a Template that is locally unique
                within a combination of a Transport session and an
                Observation Domain.

                Template IDs 0-255 are reserved for Template Sets,
                Options Template Sets, and other reserved Sets yet
                to be created. Template IDs of Data Sets are
                numbered from 256 to 65535.

                Typically, this Information Element is used for
                limiting the scope of other Information Elements.
                Note that after a re-start of the Exporting Process
                Template identifiers may be re-assigned.
```

7.143. collectorIPv4Address

```
   elementId: TBD
   name: collectorIPv4Address
   dataType: ipv4Address
   status: current
   description: An IPv4 address to which the Exporting Process sends
                Flow information.
```

7.144. collectorIPv6Address

```
   elementId: TBD
   name: collectorIPv6Address
   dataType: ipv6Address
   status: current
   description: An IPv6 address to which the Exporting Process sends
                Flow information.
```

7.145. informationElementIndex

elementId: TBD
name: informationElementIndex
dataType: unsigned16
status: current
description: A zero-based index of an Information Element referenced by informationElementId within a Template referenced by templateId; used to disambiguate scope for templates containing multiple identical Information Elements.

7.146. informationElementId

elementId: TBD
name: informationElementId
dataType: unsigned16
status: current
description: This Information Element contains the ID of another Information Element.

7.147. informationElementDataType

elementId: TBD
name: informationElementDataType
dataType: unsigned8
status: current
description: A description of the abstract data type of an IPFIX information element.These are taken from the abstract data types defined in section 3.1 of the IPFIX Information Model [RFC5102]; see that section for more information on the types described in the informationElementDataType sub-registry.

These types are registered in the IANA IPFIX Information Element Data Type subregistry. This subregistry is intended to assign numbers for type names, not to provide a mechanism for adding data types to the IPFIX Protocol, and as such requires a Standards Action [RFC5226] to modify.

7.148. informationElementDescription

elementId: TBD
name: informationElementDescription
dataType: string
status: current
description: A UTF-8 [RFC3629] encoded Unicode string containing a human-readable description of an Information Element. The content of the informationElementDescription MAY be annotated with one or more language tags [RFC4646], encoded in-line [RFC2482] within the UTF-8 string, in order to specify the language in which the description is written. Description text in multiple languages MAY tag each section with its own language tag; in this case, the description information in each language SHOULD have equivalent meaning. In the absence of any language tag, the "i-default" [RFC2277] language SHOULD be assumed. See the Security Considerations section for notes on string handling for Information Element type records.

7.149. informationElementName

elementId: TBD
name: informationElementName
dataType: string
status: current
description: A UTF-8 [RFC3629] encoded Unicode string containing the name of an Information Element, intended as a simple identifier. See the Security Considerations section for notes on string handling for Information Element type records.

7.150. informationElementRangeBegin

elementId: TBD
name: informationElementRangeBegin
dataType: unsigned64
status: current
description: Contains the inclusive low end of the range of acceptable values for an Information Element.

7.151. informationElementRangeEnd

elementId: TBD
name: informationElementRangeEnd
dataType: unsigned64
status: current
description: Contains the inclusive high end of the range of acceptable values for an Information Element.

7.152. informationElementSemantics

elementId: TBD
name: informationElementSemantics
dataType: unsigned8
status: current
description: A description of the semantics of an IPFIX Information Element. These are taken from the data type semantics defined in section 3.2 of the IPFIX Information Model [RFC5102]; see that section for more information on the types defined in the informationElementSemantics sub-registry. This field may take the values in Table ; the special value 0x00 (default) is used to note that no semantics apply to the field; it cannot be manipulated by a Collecting Process or File Reader that does not understand it a priori.

These semantics are registered in the IANA IPFIX Information Element Semantics subregistry. This subregistry is intended to assign numbers for semantics names, not to provide a mechanism for adding semantics to the IPFIX Protocol, and as such requires a Standards Action [RFC5226] to modify.

7.153. informationElementUnits

```
   elementId: TBD
   name: informationElementUnits
   dataType: unsigned16
   status: current
   description: A description of the units of an IPFIX Information
                Element.  These correspond to the units implicitly
                defined in the Information Element definitions in
                section 5 of the IPFIX Information Model [RFC5102];
                see that section for more information on the types
                described in the informationElementsUnits
                sub-registry. This field may take the values in
                Table 3 below; the special value 0x00 (none) is
                used to note that the field is unitless.

                These types are registered in the IANA IPFIX
                Information Element Units subregistry; new types
                may be added on a First Come First Served [RFC5226]
                basis.
```

7.154. applicationCategoryName

```
   elementId: TBD
   name: applicationCategoryName
   dataType: string
   status: current
   description: An attribute that provides a first level
                categorization for each Application ID.
```

7.155. mibObjectValueInteger

```
   elementId: TBD
   name: mibObjectValueInteger
   dataType: signed64
   status: current
   description: An IPFIX Information Element which denotes that the
                integer value of a MIB object will be exported.
                The MIB Object Identifier ("mibObjectIdentifier")
                for this field MUST be exported in a MIB Field
                Option or via another means.  This Information
                Element is used for MIB objects with the Base
                Syntax of Integer32 and INTEGER with IPFIX Reduced
                Size Encoding used as required. The value is
                encoded as per the standard IPFIX Abstract Data Type
                of signed64.
```

7.156. mibObjectValueOctetString

elementId: TBD
name: mibObjectValueOctetString
dataType: octetArray
status: current
description: An IPFIX Information Element which denotes that an Octet String or Opaque value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of OCTET STRING and Opaque. The value is encoded as per the standard IPFIX Abstract Data Type of octetArray.

7.157. mibObjectValueOID

elementId: TBD
name: mibObjectValueOID
dataType: octetArray
status: current
description: An IPFIX Information Element which denotes that an Object Identifier or OID value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of OBJECT IDENTIFIER. Note - In this case the "mibObjectIdentifier" will define which MIB object is being exported while the value contained in this Information Element will be an OID as a value. The mibObjectValueOID Information Element is encoded as ASN.1/BER [BER] in an octetArray.

7.158. mibObjectValueBits

elementId: TBD
name: mibObjectValueBits
dataType: octetArray
status: current
description: An IPFIX Information Element which denotes that a set of Enumerated flags or bits from a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of BITS. The flags or bits are encoded as per the standard IPFIX Abstract Data Type of octetArray, with sufficient length to accommodate the required number of bits. If the number of bits is not an integer multiple of octets then the most significant bits at end of the octetArray MUST be set to zero.

7.159. mibObjectValueIPAddress

elementId: TBD
name: mibObjectValueIPAddress
dataType: ipv4Address
status: current
description: An IPFIX Information Element which denotes that the IPv4 Address of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of IPaddress. The value is encoded as per the standard IPFIX Abstract Data Type of ipv4Address.

7.160. mibObjectValueCounter

elementId: TBD
name: mibObjectValueCounter
dataType: unsigned64
status: current
description: An IPFIX Information Element which denotes that the counter value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of Counter32 or Counter64 with IPFIX Reduced Size Encoding used as required. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64.

7.161. mibObjectValueGauge

elementId: TBD
name: mibObjectValueGauge
dataType: unsigned32
status: current
description: An IPFIX Information Element which denotes that the Gauge value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of Gauge32. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64. This value will represent a non-negative integer, which may increase or decrease, but shall never exceed a maximum value, nor fall below a minimum value.

7.162. mibObjectValueTimeTicks

elementId: TBD
name: mibObjectValueTimeTicks
dataType: unsigned32
status: current
description: An IPFIX Information Element which denotes that the TimeTicks value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of TimeTicks. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned32.

7.163. mibObjectValueUnsigned

elementId: TBD
name: mibObjectValueUnsigned
dataType: unsigned64
status: current
description: An IPFIX Information Element which denotes that an unsigned integer value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base Syntax of unsigned64 with IPFIX Reduced Size Encoding used as required. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64.

7.164. mibObjectValueTable

elementId: TBD
name: mibObjectValueTable
dataType: orderedList
status: current
description: An IPFIX Information Element which denotes that a complete or partial conceptual table will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with a SYNTAX of SEQUENCE. This is encoded as a subTemplateList of mibObjectValue Information Elements. The template specified in the subTemplateList MUST be an Options Template and MUST include all the Objects listed in the INDEX clause as Scope Fields.
structure: orderedList(mibObjectValueRow+)

7.165. mibObjectValueRow

```
elementId: TBD
name: mibObjectValueRow
dataType: orderedList
status: current
description: An IPFIX Information Element which denotes that a
             single row of a conceptual table will be exported.
             The MIB Object Identifier ("mibObjectIdentifier")
             for this field MUST be exported in a MIB Field
             Option or via another means.  This Information
             Element is used for MIB objects with a SYNTAX of
             SEQUENCE.  This is encoded as a subTemplateList of
             mibObjectValue Information Elements.  The
             subTemplateList exported MUST contain exactly one
             row (i.e., one instance of the subtemplate).  The
             template specified in the subTemplateList MUST be
             an Options Template and MUST include all the
             Objects listed in the INDEX clause as Scope Fields.
structure:   orderedList(mibObjectValue+)
```

7.166. mibObjectIdentifier

```
elementId: TBD
name: mibObjectIdentifier
dataType: octetArray
status: current
description: An IPFIX Information Element which denotes that a
             MIB Object Identifier (MIB OID) is exported in the
             (Options) Template Record.  The mibObjectIdentifier
             Information Element contains the OID assigned to
             the MIB Object Type Definition encoded as
             ASN.1/BER [BER].
```

7.167. mibSubIdentifier

```
elementId: TBD
name: mibSubIdentifier
dataType: unsigned32
status: current
description: A non-negative sub-identifier of an Object
             Identifier (OID).
```

7.168. mibIndexIndicator

```
   elementId: TBD
   name: mibIndexIndicator
   dataType: unsigned64
   status: current
   description: This set of bit fields is used for marking the
                Information Elements of a Data Record that serve as
                INDEX MIB objects for an indexed Columnar MIB
                object.  Each bit represents an Information Element
                in the Data Record with the n-th bit representing
                the n-th Information Element.  A bit set to value 1
                indicates that the corresponding Information Element
                is an index of the Columnar Object represented by
                the mibFieldValue.  A bit set to value 0 indicates
                that this is not the case.

                If the Data Record contains more than 64
                Information Elements, the corresponding Template
                SHOULD be designed such that all INDEX
                Fields are among the first 64 Information Elements,
                because the mibIndexIndicator only contains 64 bits.
                If the Data Record contains less than 64
                Information Elements, then the extra bits in the
                mibIndexIndicator for which no corresponding
                Information Element exists MUST have the value 0,
                and must be disregarded by the Collector.  This
                Information Element may be exported with
                IPFIX Reduced Size Encoding.
```

7.169. mibCaptureTimeSemantics

elementId: TBD
name: mibCaptureTimeSemantics
dataType: unsigned8
status: current
description: Indicates when in the lifetime of the flow the MIB value was retrieved from the MIB for a mibObjectIdentifier. This is used to indicate if the value exported was collected from the MIB closer to flow creation or flow export time and will refer to the Timestamp fields included in the same record. This field SHOULD be used when exporting a mibObjectValue that specifies counters or statistics.

If the MIB value was sampled by SNMP prior to the IPFIX Metering Process or Exporting Process retrieving the value (i.e., the data is already stale) and it’s important to know the exact sampling time, then an additional observationTime* element should be paired with the OID using structured data. Similarly, if different mibCaptureTimeSemantics apply to different mibObject elements within the Data Record, then individual mibCaptureTimeSemantics should be paired with each OID using structured data.

Values:
0. undefined
1. begin - The value for the MIB object is captured from the MIB when the Flow is first observed
2. end - The value for the MIB object is captured from the MIB when the Flow ends
3. export - The value for the MIB object is captured from the MIB at export time
4. average - The value for the MIB object is an average of multiple captures from the MIB over the observed life of the Flow

7.170. mibContextEngineID

elementId: TBD
name: mibContextEngineID
dataType: octetArray
status: current
description: A mibContextEngineID that specifies the SNMP engine ID for a MIB field being exported over IPFIX. Definition as per [RFC3411] section 3.3.

7.171. mibContextName

elementId: TBD
name: mibContextName
dataType: string
status: current
description: This Information Element denotes that a MIB Context Name is specified for a MIB field being exported over IPFIX. Reference [RFC3411] section 3.3.

7.172. mibObjectName

elementId: TBD
name: mibObjectName
dataType: string
status: current
description: The name (called a descriptor in [RFC2578] of an object type definition.

7.173. mibObjectDescription

elementId: TBD
name: mibObjectDescription
dataType: string
status: current
description: The value of the DESCRIPTION clause of an MIB object type definition.

7.174. mibObjectSyntax

elementId: TBD
name: mibObjectSyntax
dataType: string
status: current
description: The value of the SYNTAX clause of an MIB object type definition, which may include a Textual Convention or Subtyping. See [RFC2578].

7.175. mibModuleName

elementId: TBD
name: mibModuleName
dataType: string
status: current
description: The textual name of the MIB module that defines a MIB Object.

7.176. interface

elementId: TBD
name: interface
dataType: list
structure: list (interfaceName, hwAddress, inetAddr, netmask)
status: current
description: Represents an interface and its configuration options.

7.177. iflisteners

elementId: TBD
name: iflisteners
dataType: list
structure: list (interfaceName, physicalProtocol, hwAddress, programName, pid, userId)
status: current
description: Stores the results of checking for applications that are bound to an ethernet interface on the system.

7.178. physicalProtocol

```
elementId: TBD
name: physicalProtocol
dataType: enumeration
structure:
ETH_P_LOOP ; 0x1 ; Ethernet loopback packet.
ETH_P_PUP ; 0x2 ; Xerox PUP packet.
ETH_P_PUPAT ; 0x3 ; Xerox PUP Address Transport packet.
ETH_P_IP ; 0x4 ; Internet protocol packet.
ETH_P_X25 ; 0x5 ; CCITT X.25 packet.
ETH_P_ARP ; 0x6 ; Address resolution packet.
ETH_P_BPQ ; 0x7 ; G8BPQ AX.25 ethernet packet.
ETH_P_IEEEPUP ; 0x8 ; Xerox IEEE802.3 PUP packet.
ETH_P_IEEEPUPAT ; 0x9 ; Xerox IEEE802.3 PUP address transport
                        packet.
ETH_P_DEC ; 0xA ; DEC assigned protocol.
ETH_P_DNA_DL ; 0xB ; DEC DNA Dump/Load.
ETH_P_DNA_RC ; 0xC ; DEC DNA Remote Console.
ETH_P_DNA_RT ; 0xD ; DEC DNA Routing.
ETH_P_LAT ; 0xE ; DEC LAT.
ETH_P_DIAG ; 0xF ; DEC Diagnostics.
ETH_P_CUST ; 0x10 ; DEC Customer use.
ETH_P_SCA ; 0x11 ; DEC Systems Comms Arch.
ETH_P_RARP ; 0x12 ; Reverse address resolution packet.
ETH_P_ATALK ; 0x13 ; Appletalk DDP.
ETH_P_AARP ; 0x14 ; Appletalk AARP.
```

```
   ETH_P_8021Q ; 0x15 ; 802.1Q VLAN Extended Header.
   ETH_P_IPX ; 0x16 ; IPX over DIX.
   ETH_P_IPV6 ; 0x17 ; IPv6 over bluebook.
   ETH_P_SLOW ; 0x18 ; Slow Protocol. See 802.3ad 43B.
   ETH_P_WCCP ; 0x19 ; Web-cache coordination protocol.
   ETH_P_PPP_DISC ; 0x1A ; PPPoE discovery messages.
   ETH_P_PPP_SES ; 0x1B ; PPPoE session messages.
   ETH_P_MPLS_UC ; 0x1C ; MPLS Unicast traffic.
   ETH_P_MPLS_MC ; 0x1D ; MPLS Multicast traffic.
   ETH_P_ATMMPOA ; 0x1E ; MultiProtocol Over ATM.
   ETH_P_ATMFATE ; 0x1F ; Frame-based ATM Transport over Ethernet.
   ETH_P_AOE ; 0x20 ; ATA over Ethernet.
   ETH_P_TIPC ; 0x21 ; TIPC.
   ETH_P_802_3 ; 0x22 ; Dummy type for 802.3 frames.
   ETH_P_AX25 ; 0x23 ; Dummy protocol id for AX.25.
   ETH_P_ALL ; 0x24 ; Every packet.
   ETH_P_802_2 ; 0x25 ; 802.2 frames.
   ETH_P_SNAP ; 0x26 ; Internal only.
   ETH_P_DDCMP ; 0x27 ; DEC DDCMP: Internal only
   ETH_P_WAN_PPP ; 0x28 ; Dummy type for WAN PPP frames.
   ETH_P_PPP_MP ; 0x29 ; Dummy type for PPP MP frames.
   ETH_P_PPPTALK ; 0x2A ; Dummy type for Atalk over PPP.
   ETH_P_LOCALTALK ; 0x2B ; Localtalk pseudo type.
   ETH_P_TR_802_2 ; 0x2C ; 802.2 frames.
   ETH_P_MOBITEX ; 0x2D ; Mobitex.
   ETH_P_CONTROL ; 0x2E ; Card specific control frames.
   ETH_P_IRDA ; 0x2F ; Linux-IrDA.
   ETH_P_ECONET ; 0x30 ; Acorn Econet.
   ETH_P_HDLC ; 0x31 ; HDLC frames.
   ETH_P_ARCNET ; 0x32 ; 1A for ArcNet.
                ; 0x33 ; The empty string value is permitted here
                to allow for detailed error reporting.
   status: current
   description: The physical layer protocol used by the AF_PACKET
   socket.
```

7.179. hwAddress

```
   elementId: TBD
   name: hwAddress
   dataType: string
   status: current
   description: The hardware address associated
         with the interface.
```

7.180. programName

elementId: TBD
name: programName
dataType: string
status: current
description: The name of the communicating
      program.

7.181. userId

elementId: TBD
name: userId
dataType: unsigned32
status: current
description: The numeric user id.

7.182. inetlisteningserver

elementId: TBD
name: inetlisteningserver
dataType: list
structure: list (transportProtocol, localAddress,
      localPort, localFullAddress, programName, foreignAddress,
      foreignPort, foreignFullAddress, pid, userId)
status: current
description: Stores the results of checking for network servers
currently active on a system. It holds information pertaining to
a specific protocol-address-port combination.

7.183. transportProtocol

elementId: TBD
name: transportProtocol
dataType: string
status: current
description: The transport-layer
      protocol (tcp or udp).

7.184. localAddress

elementId: TBD
name: localAddress
dataType: ipAddress
status: current
description: This is the IP address being listened to. Note that
the IP address can be IPv4 or IPv6.

7.185. localPort

elementId: TBD
name: localPort
dataType: unsigned32
status: current
description: This is the TCP or UDP port
being listened to.

7.186. localFullAddress

elementId: TBD
name: localFullAddress
dataType: string
status: current
description: The IP address and network port on which the program
listens, including the local address and the local port. Note
that the IP address can be IPv4 or IPv6.

7.187. foreignAddress

elementId: TBD
name: foreignAddress
dataType: ipAddress
status: current
description: The IP address with which the program is
communicating, or with which it will communicate. Note that the
IP address can be IPv4 or IPv6.

7.188. foreignFullAddress

elementId: TBD
name: foreignFullAddress
dataType: ipAddress
status: current
description: The IP address and network port to which the program
is communicating or will accept communications from, including
the foreign address and foreign port. Note that the IP address
can be IPv4 or IPv6.

7.189. selinuxboolean

elementId: TBD
name: selinuxboolean
dataType: list
structure: list (selinuxName, currentStatus,
pendingStatus)
status: current
description: Describes the current and pending status of a
SELinux boolean.

7.190. selinuxName

elementId: TBD
name: selinuxName
dataType: string
status: current
description: The name of the SELinux
boolean.

7.191. currentStatus

elementId: TBD
name: currentStatus
dataType: boolean
status: current
description: Indicates current state of
the specified SELinux boolean.

7.192. pendingStatus

elementId: TBD
name: pendingStatus
dataType: boolean
status: current
description: Indicates the pending
state of the specified SELinux boolean.

7.193. selinuxsecuritycontext

elementId: TBD
name: selinuxsecuritycontext
dataType: list
structure: list (filepath, path, filename, pid,
username, role, domainType, lowSensitivity, lowCategory,
highSensitivity, highCategory, rawlowSensitivity,
rawlowCategory, rawhighSensitivity, rawhighCategory)
status: current
description: Describes the SELinux security
context of a file or process on the local system.

7.194. filepath

elementId: TBD
name: filepath
dataType: string
status: current
description: Specifies the absolute path for a file on the
machine. A directory cannot be specified as a filepath.

7.195. path

elementId: TBD
name: path
dataType: string
status: current
description: Specifies the directory component of
the absolute path to a file on the machine.

7.196. filename

elementId: TBD
name: filename
dataType: string
status: current
description: The name of the file.

7.197. pid

elementId: TBD
name: pid
dataType: unsigned32
status: current
description: The process ID of the
process.

7.198. role

elementId: TBD
name: role
dataType: string
status: current
description: Specifies the types that a process
may transition to (domain transitions).

7.199. domainType

elementId: TBD
name: domainType
dataType: string
status: current
description: Specifies the domain in which the file is accessible or the domain in which a process executes.

7.200. lowSensitivity

elementId: TBD
name: lowSensitivity
dataType: string
status: current
description: Specifies the current sensitivity of a file or process.

7.201. lowCategory

elementId: TBD
name: lowCategory
dataType: string
status: current
description: Specifies the set of categories associated with the low sensitivity.

7.202. highSensitivity

elementId: TBD
name: highSensitivity
dataType: string
status: current
description: Specifies the maximum range for a file or the clearance for a process.

7.203. highCategory

elementId: TBD
name: highCategory
dataType: string
status: current
description: Specifies the set of categories associated with the high sensitivity.

7.204. rawlowSensitivity

elementId: TBD
name: rawlowSensitivity
dataType: string
status: current
description: Specifies the current sensitivity of a file or process but in its raw context.

7.205. rawlowCategory

elementId: TBD
name: rawlowCategory
dataType: string
status: current
description: Specifies the set of categories associated with the low sensitivity but in its raw context.

7.206. rawhighSensitivity

elementId: TBD
name: rawhighSensitivity
dataType: string
status: current
description: Specifies the maximum range for a file or the clearance for a process but in its raw context.

7.207. rawhighCategory

elementId: TBD
name: rawhighCategory
dataType: string
status: current
description: Specifies the set of categories associated with the high sensitivity but in its raw context.

7.208. systemdunitdependency

elementId: TBD
name: systemdunitdependency
dataType: list
structure: list (unit, dependency)
status: current

description: Stores the dependencies of the systemd unit.

7.209. unit

elementId: TBD
name: unit
dataType: string
status: current
description: Refers to the full systemd unit name, which has a form of "$name.$type". For example "cupsd.service". This name is usually also the filename of the unit configuration file.

7.210. dependency

elementId: TBD
name: dependency
dataType: string
status: current
description: Refers to the name of a unit that was confirmed to be a dependency of the given unit.

7.211. systemdunitproperty

elementId: TBD
name: systemdunitproperty
dataType: list
structure: list (unit, property, systemdunitValue)

status: current
description: Stores the properties and values of a systemd unit.

7.212. property

elementId: TBD
name: property
dataType: string
status: current
description: The property associated with a systemd unit.

7.213. systemdunitValue

elementId: TBD
name: systemdunitValue
dataType: string
status: current
description: The value of the property associated with a systemd unit. Exactly one value shall be used for all property types except dbus arrays - each array element shall be represented by one value.

7.214. file

elementId: TBD
name: file
dataType: list
structure: list (filepath, path, filename, fileType, userId, aTime, cTime, mTime, size)
status: current
description: The metadata associated with a file on the endpoint.

7.215. fileType

elementId: TBD
name: fileType
dataType: string
status: current
description: The file’s type (e.g., regular file (regular), directory, named pipe (fifo), symbolic link, socket or block special.)

7.216. groupId

elementId: TBD
name: groupId
dataType: unsigned32
status: current
description: The group owner of the file, by group number.

7.217. aTime

elementId: TBD
name: aTime
dataType: dateTimeSeconds
status: current
description: The time that the file was last accessed.

7.218. cTime

elementId: TBD
name: cTime
dataType: dateTimeSeconds
status: current
description: The time of the last change to the file’s inode.

7.219. mTime

elementId: TBD
name: mTime
dataType: dateTimeSeconds
status: current
description: The time of the last change to
the file's contents.

7.220. size

elementId: TBD
name: size
dataType: unsigned32
status: current
description: This is the size of the file in
bytes.

7.221. suid

elementId: TBD
name: suid
dataType: boolean
status: current
description: Indicates whether the program runs with the uid
(thus privileges) of the file's owner, rather than the calling
user.

7.222. sgid

elementId: TBD
name: sgid
dataType: boolean
status: current
description: Indicates whether the program runs with the gid
(thus privileges) of the file's group owner, rather than the
calling user's group.

7.223. sticky

elementId: TBD
name: sticky
dataType: boolean
status: current
description: Indicates whether users can delete each other's
files in this directory, when said directory is writable by
those users.

7.224. hasExtendedAcl

elementId: TBD
name: hasExtendedAcl
dataType: boolean
status: current
description: Indicates whether the file or directory hasACL permissions applied to it. If a system supports ACLs and the file or directory doesn't have an ACL, or it matches the standard UNIX permissions, the entity will have a status of 'exists' and a value of 'false'. If the system supports ACLs and the file or directory has an ACL, the entity will have a status of 'exists' and a value of 'true'. Lastly, if a system doesn't support ACLs, the entity will have a status of 'does not exist'.

7.225. inetd

elementId: TBD
name: inetd
dataType: list
structure: list (serviceProtocol, serviceName, serverProgram, serverArguments, inetdEndpointType, execAsUser, waitStatus)
status: current
description: Holds information associated with different Internet services.

7.226. serverProgram

elementId: TBD
name: serverProgram
dataType: string
status: current
description: Either the pathname of a server program to be invoked by inetd to perform the requested service, or the value internal if inetd itself provides the service.

7.227. inetdEndpointType

elementId: TBD
name: inetdEndpointType
dataType: enumeration
structure:
stream ; 0x1 ; The stream value is used to describe a stream socket.
dgram ; 0x2 ; The dgram value is used to describe a datagram socket.
raw ; 0x3 ; The raw value is used to describe a raw socket.
seqpacket ; 0x4 ; The seqpacket value is used to describe a sequenced packet socket.
tli ; 0x5 ; The tli value is used to describe all TLI endpoints.
sunrpc_tcp ; 0x6 ; The sunrpc_tcp value is used to describe all SUNRPC TCP endpoints.
sunrpc_udp ; 0x7 ; The sunrpc_udp value is used to describe all SUNRPC UDP endpoints.
; 0x8 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: The endpoint type (aka, socket type) associated with the service.

7.228. execAsUser

elementId: TBD
name: execAsUser
dataType: string
status: current
description: The user id of the user the server program should run under.

7.229. waitStatus

elementId: TBD
name: waitStatus
dataType: enumeration
structure: wait ; 0x1 ; The value of 'wait' specifies that the server that is invoked by inetd will take over the listening socket associated with the service, and once launched, inetd will wait for that server to exit, if ever, before it resumes listening for new service requests.

nowait ; 0x2 ; The value of 'nowait' specifies that the server that is invoked by inetd will not wait for any existing server to finish before taking over the listening socket associated with the service.

; 0x3 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Specifies whether the server that is invoked by inetd will take over the listening socket associated with the service, and whether once launched, inetd will wait for that server to exit, if ever, before it resumes listening for new service requests. The legal values are "wait" or "nowait".

7.230. inetAddr

elementId: TBD
name: inetAddr
dataType: ipAddress
status: current
description: The IP address of the specific interface. Note that the IP address can be IPv4 or IPv6.

7.231. netmask

elementId: TBD
name: netmask
dataType: ipAddress
status: current
description: The bitmask used to calculate the interface's IP network.

7.232. passwordInfo

```
elementId: TBD
name: passwordInfo
dataType: list
structure: list (username, password, userId, groupId, gcos,
      homeDir, loginShell, lastLogin)
status: current
description: Describes user account information for a
      system.
```

7.233. username

```
elementId: TBD
name: username
dataType: string
status: current
description: The name of the user.
```

7.234. password

```
elementId: TBD
name: password
dataType: string
status: current
description: The encrypted version of the
      user's password.
```

7.235. gcos

```
elementId: TBD
name: gcos
dataType: string
status: current
description:
```

7.236. homeDir

```
elementId: TBD
name: homeDir
dataType: string
status: current
description: The user's home
      directory.
```

7.237. loginShell

elementId: TBD
name: loginShell
dataType: string
status: current
description: The user’s shell
program.

7.238. lastLogin

elementId: TBD
name: lastLogin
dataType: unsigned32
status: current
description: The date and time when the
last login occurred.

7.239. process

elementId: TBD
name: process
dataType: list
structure: list (commandLine, pid, ppid, priority, startTime)

status: current
description: Information about a process running on an endpoint.

7.240. commandLine

elementId: TBD
name: commandLine
dataType: string
status: current
description: The string used to start the
process. This includes any parameters that are part of the
command line.

7.241. ppid

elementId: TBD
name: ppid
dataType: unsigned32
status: current
description: The process ID of the process’s
parent process.

7.242. priority

```
elementId: TBD
name: priority
dataType: unsigned32
status: current
description: The scheduling priority with
      which the process runs.
```

7.243. startTime

```
elementId: TBD
name: startTime
dataType: string
status: current
description: The time of day the process
      started.
```

7.244. routingtable

```
elementId: TBD
name: routingtable
dataType: list
structure: list (destination, gateway, flags,
      interfaceName)
status: current
description: Holds information about an individual routing table
entry found in a system's primary routing table.
```

7.245. destination

```
elementId: TBD
name: destination
dataType: ipAddress
status: current
description: The destination IP address
      prefix of the routing table entry.
```

7.246. gateway

```
elementId: TBD
name: gateway
dataType: ipAddress
status: current
description: The gateway of the specified
      routing table entry.
```

7.247. runlevelInfo

elementId: TBD
name: runlevelInfo
dataType: list
structure: list (serviceName, runlevel, start, kill)

status: current
description: Information about the start or kill state of a specified service at a given runlevel.

7.248. runlevel

elementId: TBD
name: runlevel
dataType: string
status: current
description: Specifies the system runlevel associated with a service.

7.249. start

elementId: TBD
name: start
dataType: boolean
status: current
description: Specifies whether the service is scheduled to start at the runlevel.

7.250. kill

elementId: TBD
name: kill
dataType: boolean
status: current
description: Specifies whether the service is scheduled to be killed at the runlevel.

7.251. shadowItem

elementId: TBD
name: shadowItem
dataType: list
structure: list (username, password, chgLst, chgAllow, chgReq, expWarn, expInact, expDate, flags, encryptMethod)
status: current
description:

7.252. chgLst

```
elementId: TBD
name: chgLst
dataType: dateTimeSeconds
status: current
description: The date of the last password
      change.
```

7.253. chgAllow

```
elementId: TBD
name: chgAllow
dataType: unsigned32
status: current
description: Specifies how often in days a
      user may change their password. It can also be thought of
      as the minimum age of a password.
```

7.254. chgReq

```
elementId: TBD
name: chgReq
dataType: unsigned32
status: current
description: Describes how long a user can
      keep a password before the system forces her to change it.
```

7.255. expWarn

```
elementId: TBD
name: expWarn
dataType: unsigned32
status: current
description: Describes how long before
      password expiration the system begins warning the user.
```

7.256. expInact

```
elementId: TBD
name: expInact
dataType: unsigned32
status: current
description: Describes how many days of
      account inactivity the system will wait after a password
      expires before locking the account.
```

7.257. expDate

elementId: TBD
name: expDate
dataType: dateTimeSeconds
status: current
description: Specifies when will the
account’s password expire.

7.258. encryptMethod

elementId: TBD
name: encryptMethod
dataType: enumeration
structure: DES ; 0x1 ; The DES method corresponds to the (none)
prefix.
BSDi ; 0x2 ; The BSDi method corresponds to BSDi modified
DES or the ’_’ prefix.
MD5 ; 0x3 ; The MD5 method corresponds to MD5 for Linux/BSD
or the 1 prefix.
Blowfish ; 0x4 ; The Blowfish method corresponds to Blowfish
(OpenBSD) or the 2 or $2a$ prefixes.
Sun MD5 ; 0x5 ; The Sun MD5 method corresponds to the $md5$
prefix.
SHA-256 ; 0x6 ; The SHA-256 method corresponds to the 5
prefix.
SHA-512 ; 0x7 ; The SHA-512 method corresponds to the 6
prefix. ; 0x8 ; The empty string value is permitted here to
allow for empty elements associated with variable references.
status: current
description: Describes method that is used for hashing
passwords.

7.259. symlink

elementId: TBD
name: symlink
dataType: list
structure: list (symlinkFilepath, canonicalPath)
status: current

description: Identifies the result generated for a symlink.

7.260. symlinkFilepath

```
   elementId: TBD
   name: symlinkFilepath
   dataType: string
   status: current
   description: Specifies the filepath to
         the subject symbolic link file.
```

7.261. canonicalPath

```
   elementId: TBD
   name: canonicalPath
   dataType: string
   status: current
   description: Specifies the canonical
         path for the target of the symbolic link file specified by
         the filepath.
```

7.262. sysctl

```
   elementId: TBD
   name: sysctl
   dataType: list
   structure: list (kernelParameterName, kernelParameterValue+,
         uname, machineClass, nodeName, osName, osRelease,
         osVersion, processorType)
   status: current
   description: Stores
         information retrieved from the local system about a kernel
         parameter and its respective value(s).
```

7.263. kernelParameterName

```
   elementId: TBD
   name: kernelParameterName
   dataType: string
   status: current
   description: The name of a kernel
         parameter that was collected from the local system.
```

7.264. kernelParameterValue

```
   elementId: TBD
   name: kernelParameterValue
   dataType: string
   status: current
   description: The current value(s)
         for the specified kernel parameter on the local system.
```

7.265. uname

elementId: TBD
name: uname
dataType: list
structure: list (machineClass, nodeName, osName, osRelease, osVersion, processorType)
status: current
description: Information about the hardware the machine is running on.

7.266. machineClass

elementId: TBD
name: machineClass
dataType: string
status: current
description: Specifies the machine hardware name.

7.267. nodeName

elementId: TBD
name: nodeName
dataType: string
status: current
description: Specifies the host name.

7.268. osName

elementId: TBD
name: osName
dataType: string
status: current
description: Specifies the operating system name.

7.269. osRelease

elementId: TBD
name: osRelease
dataType: string
status: current
description: Specifies the build version.

7.270. processorType

elementId: TBD
name: processorType
dataType: string
status: current
description: Specifies the processor type.

7.271. internetService

elementId: TBD
name: internetService
dataType: list
structure: list (serviceProtocol, serviceName, flags, noAccess, onlyFrom, port, server, serverArguments, socketType, registeredServiceType, user, wait, disabled)

status: current
description: Holds information associated with Internet services.

7.272. serviceProtocol

elementId: TBD
name: serviceProtocol
dataType: string
status: current
description: Specifies the protocol that is used by the service.

7.273. serviceName

elementId: TBD
name: serviceName
dataType: string
status: current
description: Specifies the name of the service.

7.274. flags

elementId: TBD
name: flags
dataType: string
status: current
description: Specifies miscellaneous settings associated with the service with executing a program.

7.275. noAccess

elementId: TBD
name: noAccess
dataType: string
status: current
description: Specifies the remote hosts to
which the service is unavailable.

7.276. onlyFrom

elementId: TBD
name: onlyFrom
dataType: ipAddress
status: current
description: Specifies the remote hosts to
which the service is available.

7.277. port

elementId: TBD
name: port
dataType: unsigned32
status: current
description: The port entity specifies the port
used by the service.

7.278. server

elementId: TBD
name: server
dataType: string
status: current
description: Specifies the executable that is
used to launch the service.

7.279. serverArguments

elementId: TBD
name: serverArguments
dataType: string
status: current
description: Specifies the arguments
that are passed to the executable when launching the service.

7.280. socketType

elementId: TBD
name: socketType
dataType: string
status: current
description: Specifies the type of socket
that is used by the service. Possible values include: stream,
dgram, raw, or seqpacket.

7.281. registeredServiceType

elementId: TBD
name: registeredServiceType
dataType: enumeration
structure: INTERNAL ; 0x1 ; The INTERNAL type is used to describe
services like echo, chargen, and others whose functionality is
supplied by xinetd itself.
RPC ; 0x2 ; The RPC type is used to describe services that
use remote procedure call ala NFS.
UNLISTED ; 0x3 ; The UNLISTED type is used to describe
services that aren't listed in /etc/protocols or /etc/rpc.
TCPMUX ; 0x4 ; The TCPMUX type is used to describe services
that conform to RFC 1078. This type indiciates that the service
is responsible for handling the protocol handshake.
TCPMUXPLUS ; 0x5 ; The TCPMUXPLUS type is used to describe
services that conform to RFC 1078. This type indicates that
xinetd is responsible for handling the protocol
handshake.
; 0x6 ; The empty string value is permitted here to allow
for detailed error reporting.
status: current

description: Specifies the type of internet service.

7.282. wait

elementId: TBD
name: wait
dataType: boolean
status: current
description: Specifies whether or not the service is single-threaded
or multi-threaded and whether or not xinetd accepts the connection
or the service accepts the connection. A value of 'true' indicates
that the service is single-threaded and the service will accept the
connection. A value of 'false' indicates that the service is multi-
threaded and xinetd will accept the connection.

7.283. disabled

elementId: TBD
name: disabled
dataType: boolean
status: current
description: Specifies whether or not the
service is disabled. A value of 'true' indicates that the
service is disabled and will not start. A value of
'false' indicates that the service is not disabled.

7.284. windowsView

elementId: TBD
name: windowsView
dataType: enumeration
structure: 32_bit ; 0x1 ; Indicates the 32_bit windows view.
64_bit ; 0x2 ; Indicates the 64_bit windows view.
; 0x3 ; The empty string value is permitted here to allow for
empty elements associated with error conditions.
status: current
description: Indicates from which
view (32-bit or 64-bit), the information was collected.
A value of '32_bit' indicates the Item was collected from
the 32-bit view. A value of '64-bit' indicates the Item
was collected from the 64-bit view.

7.285. fileauditedpermissions

elementId: TBD
name: fileauditedpermissions
dataType: list
structure: list (filepath, path, filename,
trusteeSid, trusteeName, auditStandardDelete,
auditStandardReadControl, auditStandardWriteDac,
auditStandardWriteOwner, auditStandardSynchronize,
auditAccessSystemSecurity, auditGenericRead, auditGenericWrite,
auditGenericExecute, auditGenericAll, auditFileReadData,
auditFileWriteData, auditFileAppendData, auditFileReadEa,
auditFileWriteEa, auditFileExecute, auditFileDeleteChild,
auditFileReadAttributes, auditFileWriteAttributes,
windowsView)
status: current
description: Stores the audited access rights of a file that a
system access control list (SACL) structure grants to a specified
trustee. The trustee's audited access rights are determined checking
all access control entries (ACEs) in the SACL.

7.286. trusteeName

elementId: TBD
name: trusteeName
dataType: string
status: current
description: Specifies the trustee name. A
trustee can be a user, group, or program (such as a Windows
service).

7.287. auditStandardDelete

elementId: TBD
name: auditStandardDelete
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is
used to perform audits on all unsuccessful occurrences of
specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel
all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to
perform audits on all successful occurrences of the specified
events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE
is used to perform audits on all successful and unsuccessful
occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for
detailed error reporting.
status: current
description: The right to delete the object.

7.288. auditStandardReadControl

elementId: TBD
name: auditStandardReadControl
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: The right to read the information in the object’s security descriptor, not including the information in the SACL.

7.289. auditStandardWriteDac

elementId: TBD
name: auditStandardWriteDac
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: The right to modify the DACL in the object’s security descriptor.

7.290. auditStandardWriteOwner

elementId: TBD
name: auditStandardWriteOwner
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: The right to change the owner in the object's security descriptor.

7.291. auditStandardSynchronize

elementId: TBD
name: auditStandardSynchronize
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: The right to use the object for synchronization.
This enables a thread to wait until the object is in the signaled state. Some object types do not support this access right.

7.292. auditAccessSystemSecurity

elementId: TBD
name: auditAccessSystemSecurity
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Indicates access to a system access control list (SACL).

7.293. auditGenericRead

elementId: TBD
name: auditGenericRead
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Read access.

7.294. auditGenericWrite

elementId: TBD
name: auditGenericWrite
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Write access.

7.295. auditGenericExecute

elementId: TBD
name: auditGenericExecute
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Execute access.

7.296. auditGenericAll

elementId: TBD
name: auditGenericAll
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Read, write, and execute access.

7.297. auditFileReadData

elementId: TBD
name: auditFileReadData
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to read data from the file.

7.298. auditFileWriteData

elementId: TBD
name: auditFileWriteData
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to write data to the file.

7.299. auditFileAppendData

elementId: TBD
name: auditFileAppendData
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to append data to the file.

7.300. auditFileReadEa

elementId: TBD
name: auditFileReadEa
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to read extended attributes.

7.301. auditFileWriteEa

elementId: TBD
name: auditFileWriteEa
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to write extended attributes.

7.302. auditFileExecute

elementId: TBD
name: auditFileExecute
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to execute a file.

7.303. auditFileDeleteChild

elementId: TBD
name: auditFileDeleteChild
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Right to delete a directory and all the files it contains (its children), even if the files are read-only.

7.304. auditFileReadAttributes

elementId: TBD
name: auditFileReadAttributes
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to read file attributes.

7.305. auditFileWriteAttributes

elementId: TBD
name: auditFileWriteAttributes
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description: Grants the right to change file attributes.

7.306. fileeffectiverights

elementId: TBD
name: fileeffectiverights
dataType: list
structure: list (filepath, path, filename,
trusteeSid, trusteeName, standardDelete, standardReadControl,
standardWriteDac, standardWriteOwner,
standardSynchronize, accessSystemSecurity, genericRead,
genericWrite, genericExecute, genericAll, fileReadData,
fileWriteData, fileAppendData, fileReadEa, fileWriteEa,
fileExecute, fileDeleteChild, fileReadAttributes,
fileWriteAttributes, windowsView)
status: current
description: Stores the effective rights of a file that a discretionary access control list (DACL) structure grants to a specified trustee. The trustee’s effective rights are determined checking all access-allowed and access-denied access control entries (ACEs) in the DACL.

7.307. standardDelete

elementId: TBD
name: standardDelete
dataType: boolean
status: current
description: The right to delete the object.

7.308. standardReadControl

elementId: TBD
name: standardReadControl
dataType: boolean
status: current
description: The right to read the information in the object’s security descriptor, not including the information in the SACL.

7.309. standardWriteDac

elementId: TBD
name: standardWriteDac
dataType: boolean
status: current
description: The right to modify the DACL in the object’s security descriptor.

7.310. standardWriteOwner

elementId: TBD
name: standardWriteOwner
dataType: boolean
status: current
description: The right to change
the owner in the object’s security descriptor.

7.311. standardSynchronize

elementId: TBD
name: standardSynchronize
dataType: boolean
status: current
description: The right to use the
object for synchronization. This enables a thread to wait
until the object is in the signaled state. Some object
types do not support this access right.

7.312. accessSystemSecurity

elementId: TBD
name: accessSystemSecurity
dataType: boolean
status: current
description: Indicates access to
a system access control list (SACL).

7.313. genericRead

elementId: TBD
name: genericRead
dataType: boolean
status: current
description: Read access.

7.314. genericWrite

elementId: TBD
name: genericWrite
dataType: boolean
status: current
description: Write access.

7.315. genericExecute

elementId: TBD
name: genericExecute
dataType: boolean
status: current
description: Execute access.

7.316. genericAll

elementId: TBD
name: genericAll
dataType: boolean
status: current
description: Read, write, and execute
access.

7.317. fileReadData

elementId: TBD
name: fileReadData
dataType: boolean
status: current
description: Grants the right to read
data from the file

7.318. fileWriteData

elementId: TBD
name: fileWriteData
dataType: boolean
status: current
description: Grants the right to write
data to the file.

7.319. fileAppendData

elementId: TBD
name: fileAppendData
dataType: boolean
status: current
description: Grants the right to
append data to the file.

7.320. fileReadEa

elementId: TBD
name: fileReadEa
dataType: boolean
status: current
description: Grants the right to read
extended attributes.

7.321. fileWriteEa

elementId: TBD
name: fileWriteEa
dataType: boolean
status: current
description: Grants the right to write
extended attributes.

7.322. fileExecute

elementId: TBD
name: fileExecute
dataType: boolean
status: current
description: Grants the right to execute
a file.

7.323. fileDeleteChild

elementId: TBD
name: fileDeleteChild
dataType: boolean
status: current
description: Right to delete a
directory and all the files it contains (its children),
even if the files are read-only.

7.324. fileReadAttributes

elementId: TBD
name: fileReadAttributes
dataType: boolean
status: current
description: Grants the right to
read file attributes.

7.325. fileWriteAttributes

elementId: TBD
name: fileWriteAttributes
dataType: boolean
status: current
description: Grants the right to
change file attributes.

7.326. groupInfo

elementId: TBD
name: groupInfo
dataType: list
structure: list (group, username, subgroup)
status: current
description: Specifies the different users and subgroups, that
directly belong to specific groups.

7.327. group

elementId: TBD
name: group
dataType: string
status: current
description: Represents the name of a particular
group.

7.328. subgroup

elementId: TBD
name: subgroup
dataType: string
status: current
description: Represents the name of a
particular subgroup in the specified group.

7.329. groupSidInfo

elementId: TBD
name: groupSidInfo
dataType: list
structure: list (groupSid, userSid, subgroupSid)
status: current
description: Specifies the different users and subgroups, that
directly belong to specific groups
(identified by SID).

7.330. userSidInfo

elementId: TBD
name: userSidInfo
dataType: list
structure: list (userSid, enabled, groupSid, lastLogon)

status: current
description: Specifies the different groups (identified by SID) that a user belongs to.

7.331. userSid

elementId: TBD
name: userSid
dataType: string
status: current
description: Represents the SID of a particular user.

7.332. subgroupSid

elementId: TBD
name: subgroupSid
dataType: string
status: current
description: Represents the SID of a particular subgroup.

7.333. lockoutpolicy

elementId: TBD
name: lockoutpolicy
dataType: list
structure: list (forceLogoff, lockoutDuration, lockoutObservationWindow, lockoutThreshold)
status: current
description: Specifies various attributes associated with lockout information for users and global groups in the security database.

7.334. forceLogoff

elementId: TBD
name: forceLogoff
dataType: unsigned32
status: current
description: Specifies, in seconds, the
amount of time between the end of the valid logon time and
the time when the user is forced to log off the
network.

7.335. lockoutDuration

elementId: TBD
name: lockoutDuration
dataType: unsigned32
status: current
description: Specifies, in seconds,
how long a locked account remains locked before it is
automatically unlocked.

7.336. lockoutObservationWindow

elementId: TBD
name: lockoutObservationWindow
dataType: unsigned32
status: current
description: Specifies the
maximum time, in seconds, that can elapse between any two
failed logon attempts before lockout occurs.

7.337. lockoutThreshold

elementId: TBD
name: lockoutThreshold
dataType: unsigned32
status: current
description: Specifies the number of
invalid password authentications that can occur before an
account is marked "locked out."

7.338. passwordpolicy

elementId: TBD
name: passwordpolicy
dataType: list
structure: list (maxPasswdAge, minPasswdAge,
minPasswdLen, passwordHistLen, passwordComplexity,
reversibleEncryption)
status: current
description: Specifies
policy information associated with passwords.

7.339. maxPasswdAge

elementId: TBD
name: maxPasswdAge
dataType: unsigned32
status: current
description: Specifies, in seconds (from
a DWORD), the maximum allowable password age. A value of
TIMEQ_FOREVER (max DWORD value, 4294967295) indicates
that the password never expires. The minimum valid value
for this element is ONE_DAY (86400). See the
USER_MODALS_INFO_0 structure returned by a call to
NetUserModalsGet().

7.340. minPasswdAge

elementId: TBD
name: minPasswdAge
dataType: unsigned32
status: current
description: Specifies the minimum
number of seconds that can elapse between the time a password
changes and when it can be changed again. A value of
zero indicates that no delay is required between password
updates.

7.341. minPasswdLen

elementId: TBD
name: minPasswdLen
dataType: unsigned32
status: current
description: Specifies the minimum
allowable password length. Valid values for this element are
zero through PWLEN.

7.342. passwordHistLen

elementId: TBD
name: passwordHistLen
dataType: unsigned32
status: current
description: Specifies the length of password history maintained. A new password cannot match any of the previous usrmod0_password_hist_len passwords. Valid values for this element are zero through DEF_MAX_PWHIST.

7.343. passwordComplexity

elementId: TBD
name: passwordComplexity
dataType: boolean
status: current
description: Indicates whether passwords must meet the complexity requirements put forth by the operating system.

7.344. reversibleEncryption

elementId: TBD
name: reversibleEncryption
dataType: boolean
status: current
description: Indicates whether or not passwords are stored using reversible encryption.

7.345. portInfo

elementId: TBD
name: portInfo
dataType: list
structure: list (localAddress, localPort, transportProtocol, pid, foreignAddress, foreignPort)
status: current
description: Information about open listening ports.

7.346. foreignPort

elementId: TBD
name: foreignPort
dataType: string
status: current
description: The TCP or UDP port to which the program communicates.

7.347. printereffectiverights

elementId: TBD
name: printereffectiverights
dataType: list
structure: list (printerName, trusteeSid,
      standardDelete, standardReadControl, standardWriteDac,
      standardWriteOwner, standardSynchronize,
      accessSystemSecurity, genericRead, genericWrite,
      genericExecute, genericAll, printerAccessAdminister,
      printerAccessUse, jobAccessAdminister, jobAccessRead)
status: current
description: Stores the effective rights of a printer that a discretionary access control list (DACL) structure grants to a specified trustee. The trustee’s effective rights are determined checking all access-allowed and access-denied access control entries (ACEs) in the DACL.

7.348. printerName

elementId: TBD
name: printerName
dataType: string
status: current
description: Specifies the name of the
      printer.

7.349. printerAccessAdminister

elementId: TBD
name: printerAccessAdminister
dataType: boolean
status: current
description:

7.350. printerAccessUse

elementId: TBD
name: printerAccessUse
dataType: boolean
status: current
description:

7.351. jobAccessAdminister

elementId: TBD
name: jobAccessAdminister
dataType: boolean
status: current
description:

7.352. jobAccessRead

elementId: TBD
name: jobAccessRead
dataType: boolean
status: current
description:

7.353. registry

elementId: TBD
name: registry
dataType: list
structure: list (registryHive, registryKey, registryKeyName,
lastWriteTime, registryKeyType, registryKeyValue,
windowsView)
status: current
description: Specifies information that can be
collected about a particular registry key.

7.354. registryHive

```
   elementId: TBD
   name: registryHive
   dataType: enumeration
   structure: HKEY_CLASSES_ROOT ; 0x1 ; This registry subtree
         contains information that associates file types with programs
         and configuration data for automation (e.g. COM
         objects and Visual Basic Programs).
         HKEY_CURRENT_CONFIG ; 0x2 ; This registry subtree contains
         configuration data for the current hardware profile.
         HKEY_CURRENT_USER ; 0x3 ; This registry subtree contains the
         user profile of the user that is currently logged into the
         system.
         HKEY_LOCAL_MACHINE ; 0x4 ; This registry subtree contains
         information about the local system.
         HKEY_USERS ; 0x5 ; This registry subtree contains user-specific
         data.
         ; 0x6 ; The empty string value is permitted here to allow
         for detailed error reporting.
   status: current
   description: The
         hive that the registry key belongs to.
```

7.355. registryKey

```
   elementId: TBD
   name: registryKey
   dataType: string
   status: current
   description: Describes the registry key.
         Note that the hive portion of the string should not be
         included, as this data can be found under the hive
         element.
```

7.356. registryKeyName

```
   elementId: TBD
   name: registryKeyName
   dataType: string
   status: current
   description: Describes the name of a
         registry key.
```

7.357. lastWriteTime

elementId: TBD
name: lastWriteTime
dataType: unsigned64
status: current
description: The last time that the key or any of its value entries were modified. The value of this entity represents the FILETIME structure which is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). Last write time can be queried on any key, with hives being classified as a type of key. When collecting only information about a registry hive or key the last write time will be the time the key or any of its entries were modified. When collecting only information about a registry name the last write time will be the time the containing key was modified. Thus when collecting information about a registry name, the last write time does not correlate directly to the specified name. See the RegQueryInfoKey function lpftLastWriteTime.

7.358. registryKeyType

elementId: TBD
name: registryKeyType
dataType: enumeration
structure: reg_binary ; 0x1 ; The reg_binary type is used by registry keys that specify binary data in any form.
reg_dword ; 0x2 ; The reg_dword type is used by registry keys that specify an unsigned 32-bit integer.
reg_dword_little_endian ; 0x3 ; The reg_dword_little_endian type is used by registry keys that specify an unsigned 32-bit little-endian integer. It is designed to run on little-endian computer architectures.
reg_dword_big_endian ; 0x4 ; The reg_dword_big_endian type is used by registry keys that specify an unsigned 32-bit big-endian integer. It is designed to run on big-endian computer architectures.
reg_expand_sz ; 0x5 ; The reg_expand_sz type is used by registry keys to specify a null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%").
reg_link ; 0x6 ; The reg_link type is used by the registry keys for null-terminated unicode strings. It is related to target path of a symbolic link created by the RegCreateKeyEx function.
reg_multi_sz ; 0x7 ; The reg_multi_sz type is used by registry keys that specify an array of null-terminated strings, terminated by two null characters.

```
        reg_none; 0x8 ;
        The reg_none type is used by registry keys that have no
        defined value type.
        reg_qword; 0x9 ; The reg_qword type is used by registry keys
        that specify an unsigned 64-bit integer.
        reg_qword_little_endian; 0xA ; The reg_qword_little_endian
        type is used by registry keys that specify an unsigned
        64-bit integer in little-endian computer architectures.
        reg_sz; 0xB ; The reg_sz type is used by registry keys that
        specify a single null-terminated string.
        reg_resource_list; 0xC ; The reg_resource_list type is used
        by registry keys that specify a resource list.
        reg_full_resource_descriptor; 0xD ; The
        reg_full_resource_descriptor type is used by registry
        keys that specify a full resource descriptor.
        reg_resource_requirements_list; 0xE ; The
        reg_resource_requirements_list type is used by registry keys
        that specify a resource requirements list.
        ; 0xF ; The empty string value is permitted here to allow
        for detailed error reporting.
  status: current
  description:
        Specifies the type of data stored by the registry key.
```

7.359. registryKeyValue

elementId: TBD
name: registryKeyValue
dataType: string
status: current
description: Holds the actual value of the specified registry key. The representation of the value as well as the associated datatype attribute depends on type of data stored in the registry key. If the value being tested is of type REG_BINARY, then the datatype attribute should be set to 'binary' and the data represented by the value entity should follow the xsd:hexBinary form. (each binary octet is encoded as two hex digits) If the value being tested is of type REG_DWORD, REG_QWORD, REG_DWORD_LITTLE_ENDIAN, REG_DWORD_BIG_ENDIAN, or REG_QWORD_LITTLE_ENDIAN then the datatype attribute should be set to 'int' and the value entity should represent the data as an unsigned integer. DWORD and QWORD values represnt unsigned 32-bit and 64-bit integers, respectively. If the value being tested is of type REG_EXPAND_SZ, then the datatype attribute should be set to 'string' and the pre-expanded string should be represented by the value entity. If the value being tested is of type REG_MULTI_SZ, then only a single string (one of the multiple strings) should be tested using the value entity with the datatype attribute set to 'string'. In order to test multiple values, multiple OVAL registry tests should be used. If the specified registry key is of type REG_SZ, then the datatype should be 'string' and the value entity should be a copy of the string. If the value being tested is of type REG_LINK, then the datatype attribute should be set to 'string' and the null-terminated Unicode string should be represented by the value entity.

7.360. regkeyauditedpermissions

elementId: TBD
name: regkeyauditedpermissions
dataType: list
structure: list (registryKey, trusteeSid, trusteeName,
standardDelete, standardReadControl, standardWriteDac,
standardWriteOwner, standardSynchronize,
accessSystemSecurity, genericRead, genericWrite,
genericExecute, genericAll, keyQueryValue, keySetValue,
keyCreateSubKey, keyEnumerateSubKeys, keyNotify,
keyCreateLink, keyWow6464Key, keyWow6432Key, keyWow64Res,
windowsView)
status: current
description: Stores the audited access rights of a registry key that a system access control list (SACL) structure grants to a specified trustee. The trustee's audited access rights are determined checking all access control entries (ACEs) in the SACL.

7.361. auditKeyQueryValue

elementId: TBD
name: auditKeyQueryValue
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.362. auditKeySetValue

elementId: TBD
name: auditKeySetValue
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.363. auditKeyCreateSubKey

elementId: TBD
name: auditKeyCreateSubKey
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.364. auditKeyEnumerateSubKeys

elementId: TBD
name: auditKeyEnumerateSubKeys
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.365. auditKeyNotify

elementId: TBD
name: auditKeyNotify
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.366. auditKeyCreateLink

elementId: TBD
name: auditKeyCreateLink
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.367. auditKeyWow6464Key

elementId: TBD
name: auditKeyWow6464Key
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.368. auditKeyWow6432Key

elementId: TBD
name: auditKeyWow6432Key
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.369. auditKeyWow64Res

elementId: TBD
name: auditKeyWow64Res
dataType: enumeration
structure: AUDIT_FAILURE ; 0x1 ; The audit type AUDIT_FAILURE is used to perform audits on all unsuccessful occurrences of specified events when auditing is enabled.
AUDIT_NONE ; 0x2 ; The audit type AUDIT_NONE is used to cancel all auditing options for the specified events.
AUDIT_SUCCESS ; 0x3 ; The audit type AUDIT_SUCCESS is used to perform audits on all successful occurrences of the specified events when auditing is enabled.
AUDIT_SUCCESS_FAILURE ; 0x4 ; The audit type AUDIT_SUCCESS_FAILURE is used to perform audits on all successful and unsuccessful occurrences of the specified events when auditing is enabled.
; 0x5 ; The empty string value is permitted here to allow for detailed error reporting.
status: current
description:

7.370. regkeyeffectiverights

elementId: TBD
name: regkeyeffectiverights
dataType: list
structure: list (registryHive, registryKey, trusteeSid,
trusteeName, standardDelete, standardReadControl,
standardWriteDac, standardWriteOwner, standardSynchronize,
accessSystemSecurity, genericRead, genericWrite,
genericExecute, genericAll, keyQueryValue, keySetValue,
keyCreateSubKey, keyEnumerateSubKeys, keyNotify,
keyCreateLink, keyWow6464Key, keyWow6432Key, keyWow64Res,
windowsView)
status: current
description: Stores the effective rights of a registry key that a discretionary access control list (DACL) structure grants to a specified trustee. The trustee’s effective rights are determined checking all access-allowed and access-denied access control entries (ACEs) in the DACL.

7.371. keyQueryValue

elementId: TBD
name: keyQueryValue
dataType: boolean
status: current
description: Specifies whether or not permission is granted to query the key’s value.

7.372. keySetValue

elementId: TBD
name: keySetValue
dataType: boolean
status: current
description: Specifies whether or not permission is granted to set the key’s value.

7.373. keyCreateSubKey

elementId: TBD
name: keyCreateSubKey
dataType: boolean
status: current
description: Specifies whether or not permission is granted to create a subkey.

7.374. keyEnumerateSubKeys

elementId: TBD
name: keyEnumerateSubKeys
dataType: boolean
status: current
description: Specifies whether or
not permission is granted to list the subkeys associated
with key.

7.375. keyNotify

elementId: TBD
name: keyNotify
dataType: boolean
status: current
description:

7.376. keyCreateLink

elementId: TBD
name: keyCreateLink
dataType: boolean
status: current
description:

7.377. keyWow6464Key

elementId: TBD
name: keyWow6464Key
dataType: boolean
status: current
description:

7.378. keyWow6432Key

elementId: TBD
name: keyWow6432Key
dataType: boolean
status: current
description:

7.379. keyWow64Res

elementId: TBD
name: keyWow64Res
dataType: boolean
status: current
description:

7.380. service

elementId: TBD
name: service
dataType: list
structure: list (serviceName, displayName, description,
      serviceType, startType, currentState, controlsAccepted,
      startName, path, pid, serviceFlag, dependencies)
status: current
description: Stores information about Windows services that are
present on the system.

7.381. displayName

elementId: TBD
name: displayName
dataType: string
status: current
description: Specifies the name of the
      service as specified in administrative tools.

7.382. description

elementId: TBD
name: description
dataType: string
status: current
description: Specifies the description of
      the service.

7.383. serviceType

elementId: TBD
name: serviceType
dataType: enumeration
structure: SERVICE_FILE_SYSTEM_DRIVER ; 0x1 ; The
SERVICE_FILE_SYSTEM_DRIVER type means that the service is
a file system driver. The DWORD value that this
corresponds to is 0x00000002.
SERVICE_KERNEL_DRIVER ; 0x2 ; The SERVICE_KERNEL_DRIVER type
means that the service is a driver. The DWORD value that
this corresponds to is 0x00000001.
SERVICE_WIN32_OWN_PROCESS ; 0x3 ; The SERVICE_WIN32_OWN_PROCESS
type means that the service runs in its own process. The DWORD
value that this corresponds to is 0x00000010.
SERVICE_WIN32_SHARE_PROCESS ; 0x4 ; The
SERVICE_WIN32_SHARE_PROCESS type means that the service runs
in a process with other services. The DWORD value that this
corresponds to is 0x00000020.
SERVICE_INTERACTIVE_PROCESS ; 0x5 ; The
SERVICE_WIN32_SHARE_PROCESS type means that the service runs
in a process with other services. The DWORD value that this
corresponds to is 0x00000100.
; 0x6 ; The empty string value is permitted here to allow for
empty elements associated with error conditions.
status: current
description:
Specifies the type of the service.

7.384. startType

```
   elementId: TBD
   name: startType
   dataType: enumeration
   structure: SERVICE_AUTO_START ; 0x1 ; The SERVICE_AUTO_START type
         means that the service is started automatically by the Service
         Control Manager (SCM) during startup. The DWORD value that
         this corresponds to is 0x00000002.
         SERVICE_BOOT_START ; 0x2 ; The SERVICE_BOOT_START type means
         that the driver service is started by the system loader. The
         DWORD value that this corresponds to is 0x00000000.
         SERVICE_DEMAND_START ; 0x3 ; The SERVICE_DEMAND_START type
         means that the service is started by the Service Control
         Manager (SCM) when StartService() is called. The DWORD value
         that this corresponds to is 0x00000003.
         SERVICE_DISABLED ; 0x4 ; The SERVICE_DISABLED type means
         that the service cannot be started. The DWORD value that
         this corresponds to is 0x00000004.
         SERVICE_SYSTEM_START ; 0x5 ; The SERVICE_SYSTEM_START type
         means that the service is a device driver started by
         IoInitSystem(). The DWORD value that this corresponds to is
         0x00000001.
         ; 0x6 ; The empty string value is permitted here to allow
         for empty elements associated with error conditions.
   status: current
   description: Specifies when the service should be started.
```

7.385. currentState

```
   elementId: TBD
   name: currentState
   dataType: enumeration
   structure: SERVICE_CONTINUE_PENDING ; 0x1 ; The
         SERVICE_CONTINUE_PENDING type means that the service has been
         sent a command to continue, however, the command has
         not yet been executed. The DWORD value that this corresponds
         to is 0x00000005. SERVICE_PAUSE_PENDING ; 0x2 ; The
         SERVICE_PAUSE_PENDING type means that the service has been
         sent a command to pause, however, the command has not
         yet been executed. The DWORD value that this corresponds to
         is 0x00000006.
         SERVICE_PAUSED ; 0x3 ; The SERVICE_PAUSED type means that
         the service is paused. The DWORD value that this corresponds
         to is 0x00000007.
         SERVICE_RUNNING ; 0x4 ; The SERVICE_RUNNING type means that
         the service is running. The DWORD value that this
         corresponds to is 0x00000004.
         SERVICE_START_PENDING ; 0x5 ; The SERVICE_START_PENDING type
         means that the service has been sent a command to start,
         however, the command has not yet been executed. The DWORD
         value that this corresponds to is 0x00000002.
         SERVICE_STOP_PENDING ; 0x6 ; The SERVICE_STOP_PENDING type
         means that the service
         has been sent a command to stop, however, the command has
         not yet been executed. The DWORD value that this
         corresponds to is 0x00000003.
         SERVICE_STOPPED ; 0x7 ; The SERVICE_STOPPED type means that
         the service is stopped. The DWORD value that this corresponds
         to is 0x00000001.
         ; 0x8 ; The empty string value is permitted here to allow
         for empty elements associated with error conditions.
   status: current
   description: Specifies the current state of
         the service.
```

7.386. controlsAccepted

```
   elementId: TBD
   name: controlsAccepted
   dataType: enumeration
   structure:
     SERVICE_ACCEPT_NETBINDCHANGE ; 0x1 ;
     The SERVICE_ACCEPT_NETBINDCHANGE type means that the
     service is a network component and can accept changes in its
     binding without being stopped or restarted. The DWORD value
     that this corresponds to is 0x00000010.
     SERVICE_ACCEPT_PARAMCHANGE ; 0x2 ; The SERVICE_ACCEPT_PARAMCHANGE
```

```
     type means that the service can re-read its
     startup parameters without being stopped or restarted. The
     DWORD value that this corresponds to is 0x00000008.
     SERVICE_ACCEPT_PAUSE_CONTINUE ; 0x3 ; The
     SERVICE_ACCEPT_PAUSE_CONTINUE type means that the service
     can be paused or continued. The DWORD value that this
     corresponds to is 0x00000002.
     SERVICE_ACCEPT_PRESHUTDOWN ; 0x4 ; The
     SERVICE_ACCEPT_PRESHUTDOWN type means that the service can
     receive pre-shutdown notifications. The DWORD value
     that this corresponds to is 0x00000100.
     SERVICE_ACCEPT_SHUTDOWN ; 0x5 ; The SERVICE_ACCEPT_SHUTDOWN
     type means that the service can receive shutdown notifications.
     The DWORD value that this corresponds to is 0x00000004.
     SERVICE_ACCEPT_STOP ; 0x6 ; The SERVICE_ACCEPT_STOP type
     means that the service can be stopped. The DWORD value
     that this corresponds to is 0x00000001.
     SERVICE_ACCEPT_HARDWAREPROFILECHANGE ; 0x7 ; The
     SERVICE_ACCEPT_HARDWAREPROFILECHANGE type means that the
     service can receive notifications when the system's
     hardware profile changes. The DWORD value that this
     corresponds to is 0x00000020.
     SERVICE_ACCEPT_POWEREVENT ; 0x8 ; The SERVICE_ACCEPT_POWEREVENT
     type means that the service can receive notifications when the
     system's power status has changed. The DWORD value that this
     corresponds to is 0x00000040.
     SERVICE_ACCEPT_SESSIONCHANGE ; 0x9 ; The
     SERVICE_ACCEPT_SESSIONCHANGE type means that the service can
     receive notifications when the system's session
     status has changed. The DWORD value that this corresponds
     to is 0x00000080.
     SERVICE_ACCEPT_TIMECHANGE ; 0xA ; The SERVICE_ACCEPT_TIMECHANGE
     type means that the service can receive notifications when
     the system time changes. The DWORD value that this corresponds
     to is 0x00000200.
     SERVICE_ACCEPT_TRIGGEREVENT ; 0xB ; The
     SERVICE_ACCEPT_TRIGGEREVENT type means that the service can
     receive notifications when an event that the service
     has registered for occurs on the system. The DWORD value that
     this corresponds to is 0x00000400.
     ; 0xC ; The empty string value is permitted here to allow
     for empty elements associated with error conditions.
   status: current

   description: Specifies the control codes that a service will
                accept and process.
```

7.387. startName

elementId: TBD
name: startName
dataType: string
status: current
description: Specifies the account under which the process should run.

7.388. serviceFlag

elementId: TBD
name: serviceFlag
dataType: boolean
status: current
description: Specifies whether the service is in a system process that must always run (true) or if the service is in a non-system process or is not running (false).

7.389. dependencies

elementId: TBD
name: dependencies
dataType: string
status: current
description: Specifies the dependencies of this service on other services.

7.390. serviceeffectiverights

elementId: TBD
name: serviceeffectiverights
dataType: list
structure: list (serviceName, trusteeSid, standardDelete, standardReadControl, standardWriteDac, standardWriteOwner, genericRead, genericWrite, genericExecute, serviceQueryConf, serviceChangeConf, serviceQueryStat, serviceEnumDependents, serviceStart, serviceStop, servicePause, serviceInterrogate, serviceUserDefined)
status: current
description: Stores the effective rights of a service that a discretionary access control list (DACL) structure grants to a specified trustee. The trustee’s effective rights are determined by checking all access-allowed and access-denied access control entries (ACEs) in the DACL.

7.391. trusteeSid

elementId: TBD
name: trusteeSid
dataType: string
status: current
description: Specifies the SID that is associated with a user, group, system, or program (such as a Windows service).

7.392. serviceQueryConf

elementId: TBD
name: serviceQueryConf
dataType: boolean
status: current
description: Specifies whether or not permission is granted to query the service configuration.

7.393. serviceChangeConf

elementId: TBD
name: serviceChangeConf
dataType: boolean
status: current
description: Specifies whether or not permission is granted to change service configuration.

7.394. serviceQueryStat

elementId: TBD
name: serviceQueryStat
dataType: boolean
status: current
description: Specifies whether or not permission is granted to query the service control manager about the status of the service.

7.395. serviceEnumDependents

elementId: TBD
name: serviceEnumDependents
dataType: boolean
status: current
description: Specifies whether or not permission is granted to query for an enumeration of all the services dependent on the service.

7.396. serviceStart

elementId: TBD
name: serviceStart
dataType: boolean
status: current
description: Specifies whether or not permission is granted to start the service.

7.397. serviceStop

elementId: TBD
name: serviceStop
dataType: boolean
status: current
description: Specifies whether or not permission is granted to stop the service.

7.398. servicePause

elementId: TBD
name: servicePause
dataType: boolean
status: current
description: Specifies whether or not permission is granted to pause or continue the service.

7.399. serviceInterrogate

elementId: TBD
name: serviceInterrogate
dataType: boolean
status: current
description: Specifies whether or not permission is granted to request the service to report its status immediately.

7.400. serviceUserDefined

elementId: TBD
name: serviceUserDefined
dataType: boolean
status: current
description: Specifies whether or not permission is granted to specify a user-defined control code.

7.401. sharedresourceauditedpermissions

```
elementId: TBD
name: sharedresourceauditedpermissions
dataType: list
structure: list (netname, trusteeSid,
      standardDelete, standardReadControl, standardWriteDac,
      standardWriteOwner, standardSynchronize,
      accessSystemSecurity, genericRead, genericWrite,
      genericExecute, genericAll)
status: current
description: Stores
      the audited access rights of a shared resource that a system
      access control list (SACL) structure grants to a
      specified trustee. The trustee’s audited access rights are
      determined checking all access control entries (ACEs)
      in the SACL.
```

7.402. netname

```
elementId: TBD
name: netname
dataType: string
status: current
description: Specifies the name associated
      with a particular shared resource.
```

7.403. sharedresourceeffectiverights

```
elementId: TBD
name: sharedresourceeffectiverights
dataType: list
structure: list (netname, trusteeSid,
      standardDelete, standardReadControl, standardWriteDac,
      standardWriteOwner, standardSynchronize,
      accessSystemSecurity, genericRead, genericWrite,
      genericExecute, genericAll)
status: current
description: Stores
      the effective rights of a shared resource that a
      discretionary access control list (DACL) structure grants
      to a specified trustee. The trustee’s effective rights are
      determined checking all access-allowed and access-denied
      access control entries (ACEs) in the DACL.
```

7.404. user

elementId: TBD
name: user
dataType: list
structure: list (username, enabled, group, lastLogon)
status: current
description: Specifies the groups to which a user belongs.

7.405. enabled

elementId: TBD
name: enabled
dataType: boolean
status: current
description: Represents whether the particular user is enabled or not.

7.406. lastLogon

elementId: TBD
name: lastLogon
dataType: unsigned32
status: current
description: The date and time when the last logon occurred.

7.407. groupSid

elementId: TBD
name: groupSid
dataType: string
status: current
description: Represents the SID of a particular group. If the specified user belongs to more than one group, then multiple groupSid elements are applicable. If the specified user is not a member of a single group, then a single groupSid element should be incldued with a status of 'does not exist'. If there is an error determining the groups that the user belongs to, then a single groupSid element should be included with a status of 'error'.

7.408. endpointType

```
elementId: TBD
name: endpointType
dataType: enumeration
status: current
description: The possible types of endpoint in the
             enterprise.
structure:
workstation; 0x1; Workstation Endpoint
printer;     0x2; Printer Endpoint
router;      0x3; Router Endpoint
tablet;      0x4; Tablet Endpoint
```

7.409. endpointPurpose

```
elementId: TBD
name: endpointPurpose
dataType: string
status: current
description: A description of how the endpoint is
             used within the enterprise.
             Examples include end user system,
             and public web server.
```

7.410. endpointCriticality

```
elementId: TBD
name: endpointCriticality
dataType: string
status: current
description: An enterprise-defined rating which
             indicates the criticality of the
             endpoint. The rating should be
             specific enough to assess the impact
             to the overall enterprise if the
             endpoint is attacked or lost.
```

7.411. ingestTimestamp

```
elementId: TBD
name: ingestTimestamp
dataType: dateTimeSeconds
status: current
description: The point in time that the
             description of a vulnerability was
             received by the enterprise.
```

7.412. vulnerabilityVersion

elementId: TBD
name: vulnerabilityVersion
dataType: string
status: current
description: The version or iteration of the vulnerability description information (reported by the author, if applicable).

7.413. vulnerabilityExternalId

elementId: TBD
name: vulnerabilityExternalId
dataType: string
status: current
description: An external or third-party ID assigned to the vulnerability description. This could be multiple IDs in some cases (e.g., vendor bug ID, global ID, discoverer’s local ID, third-party vulnerability database ID, etc.).

7.414. vulnerabilitySeverity

elementId: TBD
name: vulnerabilitySeverity
dataType: string
status: current
description: The severity of the vulnerability (reported by the author, if applicable).

7.415. assessmentTimestamp

elementId: TBD
name: assessmentTimestamp
dataType: dateTimeSeconds
status: current
description: The point in time that the assessment was performed against an endpoint.

7.416. vulnerableSoftware

```
elementId: TBD
name: vulnerableSoftware
dataType: list
status: current
description: A listing of software products
             installed on the endpoint which are
             known to have vulnerabilities.
structure: list(softwareInstance*)
```

7.417. endpointVulnerabilityStatus

```
elementId: TBD
name: endpointVulnerabilityStatus
dataType: enumeration
status: current
description: Overall vulnerability status of an
             enterprise endpoint.
structure: Pass; 0x1; Endpoint passed the
                      vulnerability test(s).
           Fail; 0x2; Endpoint failed the
                      vulnerability test(s).
```

7.418. vulnerabilityDescription

```
elementId: TBD
name: vulnerabilityDescription
dataType: string
status: current
description: A human-readable description of the
             vulnerability.
```

8. 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.

9. IANA Considerations

This document specifies an initial set of Information Elements for SACM in Section 7. An Internet Assigned Numbers Authority (IANA) registry will be created and populated with the Information Elements in Section 7. New assignments for SACM Information Elements will be administered by IANA through Expert Review [RFC2434]. The designated experts MUST check the requested Information Elements for completeness and accuracy of the submission with respect to the template and requirements expressed in Section 4 and Section 4.1. Requests for Information Elements that duplicate the functionality of existing Information Elements SHOULD be declined. The smallest available Information Element identifier SHOULD be assigned to a new Information Element. The definition of new Information Elements MUST be published using a well-established and persistent publication medium.

10. 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. This information model only contains an initial listing of items that need to be considered with respect to security and will need to be augmented as the model continues to be developed.

Security considerations include:

Authentication: Every SACM Component and asset needs to be able to identify itself and verify the identity of other SACM Components and assets.

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

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

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

Considerations with respect to the operational aspects of collection, evaluation, and storage security automation information can be found in Section 11.

Considerations concerning the privacy of security automation information can be found in Section 12.

11. Operational Considerations

The following sections outline a series of operational considerations for SACM deployments within an organization. This section may be expanded to include other considerations as the WG gains additional operational experience with SACM deployments and extending the information model.

11.1. Endpoint Designation

In order to successfully carry out endpoint posture assessment, it is necessary to be able to identify the endpoints on a network and track the changes to them over time. Specifically, enabling SACM Components to:

- Tell whether two endpoint attribute assertions concern the same endpoint

- Respond to compliance measurements, for example by reporting, remediating, and quarantining (SACM does not specify these responses, but SACM exists to enable them).

Ideally, every endpoint would be identified by a unique identifier present on the endpoint, but, this is complicated due to different factors such as the variety of endpoints on a network, the ability of tools to reliably access such an identifer, and the ability of tools to correlate disparate identifiers. As a result, it is necessary for an endpoint to be identified by a set of attributes that uniquely identify it on a network. The set of attributes that uniquely identify an endpoint on a network will likely vary by organization; however, there are a number of properties to consider when selecting identifying attributes as some are better suited for identification purposes than others.

Multiplicity: Is the attribute typically associated with a single endpoint or with multiple endpoints? If the attribute is associated with a single endpoint, it is better for identifying an endpoint on a network.

Persistence: How likely is the attribute to change? Does it never change? Does it only change when the endpoint is reprovisioned? Does it only change due to an event? Does it change on an ad-hoc and often unpredictable basis? Does it constantly change? The less likely it is for an attribute to change over time, the better it is for identifying an endpoint on a network.

Immutability: How difficult is it to change the attribute? Is the attribute hardware rooted and never changes? Can the attribute be changed by a user/process with the appropriate access? Can the attribute be changed without controlled access. The less likely an attribute is to change over time, the better chance it will be usable to identify an endpoint over time.

Verifiable: Can the attribute be corroborated? Can the attribute be externally verified with source authentication? Can the attribute be externally verified without source authentication? Is it impossible to externally verify the attribute. Attributes that can be externally verified are more likely to be accurate and are better for identifying endpoints on a network.

With that said, requiring SACM Components and end users to constantly refer to a set of attributes to identify an endpoint, is particularly burdensome. As a result, SACM supports the concept of a target endpoint label which associates an identifier (unique to a SACM domain) with the set of attributes used by an organization to identify endpoints on a network. Once defined for an endpoint, the target endpoint label can be used in place of the set of identifying attributes.

11.2. Timestamp Accuracy

An organization will likely have different collectors deployed across the network that will be configured to collect posture attributes on varying frequencies (periodic, ad-hoc, event-driven, on endpoint, off endpoint, etc.). Some collectors will detect changes as soon as they occur whereas others will detect them at a later point during a periodic scan or when an event has triggered the collection of posture attributes. Furthermore, some changes will be detected on the endpoint and others will be observed off of the endpoint. As a result of these differences, the accuracy of the timestamp associated with the collected information will vary. For example, if a collector is only running once every 12 hours, the change probably happened at some point in time prior to the scan and the timestamp is likely not accurate. Due to this, it is important for system administrators to determine if the accuracy of a timestamp is good enough for their intended purposes.

12. Privacy Considerations

In the IETF, there are privacy concerns with respect to endpoint identity and monitoring. This is especially true when the activity on an endpoint can be linked to a particular person. For example, by correlating endpoint attributes such as usernames, certificates, etc. with browser activity, it may be possible to gain insight in to user behavior and trends beyond what is required to carry out endpoint posture assessments. In the hands of the wrong person, this information could be used to negatively influence a user’s behavior or to plan attacks against the organization’s infrastructure.

As a result, SACM data models should incorporate a mechanism by which an organization can designate which endpoint attributes are considered sensitive with respect to privacy. This will allow SACM Components to handle endpoint attributes in a manner consistent with the organization’s privacy policies. Furthermore, organization’s should put the proper mechanism in place to ensure endpoint attributes are protected when transmitted, stored, and accessed to ensure only authorized parties are granted access.

It should also be noted that some of this is often mitigated by organizational policies that require a user of an organization’s network to consent to some level of monitoring in return for access to the network and other resources. The information that is monitored and collected will vary by organization and further highlights the need for a mechanism by which an organization can specify what constitutes privacy sensitive information for them.

13. References

13.1. Normative References

[PEN] Internet Assigned Numbers Authority, "Private Enterprise Numbers", July 2016, <https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>.

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

13.2. Informative References

[I-D.ietf-sacm-requirements]
Cam-Winget, N. and L. Lorenzin, "Secure Automation and Continuous Monitoring (SACM) Requirements", draft-ietf-sacm-requirements-01 (work in progress), October 2014.

[I-D.ietf-sacm-terminology]
Waltermire, D., Montville, A., Harrington, D., and N. Cam-Winget, "Terminology for Security Assessment", draft-ietf-sacm-terminology-05 (work in progress), August 2014.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, DOI 10.17487/RFC2434, October 1998, <http://www.rfc-editor.org/info/rfc2434>.

[RFC3580] Congdon, P., Aboba, B., Smith, A., Zorn, G., and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines", RFC 3580, DOI 10.17487/RFC3580, September 2003, <http://www.rfc-editor.org/info/rfc3580>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <http://www.rfc-editor.org/info/rfc4949>.

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <http://www.rfc-editor.org/info/rfc5209>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <http://www.rfc-editor.org/info/rfc7012>.

[RFC7632] Waltermire, D. and D. Harrington, "Endpoint Security Posture Assessment: Enterprise Use Cases", RFC 7632, DOI 10.17487/RFC7632, September 2015, <http://www.rfc-editor.org/info/rfc7632>.

Appendix A. Change Log

A.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 arrow. 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.

A.2. Changes in Revision 02

Added Section Identifying Attributes.

Split the figure into Figure Model of Endpoint and Figure Information Elements.

Added Figure Information Elements Take 2, proposing a triple-store model.

Some editorial cleanup

A.3. Changes in Revision 03

Moved Appendix A.1, Appendix A.2, and Mapping to SACM Use Cases into the Appendix. Added a reference to it in Section 1

Added the Section 4 section. Provided notes for the type of information we need to add in this section.

Added the Section 6 section. Moved sections on Endpoint, Hardware Component, Software Component, Hardware Instance, and Software Instance there. Provided notes for the type of information we need to add in this section.

Removed the Provenance of Information Section. SACM is not going to solve provenance rather give organizations enough information to figure it out.

Updated references to the Endpoint Security Posture Assessment: Enterprise Use Cases document to reflect that it was published as an RFC.

Fixed the formatting of a few figures.

Included references to [RFC3580] where RADIUS is mentioned.

A.4. Changes in Revision 04

Integrated the IPFIX [RFC7012] syntax into Section 4.

Converted many of the existing SACM Information Elements to the IPFIX syntax.

Included existing IPFIX Information Elements and datatypes that could likely be reused for SACM in Section 7 and Section 4 respectively.

Removed the sections related to reports as described in https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/30.

Cleaned up other text throughout the document.

A.5. Changes in Revision 05

Merged proposed changes from the I-D IM into the WG IM (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/41).

Fixed some formatting warnings.

Removed a duplicate IE and added a few IE datatypes that were missing.

A.6. Changes in Revision 06

Clarified that the SACM statement and content-element subjects are conceptual and that they do not need to be explicitly defined in a data model as long as the necessary information is provided.

Updated the IPFIX syntax used to define Information Elements. There are still a couple of open issues that need to be resolved.

Updated some of the Information Elements contained in Section 7 to use the revised IPFIX syntax. The rest of the Information Elements will be converted in a later revision.

Performed various clean-up and refactoring in Sections 6 and 7. Still need to go through Section 8.

Removed appendices that were not referenced in the body of the draft. The text from them is still available in previous revisions of this document if needed.

A.7. Changes in Revision 07

Made various changes to the IPFIX syntax based on discussions at the IETF 96 Meeting. Changes included the addition of a structure property to the IE specification template, the creation of an enumeration datatype, and the specification of an IE naming convention.

Provided text to define Collection Guidance, Evaluation Guidance, Classification Guidance, Storage Guidance, and Evaluation Results.

Included additional IEs related to software, configuration, and the vulnerability assessment scenario.

Added text for the IANA considerations, security considerations, operational considerations, and privacy considerations sections.

Performed various other editorial changes and clean-up.

A.8. Changes in Revision 08

Clarified text that describes subjects and attributes.

Clarified text that describes SACM Statements and Content Elements.

Removed stray metadata property fields from the definitions of several IEs.

Specified a syntax for defining category IEs.

Added an anyCategory IE that represents any IE in the IM.

Fixed several errors reported by the Travis-CI continuous integration service.

Performed various other editorial changes and clean-up.

A.9. Changes in Revision 09

Added "derived", "authority", and "verified" to the collectionTaskType IE (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/18).

Updated IE examples that use content-type to use statement-type (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/56).

Added "networkZoneLocation", "layer2NetworkLocation", and "layer3NetworkLocation" IEs (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/9).

Created a softwareClass attribute IE and added it to the softwareInstance subject IE. Also, removed the os* attribute IEs (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/10).

A.10. Changes in Revision 10

Added several IEs necessary for the SACM Vulnerability Assessment Scenario (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/43).

Fixed various typos and formatting issues.

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

Lisa Lorenzin
Pulse Secure, LLC
2700 Zanker Road, Suite 200
San Jose, CA 95134
USA

Email: llorenzin@pulsesecure.net

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

Email: msc@mitre.org

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

Email: dhaynes@mitre.org

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

SACM N. Cam-Winget
Internet-Draft Cisco Systems
Intended status: Informational L. Lorenzin
Expires: February 2, 2018 Pulse Secure
August 1, 2017

Security Automation and Continuous Monitoring (SACM) Requirements
draft-ietf-sacm-requirements-18

Abstract

This document defines the scope and set of requirements for the Secure Automation and Continuous Monitoring (SACM) architecture, data model and transfer protocols. The requirements and scope are based on the agreed upon use cases ([RFC7632]).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
  1.1. Requirements Language 3
2. Requirements 3
  2.1. Requirements for SACM 4
  2.2. Requirements for the Architecture 7
  2.3. Requirements for the Information Model 9
  2.4. Requirements for the Data Model 10
  2.5. Requirements for Data Model Operations 12
  2.6. Requirements for SACM Transfer Protocols 14
3. Acknowledgments 15
4. IANA Considerations 15
5. Security Considerations 15
  5.1. Trust between Provider and Requestor 16
  5.2. Privacy Considerations 17
6. References 18
  6.1. Normative References 18
  6.2. Informative References 18
Authors' Addresses 19

1. Introduction

Today's environment of rapidly-evolving security threats highlights the need to automate the sharing of security information (such as posture information) while protecting user information and the systems that store, process, and transmit this information. Security threats can be detected in a number of ways. The Secure Automation and Continuous Monitoring (SACM) 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 the enterprise, in particular, for the use cases as exemplified in [RFC7632]. As proposals are evaluated for SACM standardization, their drafts are expected to include a section that describe how they address each of the enumerated requirements.

Also, this document uses terminology defined in [I-D.ietf-sacm-terminology].

1.1. Requirements Language

Use of each capitalized word within a sentence or phrase carries the following meaning during the SACM WG's protocol selection process:

MUST - indicates an absolute requirement

MUST NOT - indicates something absolutely prohibited

SHOULD - indicates a strong recommendation of a desired result

SHOULD NOT - indicates a strong recommendation against a result

MAY - indicates a willingness to allow an optional outcome

When the words appear in lower case, their natural language meaning is used.

2. Requirements

This document defines requirements based on the SACM use cases described in [RFC7632]. This section describes the requirements used by SACM to assess and compare candidate data models, interfaces, and protocols. These requirements express characteristics or features that a candidate protocol, information model, or data model must be capable of offering to ensure security and interoperability.

Multiple data models, protocols, and transfers may be employed in a SACM environment. A SACM transfer protocol is one that runs on top of transport layer protocols such as TCP/IP or internet layer protocols such as HTTP, carries operations (requests / responses), and moves data.

SACM will define an architecture and information model focused on addressing the needs for determining, sharing, and using posture information via Posture Information Providers and Posture Information Consumers securely. With the information model defining assets and attributes to facilitate the guidance, collection, and assessment of posture, tasks that should be considered include:

1. Asset Classification: Map the target endpoint and/or the assets on the target endpoints to asset classes. This enables identification of the attributes needed to exchange information pertaining to the target endpoint.

2. Attribute Definition: Define the attributes desired to be collected from each target endpoint. For instance, organizations will want to know what software is installed and its critical security attributes such as patch level.

3. Policy Definition: This is where an organization can express its policy for acceptable or problematic values of an endpoint attribute. The expected values of an endpoint attribute are determined for later comparison against the actual endpoint attribute values during the evaluation process. Expected values may include both those values which are good as well as those values which represent problems, such as vulnerabilities. The organization can also specify the endpoint attributes that are to be present for a given target endpoint.

4. Information Collection: Collect information (attribute values) from the target endpoint to populate the endpoint data.

5. Endpoint Assessment: Evaluate the actual values of the endpoint attributes against those expressed in the policy. (An evaluation result may become additional endpoint data).

6. Result Reporting: Report the results of the evaluation for use by other components. Examples of use of a report would be additional evaluation, network enforcement, vulnerability detection, and license management.

2.1. Requirements for SACM

Many deployment scenarios can be instantiated to address the above tasks and use cases defined in [RFC7632]. 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 information model, data models, protocols, and transfers defined by SACM MUST be designed to allow support for future (SACM) extensions. SACM MUST allow for both standardized and proprietary extensions.

1. The information model and programmatic interfaces (see G-012 for one example) MUST support the ability to add new operations while maintaining backwards compatibility. SACM-defined transfer protocols MUST have extensibility to allow them to transfer operations that are defined in the future.

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

3. The information model MUST accommodate the interoperable addition of new data types and/or schemas.

G-002 Interoperability: The data models, protocols, and transpors MUST be specified with enough details to ensure interoperability.

G-003 Scalability: SACM needs to support a broad set of deployment scenarios. The data models, protocols, and transports has to be scalable unless they are specifically defined to apply to a special-purpose scenario, such as constrained devices. A SACM transfer protocol standard SHOULD include a section on scalability considerations that addresses the number of endpoints and amount of information to which it can reasonably be expected to scale. Scalability must be addressed to support:

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

* Large number of messages per second: A deployment may involve many rapid or simultaneous events that require processing, generating many messages per second.

* Large number of providers and consumers: A deployment may consist of a very large number of endpoints requesting and/or producing posture assessment information.

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

G-004 Versatility: The data model, protocols, and transports must be suitably specified to enable implementations to fit into different deployment models and scenarios, including considerations for implementations of data models and transports operating in constrained environments. Separate solutions may be necessary to meet the needs of specific deployment models and scenarios.

G-005 Information Extensibility: Non-standard (implementation-specific) attributes MUST be supported. A method SHOULD be defined for preventing collisions from occurring in the naming of all attributes independent of their source. For interoperability and scope boundary, the information model MUST define the mandatory set of attributes.

G-006 Data Protection: To protect the information being shared, SACM components MUST protect the integrity and confidentiality of data in transit (end to end) and data at rest (as information is stored in repositories). Mechanisms for this protection are unspecified but should include industry best practices. These mechanisms are required to be available (i.e. all data-handling components must support them), but are not required to be used in all cases.

G-007 Data Partitioning: A method for partitioning data MUST be supported to accommodate considerations such as geographic, regulatory, operational requirements, overlay boundaries, and federation (where the data may be collected in multiple locations and either centralized or kept in the local region). Where replication of data is supported, it is required that methods exist to prevent update loops.

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

G-009 Information Discovery: There MUST be mechanisms 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), where to go to get a specific piece of that information (i.e. which provider has the information), and what schemas are in use for organizing the 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 Target Endpoint Discovery: SACM MUST define the means by which target endpoints may be discovered. Use Case 2.1.2 describes the need to discover endpoints and their composition.

G-011 Push and Pull Access: Three methods of data access MUST be supported: a Pull model, a solicited Push model, and an unsolicited Push models. All of the methods of data access MUST support the ability for the initiator to filter the set of posture assessment information to be delivered. Additionally, the provider of the information MUST be able to filter the set of posture assessment information based on the permissions of the recipient. This requirement is driven by use cases 2.1.3, 2.1.4 and 2.1.5.

G-012 SACM Component Interface: The interfaces by which SACM components communicate to share endpoint posture information MUST be well defined. That is, the interface defines the data model, SACM transfer protocols, and network transfer protocols to enable SACM components to communicate.

G-013 Endpoint Location and Network Topology: The SACM architecture and interfaces MUST allow for the target endpoint (network) location and network topology to be modeled and understood. Where appropriate, the data model and the interfaces SHOULD allow for discovery of the target 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.

G-015 Data Access Control: Methods of access control must be supported to accommodate considerations such as geographic, regulatory, operational and federations. Entities accessing or publishing data MUST identify themselves and pass access policy.

2.2. Requirements for the Architecture

Following are the requirements for the SACM architecture:

ARCH-001 Component functions: At the simplest abstraction, the SACM architecture MUST represent the core components and interfaces needed to perform the production and consumption of posture assessment information.

ARCH-002 Scalability: The architectural components MUST account for a range of deployments, from very small sets of endpoints to very large deployments.

ARCH-003 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-004 Separation of Data and Management Functions: SACM MUST define both the configuration and management of the SACM data models and protocols used to transfer and share posture assessment information.

ARCH-005 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. The fact that a centralized or decentralized deployment is used SHOULD be invisible to a consumer. However, there may be cases where the producer chooses to include that information due to consumer preference

ARCH-006 Capability Negotiation: Announcement and negotiation of functional capabilities (such as authentication protocols, authorization schemes, data models, transfer protocols, etc.) MUST be supported, enabling a SACM component to make inquiries about the capabilities of other components in the SACM ecosystem.

ARCH-007 Role-based Authorization: The SACM architecture MUST be capable of effecting role-based authorization. Distinction of endpoints capable of and authorized to provide or consume information is required to address appropriate access controls.

ARCH-008 Context-based Authorization: The SACM architecture MUST be capable of effecting context-based authorization. Different policies (e.g. business, regulatory, etc.) might specify what data may be exposed to, or shared by, consumers based on one or more attributes of the consumer. The policy might specify that consumers are required to share specific information either back to the system or to administrators.

ARCH-009 Time Synchronization: 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. The SACM architecture MUST provide a mechanism for all components to synchronize time. A mechanism for detecting and reporting time discrepancies SHOULD be provided by the architecture and reflected in the information model.

2.3. Requirements for the Information Model

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

IM-001 Extensible Attribute Vocabulary: The information model MUST define a minimum set of attributes for communicating Posture Information, to ensure interoperability between data models. (Individual data models may define attributes beyond the mandatory-to-implement minimum set.) The attributes should be defined with a clear mechanism for extensibility to enable data models to adhere to SACM's required attributes as well as allow for their own extensions. The attribute vocabulary should be defined with a clear mechanism for extensibility to enable future versions of the information model to be interoperably expanded with new attributes.

IM-002 Posture Data Publication: The information model MUST allow for the data to be provided by a SACM component either solicited or unsolicited. No aspect of the information model should be dependent upon or assume a push or pull model of publication.

IM-003 Data Model Negotiation: SACM's information model MUST allow support for different data models, data model versions, and different versions of the operations on the data models and transfer protocols. The SACM information model MUST include the ability to discover and negotiate the use of a particular data model or any data model.

IM-004 Data Model Identification: The information model MUST provide a means to uniquely identify each data model. The identifier MUST contain both an identifier of the data model and a version indicator for the data model. The identifiers SHOULD be decomposable so that a customer can query for any version of a specific data model and compare returned values for older or newer than a desired version.

IM-005 Data Lifetime Management: The information model MUST provide a means to allow data models to include data lifetime management. The information model must identify attributes that can allow data models to, at minimum, identify the data's origination time and expected time of next update or data longevity (how long should the data be assumed to still be valid).

IM-006 Singularity and Modularity: The SACM information model MUST be singular (i.e. there is only one information model, not multiple alternative information models from which to choose) and MAY be modular (a conjunction of several sub-components) for ease of maintenance and extension. For example, endpoint identification could be an independent sub-component of the information model, to simplify updating of endpoint identification attributes.

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 meet the requirements defined in this section.

The specific requirements for candidate data models include:

DM-001 Element Association: A SACM Information Model consists of a set of SACM Information Model elements. A SACM Data Model MUST be derived from the SACM Information Model. A SACM Data Model consists of a set of SACM Data Model elements. In this derivation, a SACM Data Model element MAY map to one or more SACM Information Model elements. In addition, a SACM Data Model MAY include additional Data Model elements that are not associated with any SACM Information Model elements.

DM-002 Data Model Structure: The data model can be structured either as one single module or separated into modules and sub-modules that allow for references between them. The data model structure MAY reflect structure in the information model, but does not need to. For example, the data model might use one module to define endpoints, and that module might reference other modules that describe the various assets associated with the endpoint. Constraints and interfaces might further be defined to resolve or tolerate ambiguity in the references (e.g. same IP address used in two separate networks).

DM-003 Search Flexibility: The search interfaces and actions MUST include the ability to start a search anywhere within a data model structure, and the ability to search based on patterns ("wildcard searches") as well as specific data elements.

DM-004 Full vs. Partial Updates: The data model SHOULD include the ability to allow providers of data to provide the data as a whole, or when updates occur. For example, a consumer can request a full update on initial engagement, then request to receive deltas (updates containing only the changes since the last update) on an ongoing basis as new data is generated.

DM-005 Loose Coupling: The data model SHOULD allow for a loose coupling between the provider and the consumer, such that the consumer can request information without being required to request it from a specific provider, and a provider can publish information without having a specific consumer targeted to receive it.

DM-006 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-007 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-008 Data Origin: The data model MUST include the ability for consumers to identify the data origin (provider that collected the data).

DM-009 Origination Time: The data model SHOULD allow the provider to include the information's origination time.

DM-010 Data Generation: The data model MUST allow the provider to include attributes defining how the data was generated (e.g. self-reported, reported by aggregator, scan result, etc.).

DM-011 Data Source: The data model MUST allow the provider to include attributes identifying the data source (target endpoint from which the data was collected) - e.g. hostname, domain (DNS) name or application name.

DM-012 Data Updates: 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-013 Multiple Collectors: 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 observers.

DM-014 Attribute Extensibility: 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. The data model MUST include all attributes defined in the information model and MAY include additional attributes beyond those found in the information model. Additional attributes MUST be defined in accordance with the extensibility framework provided in the information model (see IM-001).

DM-015 Solicited vs. Unsolicited Updates: The data model MUST enable a provider to publish data either solicited (in response to a request from a consumer) or unsolicited (as new data is generated, without a request required). For example, an external collector can publish data in response to a request by a consumer for information about an endpoint, or can publish data as it observes new information about an endpoint, without any specific consumer request triggering the publication; a compliance-server provider may publish endpoint posture information in response to a request from a consumer (solicited), or it may publish posture information driven by a change in the posture of the endpoint (unsolicited).

DM-016 tTransfer Agnostic: The data model MUST be transfer agnostic, to allow for the data operations to leverage the most appropriate SACM transfer protocol.

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. Operations requirements are distinct from transfer requirements in that operations requirements are requirements on the application performing requests and responses, whereas transfer requirements are requirements on the transfer protocol carrying the requests / responses. The specific requirements for such operations include:

OP-001 Time Synchronization: Request and response operations MUST 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 Abstraction: Collection is the act of a SACM component gathering data from a target endpoint. 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 requestor. This requirement should not be interpreted as preventing a collector from providing attributes which were not part of the original request.

OP-004 Attribute-based Query: A query operation is the act of requesting data from a provider. Query operations SHOULD be based on a set of attributes. Query operations MUST support both a query for specific attributes and a query for all 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 Operation Scalability: The operation resulting from a query operation MUST be able to handle the return and receipt of large amounts of data. Use Cases 2.1.4 and 2.1.5 describe 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 Data Abstraction: 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 its 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.

OP-008 Provider Restriction: Request operations MUST include the ability to restrict the data to be provided by a specific provider or a provider with specific characteristics. Response operations MUST include the ability to identify the provider that supplied the response. For example, a SACM Consumer should be able to request that all of the data come from a specific provider by identity (e.g. Provider A) or from a Provider that is in a specific location (e.g. in the Boston office).

2.6. Requirements for SACM Transfer Protocols

The term SACM transfer protocol is intended to be distinguished from underlying transport and internet layer protocols such as TCP/IP or operating at an equivalent level as the HTTP. The SACM transfer protocol is focused on moving data and performing necessary access control operations; it is agnostic to the data model operations.

The requirements for SACM transfer protocols include:

T-001 Multiple transfer Protocol Support: SACM transfer protocols will vary depending on the deployment model that relies on different transfer layer requirements, different device capabilities, and system configurations dealing with connectivity. For example, where posture attributes may be collected directly from an endpoint using NEA’s model [RFC5209], different transports may be defined to collect them using PT-EAP [RFC7171] or PT-TLS [RFC6876] depending on the deployment scenario.

T-002 Data Integrity: SACM transfer protocols MUST be able to ensure data integrity for data in transit.

T-003 Data Confidentiality: SACM transfer protocols MUST be able to support data confidentiality. SACM transfer protocols MUST ensure data protection for data in transit (e.g. by encryption) to provide confidentiality, integrity, and robustness against protocol-based attacks. Note that while the transfer MUST be able to support data confidentiality, implementations MAY provide a configuration option that enables and disables confidentiality in deployments. Protection for data at rest is not in scope for transfer protocols. Data protection MAY be used for both privacy and non-privacy scenarios.

T-004 Transfer Protection: SACM transfer protocols MUST be capable of supporting mutual authentication and replay protection.

T-005 Transfer Reliability: SACM transfer protocols MUST provide reliable delivery of data. This includes the ability to perform fragmentation and reassembly, and to detect replays. The SACM transfer may take advantage of reliability features in the network transport; however, the network transport may be unreliable (e.g. UDP), in which case the SACM transfer running over the unreliable network transport is responsible for ensuring reliability (i.e. by provisions such as confirmations and re-transmits).

T-006 Transfer Layer Requirements: Each SACM transfer protocol MUST clearly specify the transport layer requirements it needs to operate correctly. Examples of items that may need to be specified include connectivity requirements, replay requirements, data link encryption requirements, and/or channel binding requirements. These requirements are needed in order for deployments to be done correctly.

T-007 Transfer Protocol adoption: SACM SHOULD where reasonably possible, leverage and use existing IETF transfer protocols versus defining new ones.

3. Acknowledgments

The authors would like to thank Barbara Fraser, Jim Bieda, and Adam Montville for reviewing and contributing to this draft. In addition, we recognize valuable comments and suggestions made by Jim Schaad and Chris Inacio.

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 transfer protocols may be defined or reused from already existing standards.

To address security and privacy considerations, the data model, protocols, and transports 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 is based on the subject (the information requestor), the provider (the information responder), the object (the endpoint the information is being requested on), and the attribute (what piece of data is being requested). The method by which this authorization is applied is unspecified.

SACM's charter focuses on the workflow orchestration and 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 requesting consumer, or devices that are involved in the transfer 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 transfer 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 possible for these proxies and brokers to inject and affect man-in-the-middle attacks.

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 transfer 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. SACM specifications should provide the means for verifying data origin and data integrity and at minimum, provide endpoint authentication and transfer 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 are 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:

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

5.2. Privacy Considerations

SACM information may contain sensitive information about the target endpoint as well as revealing identity information of the producer or consumer of such information. Similarly, as part of the SACM discovery mechanism, the advertised capabilities (and roles, e.g. SACM components enabled) by the endpoint may be construed as private information.

In addition to identity and SACM capabilities information disclosure, the use of time stamps (or other attributes that can be used as identifiers) could be further used to determine a target endpoint or user’s behavioral patterns. Such attributes may also be deemed sensitive and may required further protection or obfuscation to meet privacy concerns. That is, there may be applications as well as business and regulatory practices that require that aspects of such information be hidden from any parties that do not need to know it.

Data confidentiality can provide some level of privacy but may fall short where unnecessary data is still transmitted. In those cases, filtering requirements at the data model such as OP-005 must be applied to ensure that such data is not disclosed. [RFC6973] provides guidelines for which SACM protocols and information and data models should follow.

6. References

6.1. Normative References

[RFC7632] Waltermire, D. and D. Harrington, "Endpoint Security Posture Assessment: Enterprise Use Cases", RFC 7632, DOI 10.17487/RFC7632, September 2015, <http://www.rfc-editor.org/info/rfc7632>.

6.2. Informative References

[I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., and N. Cam-Winget, "Security Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-13 (work in progress), July 2017.

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <http://www.rfc-editor.org/info/rfc5209>.

[RFC6876] Sangster, P., Cam-Winget, N., and J. Salowey, "A Posture Transport Protocol over TLS (PT-TLS)", RFC 6876, DOI 10.17487/RFC6876, February 2013, <http://www.rfc-editor.org/info/rfc6876>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7171] Cam-Winget, N. and P. Sangster, "PT-EAP: Posture Transport (PT) Protocol for Extensible Authentication Protocol (EAP) Tunnel Methods", RFC 7171, DOI 10.17487/RFC7171, May 2014, <http://www.rfc-editor.org/info/rfc7171>.

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

SACM Working Group H. Birkholz
Internet-Draft Fraunhofer SIT
Intended status: Informational J. Lu
Expires: June 17, 2019 Oracle Corporation
J. Strassner
Huawei Technologies
N. Cam-Winget
Cisco Systems
A. Montville
CIS
December 14, 2018

Security Automation and Continuous Monitoring (SACM) Terminology
draft-ietf-sacm-terminology-16

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terms and Definitions 2
3. IANA Considerations 21
4. Security Considerations 21
5. Acknowledgements 22
6. Change Log 22
7. Contributors 26
8. References 27
8.1. Normative References 28
8.2. Informative References 28
Appendix A. The Attic 29
Authors' Addresses 29

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. Note that explanatory or informational augmentation to definitions are segregated from the definitions themselves. The definition for the term immediately follows the term on the same line, whereas expositional text is contained in subsequent paragraphs immediately following the definition.

Assertion: Defined by the ITU in [X.1252] as "a statement made by an entity without accompanying evidence of its validity".

In the context of SACM, an assertion is the output of a SACM Component in the form of a SACM Statement (including metadata about the data source and data origin, e.g. timestamps). While the validity of an assertion about Content and Content Metadata cannot be verified without, for example, Integrity Proofing of the

Data Source, an assertion (and therefore a SACM statement, respectively) of the validity of Statement Metadata can by enabled by including corresponding Integrity Evidence created by the Data Origin.

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."

Attribute: Is a data element, as defined in [RFC5209], that is atomic.

In the context of SACM, attributes are "atomic" information elements and an equivalent to attribute-value-pairs. Attributes can be components of Subjects, the basic composite definitions that are defined in the SACM Information Model.

Capability: A set of features that are available from a SACM Component.

See also "capability" in [I-D.ietf-i2nsf-terminology].

In the context of SACM, the extent of a SACM component's ability is enabled by the functions it is composed of. Capabilities are registered at a SACM broker (potentially also at a proxy or a repository component if it includes broker functions) by a SACM component via the SACM component registration task and can be discovered by or negotiated with other SACM components via the corresponding tasks. For example, the capability of a SACM provider may be to provide target endpoint records (declarative guidance about well-known or potential target endpoints), or only a subset of that data.

A capability's description is in itself imperative guidance on what functions are exposed to other SACM components in a SACM domain and how to use them in workflows.

The SACM Vulnerability Assessment Scenario [I-D.ietf-sacm-vuln-scenario] defines the terms Endpoint Management Capabilities, Vulnerability Management Capabilities, and Vulnerability Assessment Capabilities, which illustrate specific sets of SACM capabilities on an enterprise IT department's point of view and therefore compose sets of declarative guidance.

Collection Result: Is a composition of one or more content elements carrying information about a target endpoint, that is produced by a collector when conducting a collection task.

Collection Task: A targeted task that collects attributes and/or corresponding attribute values from target endpoint.

There are four types of frequency collection tasks can be conducted with:

ad-hoc, e.g. triggered by a unsolicited query

conditional, e.g. triggered in accordance with policies included in the compositions of workflows

scheduled, e.g. in regular intervals, such as every minute or weekly

continuously, e.g. a network behavior observation

There are three types of collection methods, each requiring an appropriate set of functions to be included in the SACM component conducting the collection task:

Self-Reporting: A SACM component located on the target endpoint itself conducts the collection task.

Remote-Acquisition: A SACM component located on an Endpoint different from the target endpoint conducts the collection task via interfaces available on the target endpoint, e.g. SNMP/NETCONF or WMI.

Behavior-Observation: A SACM component located on an Endpoint different from the target endpoint observes network traffic related to the target endpoint and conducts the collection task via interpretation of that network traffic.

Collector: A piece of software that acquires information about one or more target endpoints by conducting collection tasks.

A collector can be distributed across multiple endpoints, e.g. across a target endpoint and a SACM component. The separate parts of the collector can communicate with a specialized protocol, such as PA-TNC [RFC5792]. At least one part of a distributed collector has to take on the role of a provider of information by providing SACM interfaces to propagate capabilities and to provide SACM content in the form of collection results.

Configuration: A non-volatile subset of the endpoint attributes of a endpoint that is intended to be unaffected by a normal reboot-cycle.

Configuration is a type of imperative guidance that is stored in files (files dedicated to contain configuration and/ or files that are software components), directly on block devices, or on specific hardware components that can be accessed via corresponding software components. Modification of configuration can be conducted manually or automatically via management (plane) interfaces that support management protocols, such as SNMP or WMI. A change of configuration can occur during both run-time and down-time of an endpoint. It is common practice to scheduled a change of configuration during or directly after the completion of a boot-cycle via corresponding software components located on the target endpoint itself.

Examples: The static association of an IP address and a MAC address in a DHCP server configuration, a directory-path that identifies a log-file directory, a registry entry.

Configuration Drift: The disposition of endpoint characteristics to change over time.

Configuration drift exists for both hardware components and software components. Typically, the frequency and scale of configuration drift of software components is significantly higher than the configuration drift of hardware components.

Consumer: A SACM Role that requires a SACM Component to include SACM Functions enabling it to receive information from other SACM Components.

Content Element: Content elements constitute the payload data (SACM content) transferred via statement Subjects emitted by providers of information. Every content element Subject includes a specific content Subject and a corresponding content metadata Subject.

Content Metadata: Data about content Subjects. Every content-element includes a content metadata Subject. The Subject can include any information element that can annotate the content transferred. Examples include time stamps or data provenance Subjects.

Control Plane: An architectural component that provides common control functions to all SACM components.

Typically used as a term in the context of routing, e.g. [RFC6192]. SACM components may include authentication, authorization, (capability) discovery or negotiation, registration and subscription. The control plane orchestrates the flow on the data plane according to imperative guidance (i.e. configuration) received via the management plane. SACM components with interfaces to the control plane have knowledge of the capabilities of other SACM components within a SACM domain.

Controller: A controller is a SACM Role that is assigned to a SACM component containing control plane functions managing and facilitating information sharing or execute on security functions.

There are three types of SACM controllers: Broker, Proxy, and Repository. Depending on its type, a controller can also contain functions that have interfaces on the data plane.

Data Confidentiality: Defined in [RFC4949] as "the property that data is not disclosed to system entities unless they have been authorized to know the data."

Data In Motion: Data that is being transported via a network; also referred to as "Data in Transit" or "Data in Flight".

Data in motion requires a data model to transfer the data using a specific encoding. Typically, data in motion is serialized (marshalling) into a transport encoding by a provider of information and deserialized (unmarshalling) by a consumer of information. The termination points of provider of information and consumer of information data is transferred between are interfaces. In regard to data in motion, the interpretation of the roles consumer of information and provider of information depends on the corresponding OSI layer (e.g. on layer2: between interfaces connected to a broadcast domain, on layer4: between interfaces that maintain a TCP connection). In the context of SACM, consumer of information and provider of information are SACM components.

Data At Rest: Data that is stored.

Data at rest requires a data model to encode the data to be stored. In the context of SACM, data at rest located on a SACM component can be provided to other SACM components via discoverable capabilities.

Data Integrity: Defined in [RFC4949] as "the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner."

Data Origin: The SACM Component that initially acquired or produced data about an endpoint.

Data Origin enables a SACM component to identify the SACM component that initially acquired or produced data about a (target) endpoint (e.g. via collection from a data source) and made it available to a SACM domain via a SACM statement. Data Origin can be expressed by an endpoint label information element (e.g. to be used as metadata in statement).

Data Plane: Is an architectural component providing operational functions enabling information exchange that is not command and control or management related.

Typically used as a term in the context of routing (and used as a synonym for forwarding plane, e.g. [RFC6192]). In the context of SACM, the data plane is an architectural component providing operational functions to enable a SACM component to provide and consume SACM statements and therefore SACM content, which composes the actual SACM content. The data plane in a SACM domain is used to conduct distributed SACM tasks by transporting SACM content via specific transport encodings and corresponding operations defined by SACM data models.

Data Provenance: An historical record of the sources, origins and evolution, as it pertains to data, that is influenced by inputs, entities, functions and processes.

Additional Information - In the context of SACM, data provenance is expressed as metadata that identifies SACM statements and corresponding content elements a new statement is created from. In a downstream process, this references can cascade, creating a data provenance tree that enables SACM components to trace back the original data sources involved in the creation of SACM statements and take into account their characteristics and trustworthiness.

Data Source: Is an endpoint from which a particular set of attributes and/or attribute values have been collected.

Data Source enables a SACM component to identify - and potentially characterize - a (target) endpoint that is claimed to be the original source of endpoint attributes in a SACM statement. Data Source can be expressed as metadata by an endpoint label information element or a corresponding subject of identifying endpoint attributes.

Endpoint: Defined in [RFC5209] as "any computing device that can be connected to a network."

Additional Information - The [RFC5209] definition continues, "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.

Physical endpoints are always composites that are composed of hardware components and software components. Virtual endpoints are composed entirely of software components and rely on software components that provide functions equivalent to hardware components.

The SACM architecture differentiates two essential categories of endpoints: Endpoints whose security posture is intended to be assessed (target endpoints) and endpoints that are specifically excluded from endpoint posture assessment (excluded endpoints).

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

Endpoint Attribute: Is a discreet endpoint characteristic that is computably observable.

Endpoint Attributes typically constitute Attributes that can be bundled into Subject (e.g. information about a specific network interface can be represented via a set of multiple AVP).

Endpoint Characteristics: The state, configuration and composition of the software components and (virtual) hardware components a target endpoint is composed of, including observable behavior, e.g. sys-calls, log-files, or PDU emission on a network.

In SACM work-flows, (Target) Endpoint Characteristics are represented via Information Elements.

Endpoint Characterization Task: The task of endpoint characterization that uses endpoint attributes that represent distinct endpoint characteristics.

Endpoint Classification: The categorization of of the endpoint into one or more taxonomic structures.

Endpoint classification requires declarative guidance in the form of an endpoint profile, discovery results and potentially collection results. Types, classes or the characteristics of an individual target endpoint are defined via endpoint profiles.

Endpoint Classification Task: The task of endpoint classification that uses an endpoint's characteristics to determine how to categorize the given endpoint into one or more taxonomic structures.

Endpoint Label: A unique label associated with a unique endpoint.

Endpoint specializations have corresponding endpoint label specializations. For example, an endpoint label used on a SACM Component is a SACM Component Label.

Endpoint Management Capabilities: Enterprise IT management capabilities that are tailored to manage endpoint identity, endpoint information, and associated metadata.

Evaluation Task: A task by which an endpoint's asserted attribute value is evaluated against a policy-compliant attribute value.

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

Expected Endpoint Attribute State: The policy-compliant state of an endpoint attribute that is to be compared against.

Sets of expected endpoint attribute states are transported as declarative guidance in target endpoint profiles via the management plane. This, for example, can be a policy, but also a recorded past state. An expected state is represented by an Attribute or a Subject that represents a set of multiple attribute value pairs.

Guidance: Machine-processable input directing SACM processes or tasks.

Examples of such processes/tasks include automated device management, remediation, collection, evaluation. Guidance influences the behavior of a SACM Component and is considered content of the management plane. In the context of SACM, guidance is machine-readable and can be manually or automatically generated or provided. Typically, the tasks that provide guidance to SACM components have a low-frequency and tend to be sporadic.

There are two types of guidance:

Declarative Guidance: Guidance that defines the configuration or state an endpoint is supposed to be in, without providing specific actions or methods to produce that desired state. Examples include Target Endpoint Profiles or network topology based requirements.

Imperative Guidance: Guidance that prescribes specific actions to be conducted or methods to be used in order to achieve an outcome. Examples include a targeted Collection Task or the IP-Address of a SACM Component that provides a registration function.

Prominent examples include: modification of the configuration of a SACM component or updating a target endpoint profile that resides on an evaluator. In essence, guidance is transported via the management plane.

Endpoint Hardware Inventory: The set of hardware components that compose a specific endpoint representing its hardware configuration.

Hardware Component: A distinguishable physical component used to compose an endpoint.

The composition of an endpoint can be changed over time by adding or removing hardware components. In essence, every physical endpoint is potentially a composite of multiple hardware components, typically resulting in a hierarchical composition of hardware components. The composition of hardware components is based on interconnects provided by specific hardware types (e.g. FRU in a chassis are connected via redundant busses). In general, a hardware component can be distinguished by its serial number. Occasionally, hardware components are referred to as power sucking aliens.

Information Element: A representation of information about physical and virtual "objects of interest".

Information elements are the building blocks that constitute the SACM information model. In the context of SACM, an information element that expresses a single value with a specific name is referred to as an Attribute (analogous to an attribute-value-pair). A set of attributes that is bundled into a more complex composite information element is referred to as a Subject. Every information element in the SACM information model has a unique name. Endpoint attributes or time stamps, for example, are represented as information elements in the SACM information model.

Information Model: 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] distinguishes an information model as being protocol and implementation neutral whereas a data model would provide such details. The purpose of the SACM information model is to ensure interoperability between SACM data models (that are used as transport encoding) and to provide a standardized set of information elements for communication between SACM components.

Interaction Model: The definition of specific sequences regarding the exchange of messages (data in motion), including, for example, conditional branching, thresholds and timers.

An interaction model, for example, can be used to define operations, such as registration or discovery, on the control plane. A composition of data models for data in motion and a corresponding interaction model is a protocol.

Internal Collector: A collector that runs on a target endpoint to acquire information from that target endpoint.

Management Plane: An architectural component providing common functions to steer the behavior of SACM components, e.g. their behavior on the control plane.

Typically, a SACM component can fulfill its purpose without continuous input from the management plane. In contrast, without continuous availability of control plane functions a typical SACM component could not function properly. In general, interaction on the management plane is less frequent and less regular than on the control plane. Input via the management plane can be manual (e.g. via a CLI), or can be automated via management plane functions that are part of other SACM components.

Network Address: A layer-specific address that follows a layer-specific address scheme.

The following characteristics are a summery derived from the Common Information Model and ITU-T X.213. Each Network Interface of a specific layer can be associated with one or more addresses appropriate for that layer. There is no guarantee that a network address is globally unique. A dedicated authority entity can provide a level of assurance that a network address is unique in its given scope. In essence, there is always a scope to a network address, in which it is intended to be unique.

Examples include: physical Ethernet port with a MAC address, layer 2 VLAN interface with a MAC address, layer 3 interface with multiple IPv6 addresses, layer 3 tunnel ingress or egress with an IPv4 address.

Network Interface: An Endpoint is connected to a network via one or more Network Interfaces. Network Interfaces can be physical (Hardware Component) or logical (virtual Hardware component, i.e. a dedicated Software Component). Network Interfaces of an Endpoint can operate on different layers, most prominently what is now commonly called layer 2 and 3. Within a layer, interfaces can be nested.

In SACM, the association of Endpoints and Network Addresses via Network Interfaces is vital to maintain interdependent autonomous processes that can be targeted at Target Endpoints, unambiguously.

Examples include: physical Ethernet port, layer 2 VLAN interface, a MC-LAG setup, layer 3 Point-to-Point tunnel ingress or egress.

Metadata: Data about data.

In the SACM information model, data is referred to as Content. Metadata about the content is referred to as Content-Metadata, respectively. Content and Content-Metadata are combined into Subjects called Content-Elements in the SACM information model. Some information elements defined by the SACM information model can be part of the Content or the Content-Metadata. Therefore, if an information element is considered data or data about data depends on which kind of Subject it is associated with. The SACM information model also defines metadata about the data origin via the Subject Statement-Metadata. Typical examples of metadata are time stamps, data origin or data source.

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 SACM to represent the configuration and state information that is collected from a target endpoint in the form of endpoint attributes (e.g. software/ hardware inventory, configuration settings, dynamically assigned addresses). This information may constitute one or more 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 configuration or state (e.g. configuration setting, installed software, hardware) represented via endpoint attributes. The phrase "features of the endpoint" highlighted above refers to installed software or software components.

Provider: A provider is a SACM role assigned to a SACM component that provides role-specific functions to provide information to other SACM components.

Repository: A repository is a controller that contains functions to consume, store and provide information of a particular kind.

Such information is typically data transported on the data plane, but potentially also data and metadata from the control and management plane. A single repository may provide the functions of more than one specific repository type (i.e. configuration baseline repository, assessment results repository, etc.)

SACM Broker Controller: A SACM Broker Controller is a controller that contains control plane functions to provide and/or connect services on behalf of other SACM components via interfaces on the control plane.

A broker may provide, for example, authorization services and find, upon request, SACM components providing requested services.

SACM Component: Is a component, as defined in [I-D.ietf-i2nsf-terminology], that is composed of SACM capabilities.

In the context of SACM, a set of SACM functions composes a SACM component. A SACM component conducts SACM tasks, acting on control plane, data plane and/or management plane via corresponding SACM interfaces. SACM defines a set of standard components (e.g. a collector, a broker, or a data store). A SACM component contains at least a basic set of control plane functions and can contain data plane and management plane functions. A SACM component residing on an endpoint assigns one or more SACM roles to the corresponding endpoint due to the SACM functions it is composed of. A SACM component "resides on" an endpoint and an endpoint "contains" a SACM component, correspondingly. For example, a SACM component that is composed solely of functions that provide information would only take on the role of a provider.

SACM Component Discovery: The task of discovering the capabilities provided by SACM components within a SACM domain.

This is likely to be performed via an appropriate set of control plane functions.

SACM Component Label: A specific endpoint label that is used to identify a SACM component.

In content-metadata, this label is called data origin.

SACM Content: The payload provided by SACM components to the SACM domain on the data plane.

SACM content includes the SACM data models.

SACM Domain: Endpoints that include a SACM component compose a SACM domain.

(To be revised, additional definition content TBD, possible dependencies to SACM architecture)

SACM Function: A behavioral aspect of a SACM component that provides external SACM Interfaces or internal interfaces to other SACM Functionse.

For example, a SACM Function with SACM Interfaces on the Control Plane can provide a brokering function to other SACM Components. Via Data Plane interfaces, a SACM Function can act as a provider and/or as a consumer of information. SACM Functions can be propagated as the Capabilities of a SACM Component and can be discovered by or negotiated with other SACM Components.

SACM Interface: An interface, as defined in [I-D.ietf-i2nsf-terminology], that provides SACM-specific operations.

[I-D.ietf-i2nsf-terminology] defines interface as a "set of operations one object knows it can invoke on, and expose to, another object," and further defines interface by stating that an interface "decouples the implementation of the operation from its specification. An interface is a subset of all operations that a given object implements. The same object may have multiple types of interfaces to serve different purposes."

In the context of SACM, SACM Functions provide SACM Interfaces on the management, control, or data plane. Operations a SACM Interface provides are based on corresponding data model defined by SACM. SACM Interfaces are used for communication between SACM components.

SACM Proxy Controller: A SACM Proxy Controller is a controller that provides data plane and control plane functions, information, or services on behalf of another component, which is not directly participating in the SACM architecture.

SACM Role: Is a role, as defined in [I-D.ietf-i2nsf-terminology], that requires the SACM Component assuming the role to bear a set of SACM functions or interfaces.

SACM Roles provide three important benefits. First, it enables different behavior to be supported by the same Component for different contexts. Second, it enables the behavior of a Component to be adjusted dynamically (i.e., at runtime, in response)to changes in context, by using one or more Roles to define the behavior desired for each context. Third, it decouples the Roles of a Component from the Applications that use that Component."

In the context of SACM, SACM roles are associated with SACM components and are defined by the set of functions and interfaces a SACM component includes. There are three SACM roles: provider, consumer, and controller. The roles associated with a SACM component are determined by the purpose of the SACM functions and corresponding SACM interfaces the SACM component is composed of.

SACM Statement: Is an assertion that is made by a SACM Component.

Security Automation: The process of which security alerts can be automated through the use of different components to monitor, analyze and assess endpoints and network traffic for the purposes of detecting misconfigurations, misbehaviors or threats.

Security Automation is intended to identify target endpoints that cannot be trusted (see "trusted" in [RFC4949]. This goal is achieved by creating and processing evidence (assessment statements) that a target endpoint is not a trusted system [RFC4949].

Software Package: A generic software package (e.g. a text editor).

Software Component: A software package installed on an endpoint.

The software component may include a unique serial number (e.g. a text editor associated with a unique license key).

Software Instance: A running instance of a software component.

For example, on a multi-user system, one logged-in user has one instance of a text editor running and another logged-in user has another instance of the same text editor running, or on a single-user system, a user could have multiple independent instances of the same text editor running.

State: A volatile set of endpoint attributes of a (target) endpoint that is affected by a reboot-cycle.

Local state is created by the interaction of components with other components via the control plane, via processing data plane payload, or via the functional properties of local hardware and software components. Dynamic configuration (e.g. IP address distributed dynamically via an address distribution and management services, such as DHCP) is considered state that is the result of the interaction with another component (e.g. provided by a DHCP server with a specific configuration).

Examples: The static association of an IP address and a MAC address in a DHCP server configuration, a directory-path that identifies a log-file directory, a registry entry.

Statement: A statement is the root/top-level subject defined in the SACM information model.

A statement is used to bundle Content Elements into one subject and includes metadata about the data origin.

Subject: A semantic composite information element pertaining to a system entity that is a target endpoint.

Like Attributes, subjects have a name and are composed of attributes and/or other subjects. Every IE that is part of a subject can have a quantitiy associated with it (e.g. zero-one, none-unbounded). The content IE of a subject can be an unordered or an ordered list.

In contrast to the definitions of subject provided by [RFC4949], a subject in the scope of SACM is neither "a system entity that causes information to flow among objects or changes the system state" nor "a name of a system entity that is bound to the data items in a digital certificate".

In the context of SACM, a subject is a semantic composite of information elements about a system entity that is a target endpoint. Every acquirable subject-as defined in the scope of SACM-about a target endpoint represents and therefore identifies every subject-as defined by [RFC4949]-that is a component of that target endpoint. The semantic difference between both definitions can be subtle in practice and is in consequence important to highlight.

Supplicant: A component seeking to be authenticated via the control plane for the purpose of participating in a SACM domain.

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."

Target Endpoint: Is an endpoint that is under assessment at some point in, or region of, time.

Every endpoint that is not specifically designated as an excluded endpoint is a target endpoint. A target endpoint is not part of a SACM domain unless it contains a SACM component (e.g. a SACM component that publishes collection results coming from an internal collector).

A target endpoint is similar to a device that is a Target of Evaluation (TOE) as defined in Common Criteria and as referenced by {{RFC4949}.

Target Endpoint Address: An address that is layer specific and which follows layer specific address schemes.

Each interface of a specific layer can be associated with one or more addresses appropriate for that layer. There is no guarantee that an address is globally unique. In general, there is a scope to an address in which it is intended to be unique.

Examples include: physical Ethernet port with a MAC address, layer 2 VLAN interface with a MAC address, layer 3 interface with multiple IPv6 addresses, layer 3 tunnel ingress or egress with an IPv4 address.

Target Endpoint Characterization: The description of the distinctive nature of a target endpoint, that is based on its characteristics.

Target Endpoint Characterization Record: A set of endpoint attributes about a target endpoint that was encountered in a SACM domain, which are associated with that target endpoint as a result of a Target Endpoint Characterization Task.

A characterization record is intended to be a representation of an endpoint. It cannot be assured that a record distinctly represents a single target endpoint unless a set of one or more endpoint attributes that compose a unique set of identifying endpoint attributes are included in the record. Otherwise, the set of identifying attributes included in a record can match more than one target endpoints, which are - in consequence - indistinguishable to a SACM domain until more qualifying endpoint attributes can be acquired and added to the record. A characterization record is maintained over time in order to assert that acquired endpoint attributes are either about an endpoint that was encountered before or an endpoint that has not been encountered before in a SACM domain. A characterization record can include, for example, acquired configuration, state or observed behavior of a specific target endpoint. Multiple and even conflicting instances of this information can be included in a characterization record by using timestamps and/or data origins to differentiate them. The endpoint attributes included in a characterization record can be used to re-identify a distinct target endpoint over time. Classes or profiles can be associated with a characterization record via the Classification Task in order to guide collection, evaluation or remediation tasks.

Target Endpoint Characterization Task: An ongoing task of continuously adding acquired endpoint attributes to a corresponding record. The TE characterization task manages the representation of encountered target endpoints in the SACM domain in the form of characterization records. For example, the output of a target endpoint discovery task or a collection task can be processed by the characterization task and added to the record. The TE characterization Task also manages these representations of target endpoints encountered in the SACM domain by splitting or merging the corresponding records as new or more refined endpoint attributes become available.

Target Endpoint Classification Task: The task of associating a class from an extensible list of classes with an endpoint characterization record. TE classes function as imperative and declarative guidance for collection, evaluation, remediation and security posture assessment in general.

Target Endpoint Discovery Task: The ongoing task of detecting previously unknown interaction of a potential target endpoint in the SACM domain. TE Discovery is not directly targeted at a specific target endpoint and therefore an un-targeted task. SACM Components conducting the discovery task as a part of their function are typically distributed and located, for example, on infrastructure components or collect from those remotely via appropriate interfaces. Examples of infrastructure components that are of interest to the discovery task include routers, switches, VM hosting or VM managing components, AAA servers, or servers handling dynamic address distribution.

Target Endpoint Identifier: The target endpoint discovery task and the collection tasks can result in a set of identifying endpoint attributes added to a corresponding Characterization Record. This subset of the endpoint attributes included in the record is used as a target endpoint identifier, by which a specific target endpoint can be referenced. Depending on the available identifying attributes, this reference can be ambiguous and is a "best-effort" mechanism. Every distinct set of identifying endpoint attributes can be associated with a target endpoint label that is unique in a SACM domain.

Target Endpoint Label: An endpoint label that identifies a specific target endpoint.

Target Endpoint Profile: A bundle of expected or desired component composition, configurations and states that is associated with a target endpoint.

The corresponding task by which the association with a target endpoint takes places is the endpoint classification task. The task by which an endpoint profile is created is the endpoint characterization task. A type or class of target endpoints can be defined via a target endpoint profile. Examples include: printers, smartphones, or an office PC.

In respect to [RFC4949], a target endpoint profile is a protection profile as defined by Common Criteria (analogous to the target endpoint being the target of evaluation).

SACM Task: Is a task conducted within the scope of a SACM domain by one or more SACM functions that achieves a SACM-defined outcome.

A SACM task can be triggered by other operations or functions (e.g. a query from another SACM component or an unsolicited push on the data plane due to an ongoing subscription). A task is part of a SACM process chain. A task starts at a given point in time and ends in a deterministic state. With the exception of a collection task, a SACM task consumes SACM statements provided by other SACM components. The output of a task is a result that can be provided (e.g. published) on the data plane.

The following tasks are defined by SACM:

Target Endpoint Discovery

Target Endpoint Characterization

Target Endpoint Classification

Collection

Evaluation [TBD]

Information Sharing [TBD]

SACM Component Discovery

SACM Component Authentication [TBD]

SACM Component Authorization [TBD]

SACM Component Registration [TBD]

Timestamps : Defined in [RFC4949] as "with respect to a data object, a label or marking in which is recorded the time (time of day or other instant of elapsed time) at which the label or marking was affixed to the data object".

A timestamp always requires context, i.e. additional information elements that are associated with it. Therefore, all timestamps wrt information elements are always metadata. Timestamps in SACM Content Elements may be generated outside a SACM Domain and may be encoded in an unknown representation. Inside a SACM domain the representation of timestamps is well-defined and unambiguous.

Virtual Endpoint: An endpoint composed entirely of logical system components (see [RFC4949]).

The most common example is a virtual machine/host running on a target endpoint. Effectively, target endpoints can be nested and at the time of this writing the most common example of target endpoint characteristics about virtual components is the EntLogicalEntry in [RFC6933].

Vulnerability Assessment: An assessment specifically tailored to determining whether a set of endpoints is vulnerable according to the information contained in the vulnerability description information.

Vulnerability Description Information: Information pertaining to the existence of a flaw or flaws in software, hardware, and/or firmware, which could potentially have an adverse impact on enterprise IT functionality and/or security.

Vulnerability description information should contain enough information to support vulnerability detection.

Vulnerability Detection Data: A type of imperative guidance extracted or derived from vulnerability description information that describes the specific mechanisms of vulnerability detection that is used by an enterprise's vulnerability management capabilities to determine if a vulnerability is present on an endpoint.

Vulnerability Management Capabilities: An IT management capability tailored toward managing endpoint vulnerabilities and associated metadata on an ongoing basis by ingesting vulnerability description information and vulnerability detection data, and performing vulnerability assessments.

Vulnerability assessment capabilities: An assessment capability that is tailored toward determining whether a set of endpoints is vulnerable according to vulnerability description information.

Workflow: A workflow is a modular composition of tasks that can contain loops, conditionals, multiple starting points and multiple endpoints.

The most prominent workflow in SACM is the assessment workflow.

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

Changes from version 00 to version 01:

- 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.

Changes from version 01 to version 02:

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

Changes from version 02 to version 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.

Changes from version 03 to version 04:

- TODO

Changes from version 04 to version 05:

- TODO

Changes from version 05 to version 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.

Changes from version 06 to version 07:

- Added Contributors section.

- Updated author list.

- Changed title from "Terminology for Security Assessment" to "Secure Automation and Continuous Monitoring (SACM) Terminology".

- Changed abbrev from "SACM-Terms" to "SACM Terminology".

- Added appendix The Attic to stash terms for future updates.

- Added Authentication, Authorization, Data Confidentiality, Data Integrity, Data Origin, Data Provenance, SACM Component, SACM Component Discovery, Target Endpoint Discovery.

- Major updates to Building Block, Function, SACM Role, Target Endpoint.

- Minor updates to Broker, Capability, Collection Task, Evaluation Task, Posture.

- Relabeled Role to SACM Role, Endpoint Target to Target Endpoint, Endpoint Discovery to Endpoint Identification.

- Moved Asset Targeting, Client, Endpoint Identification to The Attic.

- Endpoint Attributes added as a TODO.

- Changed the structure of the Change Log.

Changes from version 07 to version 08:

- Added Assertion, Collection Result, Collector, Excluded Endpoint, Internal Collector, Network Address, Network Interface, SACM Domain, Statement, Target Endpoint Identifier, Target Endpoint Label, Timestamp.

- Major updates to Attributes, Broker, Collection Task, Consumer, Controller, Control Plane, Endpoint Attributes, Expected Endpoint State, SACM Function, Provider, Proxy, Repository, SACM Role, Target Endpoint.

- Minor updates to Asset, Building Block, Data Origin, Data Source, Data Provenance, Endpoint, Management Plane, Posture, Posture Attribute, SACM Component, SACM Component Discovery, Target Endpoint Discovery.

- o Relabeled Function to SACM Function.

Changes from version 08 to version 09:

- o Updated author list.
- o Added Data Plane, Endpoint Characterization, Endpoint Classification, Guidance, Interaction Model, Software Component, Software Instance, Software Package, Statement, Target Endpoint Profile, SACM Task.
- o Removed Building Block.
- o Major updates to Control Plane, Endpoint Attribute, Expected Endpoint State, Information Model, Management Plane.
- o Minor updates to Attribute, Capabilities, SACM Function, SACM Component, Collection Task.
- o Moved Asset Characterization to The Attic.

Changes from version 09 to version 10:

- o Added Configuration Drift, Data in Motion, Data at Rest, Endpoint Management Capability, Hardware Component, Hardware Inventory, Hardware Type, SACM Interface, Target Endpoint Characterization Record, Target Endpoint Characterization Task, Target Endpoint Classification Task, Target Endpoint Discovery Task, Vulnerability Description Information, Vulnerability Detection Data, Vulnerability Management Capability, Vulnerability Assessment
- o Added references to i2nsf definitions in Capability, SACM Component, SACM Interface, SACM Role.
- o Added i2nsf Terminology I-D Reference.
- o Major Updates to Endpoint, SACM Task, Target Endpoint Identifier.
- o Minor Updates to Guidance, SACM Component Discovery, Target Endpoint Label, Target Endpoint Profile.
- o Relabeled SACM Task
- o Removed Target Endpoint Discovery

Changes from version 10 to version 11:

- o Added Content Element, Content Metadata, Endpoint Label, Information Element, Metadata, SACM Component Label, Workflow.

- o Major Updates to Assessment, Capability, Collector, Endpoint Management Capabilities, Guidance, Vulnerability Assessment Capabilities, Vulnerability Detection Data, Vulnerability Assessment Capabilities.

- o Minor updates to Collection Result, Control Plane, Data in Motion, Data at Rest, Data Origin, Network Interface, Statement, Target Endpoint Label.

- o Relabeled Endpoint Management Capability, Vulnerability Management Capability, Vulnerability Assessment.

Changes from version 11 to version 12:

- o Added Configuration, Endpoint Characteristic, Event, SACM Content, State, Subject.

- o Major Updates to Assertion, Data in Motion, Data Provenance, Data Source, Interaction Model.

- o Minor Updates to Attribute, Control Plane, Data Origin, Data Provenance, Expected Endpoint State, Guidance, Target Endpoint Classification Task, Vulnerability Detection Data.

Changes from version 12 to version 13:

- o Added Virtual Component.

- o Major Updates to Capability, Collection Task, Hardware Component, Hardware Type, Security Automation, Subject, Target Endpoint, Target Endpoint Profile.

- o Minor Updates to Assertion, Data Plane, Endpoint Characteristics.

Changes from version 13 to version 14:

- o Handled a plethora of issues listed in GitHub.

- o Pruned some commonly understood terms.

- o Narrowing term labels per their definitions.

- o In some cases, excised expositional text.

o Where expositional text was left intact, it has been separated from the actual definition of a term.

Changes from version 14 to version 16:

o moved obsolete definitions into the Appendix (attic).

7. Contributors

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

Email: david.waltermire@nist.gov

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

Email: adam.w.montville@gmail.com

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

Email: ietfdbh@comcast.net

Brian Ford
Lancope
3650 Brookside Parkway, Suite 500
Alpharetta, GA 30022
USA

Email: bford@lancope.com

Merike Kaeo
Double Shot Security
3518 Fremont Avenue North, Suite 363
Seattle, WA 98103
USA

Email: merike@doubleshotsecurity.com

8. References

8.1. Normative References

[RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5792, DOI 10.17487/RFC5792, March 2010, <https://www.rfc-editor.org/info/rfc5792>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <https://www.rfc-editor.org/info/rfc6933>.

8.2. Informative References

[I-D.ietf-i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-06 (work in progress), July 2018.

[I-D.ietf-netmod-entity] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-08 (work in progress), January 2018.

[I-D.ietf-sacm-vuln-scenario] Coffin, C., Cheikes, B., Schmidt, C., Haynes, D., Fitzgerald-McKay, J., and D. Waltermire, "SACM Vulnerability Assessment Scenario", draft-ietf-sacm-vuln-scenario-02 (work in progress), September 2016.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <https://www.rfc-editor.org/info/rfc5209>.

[RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <https://www.rfc-editor.org/info/rfc6192>.

[X.1252] "ITU-T X.1252 (04/2010)", n.d..

Appendix A. The Attic

The following terms are stashed for now and will be updated later:

Asset: Is a system resource, as defined in [RFC4949], that may be composed of other assets.

Examples of Assets include: Endpoints, Software, Guidance, or X.509 public key certificates. An asset is not necessarily owned by an organization.

Asset Management: The IT process by which assets are provisioned, updated, maintained and deprecated.

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

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.

Client: An architectural component receiving services from another architectural component.

Endpoint Identification (TBD per list; was "Endpoint Discovery"): The process by which an endpoint can be identified.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Jarrett Lu
Oracle Corporation
4180 Network Circle
Santa Clara, CA 95054
USA

Email: jarrett.lu@oracle.com

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95138
USA

Email: john.sc.strassner@huawei.com

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

Email: ncamwing@cisco.com

Adam Montville
Center for Internet Security
31 Tech Valley Drive
East Greenbush, NY 12061
USA

Email: adam.w.montville@gmail.com

Security Automation and Continuous Monitoring WG   D. Waltermire
Internet-Draft   NIST
Intended status: Informational   D. Harrington
Expires: January 2, 2016   Effective Software
                                                            July 1, 2015

Endpoint Security Posture Assessment - Enterprise Use Cases
draft-ietf-sacm-use-cases-10

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.

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- 23
  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
Authors' Addresses 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:

- o securely collecting and aggregating configuration and operational data, and

- o 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:

- o concepts that are expressed as building blocks in this document,

- o characteristics to inform development of a requirements document

- o information concepts to inform development of an information model document, and

- o 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:

- o Collecting the attributes of a given endpoint;
- o Making the attributes available for evaluation and action; and
- o 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 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 the 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 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 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 and privacy 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 gain access to an item of interest. This may include personal data, private keys, software and configuration state that can be used to inform an attack against the network and endpoints, and other sensitive information. It is important that security and privacy considerations in the related documents identify methods to both identify and prevent such activity.

For consideration are 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. For exchanged information, there should be a means to authenticate the origin of the information. This is important where tracking the provenance of data is needed. Also, for any systems that store information that could be used for unauthorized or 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 gramatical 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 Identification", "Compromised Endpoint Identification", "Suspicious Endpoint Behavior", "Traditional endpoint assessment with stored results", "NAC/NAP connection with no stored results using an endpoint evaluator", and "NAC/NAP connection with no stored results using a third-party evaluator".

- 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"

- 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.

- 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.

- o Extracted the functionality from the examples and made the examples less prominent.

- o 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

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

Authors’ Addresses

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

Email: david.waltermire@nist.gov

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

Email: ietfdbh@comcast.net