SACM Working Group
Internet-Draft
Intended status: Standards Track
Expires: 10 January 2022

A. Montville
B. Munyan
CIS

9 July 2021

Security Automation and Continuous Monitoring (SACM) Architecture
draft-ietf-sacm-arch-13

Abstract

This document defines an architecture enabling a cooperative Security Automation and Continuous Monitoring (SACM) ecosystem. This work is predicated upon information gleaned from SACM Use Cases and Requirements ([RFC7632] and [RFC8248] respectively), and terminology as found in [I-D.ietf-sacm-terminology].

WORKING GROUP: The source for this draft is maintained in GitHub. Suggested changes should be submitted as pull requests at https://github.com/sacmwg/ietf-mandm-sacm-arch/. Instructions are on that page as well.

Status of This Memo

Copyright Notice

Table of Contents

```
   1.  Introduction  . . . . . . . . . . . . . . . . . . . . . . . .   4
     1.1.  Requirements notation . . . . . . . . . . . . . . . . . .   4
   2.  Terms and Definitions . . . . . . . . . . . . . . . . . . . .   4
   3.  Architectural Overview  . . . . . . . . . . . . . . . . . . .   8
     3.1.  Producer  . . . . . . . . . . . . . . . . . . . . . . . .   9
     3.2.  Consumer  . . . . . . . . . . . . . . . . . . . . . . . .   9
     3.3.  Integration Service . . . . . . . . . . . . . . . . . . .   9
     3.4.  Payload/Message . . . . . . . . . . . . . . . . . . . . .  10
     3.5.  Payload Categorization  . . . . . . . . . . . . . . . . .  10
       3.5.1.  Topic-centric . . . . . . . . . . . . . . . . . . . .  10
       3.5.2.  Payload-centric . . . . . . . . . . . . . . . . . . .  11
     3.6.  Capabilities  . . . . . . . . . . . . . . . . . . . . . .  11
     3.7.  Interaction Categories  . . . . . . . . . . . . . . . . .  12
       3.7.1.  Broadcast . . . . . . . . . . . . . . . . . . . . . .  12
       3.7.2.  Directed  . . . . . . . . . . . . . . . . . . . . . .  12
   4.  SACM Role-based Architecture  . . . . . . . . . . . . . . . .  13
     4.1.  Architectural Roles/Components  . . . . . . . . . . . . .  14
       4.1.1.  Manager . . . . . . . . . . . . . . . . . . . . . . .  14
       4.1.2.  Orchestrator(s) . . . . . . . . . . . . . . . . . . .  14
       4.1.3.  Repositories  . . . . . . . . . . . . . . . . . . . .  15
       4.1.4.  Integration Service . . . . . . . . . . . . . . . . .  15
     4.2.  Downstream Uses . . . . . . . . . . . . . . . . . . . . .  16
       4.2.1.  Reporting . . . . . . . . . . . . . . . . . . . . . .  16
       4.2.2.  Analytics . . . . . . . . . . . . . . . . . . . . . .  16
     4.3.  Sub-Architectures . . . . . . . . . . . . . . . . . . . .  16
       4.3.1.  Collection Sub-Architecture . . . . . . . . . . . . .  16
       4.3.2.  Evaluation Sub-Architecture . . . . . . . . . . . . .  19
   5.  Ecosystem Interactions  . . . . . . . . . . . . . . . . . . .  21
     5.1.  Manager . . . . . . . . . . . . . . . . . . . . . . . . .  21
     5.2.  Component Registration  . . . . . . . . . . . . . . . . .  22
     5.3.  Administrative Interface  . . . . . . . . . . . . . . . .  23
       5.3.1.  Capability Advertisement Handshake  . . . . . . . . .  23
       5.3.2.  Health Check  . . . . . . . . . . . . . . . . . . . .  23
       5.3.3.  Heartbeat . . . . . . . . . . . . . . . . . . . . . .  23
       5.3.4.  Capability-specific Requests  . . . . . . . . . . . .  24
     5.4.  Status Notifications  . . . . . . . . . . . . . . . . . .  24
     5.5.  Component Interactions  . . . . . . . . . . . . . . . . .  24
       5.5.1.  Initiate Ad-Hoc Collection  . . . . . . . . . . . . .  24
       5.5.2.  Coordinate Periodic Collection  . . . . . . . . . . .  24
       5.5.3.  Coordinate Observational/Event-based Collection . . .  25
       5.5.4.  Persist Collected Posture Attributes  . . . . . . . .  26
```

```
      5.5.5.  Initiate Ad-Hoc Evaluation . . . . . . . . . . . . .  26
      5.5.6.  Coordinate Periodic Evaluation  . . . . . . . . . . .  26
      5.5.7.  Coordinate Change-based Evaluation . . . . . . . . .  27
      5.5.8.  Queries . . . . . . . . . . . . . . . . . . . . . . .  27
  6.  Operations  . . . . . . . . . . . . . . . . . . . . . . . . .  27
    6.1.  Component Registration  . . . . . . . . . . . . . . . . .  27
      6.1.1.  Request Payload . . . . . . . . . . . . . . . . . . .  28
      6.1.2.  Request Processing  . . . . . . . . . . . . . . . . .  28
      6.1.3.  Response Payload  . . . . . . . . . . . . . . . . . .  29
      6.1.4.  Response Processing . . . . . . . . . . . . . . . . .  29
    6.2.  Administrative Interface  . . . . . . . . . . . . . . . .  29
      6.2.1.  Capability Advertisement Handshake  . . . . . . . . .  29
      6.2.2.  Health Check  . . . . . . . . . . . . . . . . . . . .  31
      6.2.3.  Heartbeat . . . . . . . . . . . . . . . . . . . . . .  32
    6.3.  Status Notification . . . . . . . . . . . . . . . . . . .  33
      6.3.1.  Request Payload . . . . . . . . . . . . . . . . . . .  34
      6.3.2.  Request Processing  . . . . . . . . . . . . . . . . .  34
      6.3.3.  Response Payload  . . . . . . . . . . . . . . . . . .  34
      6.3.4.  Response Processing . . . . . . . . . . . . . . . . .  34
    6.4.  Initiate Ad-Hoc Collection  . . . . . . . . . . . . . . .  34
      6.4.1.  SACM Producer to Orchestrator . . . . . . . . . . . .  36
      6.4.2.  Orchestrator to Posture Collection Service  . . . . .  36
      6.4.3.  Posture Collection Service to Posture Attribute
              Repository  . . . . . . . . . . . . . . . . . . . . .  38
    6.5.  Initiate Ad-Hoc Evaluation  . . . . . . . . . . . . . . .  39
  7.  Privacy Considerations  . . . . . . . . . . . . . . . . . . .  39
  8.  Security Considerations . . . . . . . . . . . . . . . . . . .  39
  9.  IANA Considerations . . . . . . . . . . . . . . . . . . . . .  39
    9.1.  Component Types . . . . . . . . . . . . . . . . . . . . .  39
    9.2.  Component Capabilities  . . . . . . . . . . . . . . . . .  40
      9.2.1.  Heartbeat . . . . . . . . . . . . . . . . . . . . . .  40
      9.2.2.  Status Notification (Publish) . . . . . . . . . . . .  40
      9.2.3.  Status Notification (Subscribe) . . . . . . . . . . .  40
  10. References  . . . . . . . . . . . . . . . . . . . . . . . . .  40
    10.1.  Normative References . . . . . . . . . . . . . . . . . .  40
    10.2.  Informative References . . . . . . . . . . . . . . . . .  41
  Appendix A.  Security Domain Workflows  . . . . . . . . . . . . .  43
    A.1.  IT Asset Management . . . . . . . . . . . . . . . . . . .  43
      A.1.1.  Components, Capabilities and Workflow(s)  . . . . . .  44
    A.2.  Vulnerability Management  . . . . . . . . . . . . . . . .  44
      A.2.1.  Components, Capabilities and Workflow(s)  . . . . . .  45
    A.3.  Configuration Management  . . . . . . . . . . . . . . . .  46
      A.3.1.  Components, Capabilities and Workflow(s)  . . . . . .  47
  Authors' Addresses  . . . . . . . . . . . . . . . . . . . . . . .  49
```

1. Introduction

The purpose of this draft is to define an architectural approach for a SACM Domain, based on the spirit of use cases found in [RFC7632] and requirements found in [RFC8248]. This approach gains the most advantage by supporting a variety of collection systems, and intends to enable a cooperative ecosystem of tools from disparate sources with minimal operator configuration.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, BCP 14 [RFC2119].

2. Terms and Definitions

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."

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.

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.

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

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

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.

Consumer: A SACM Role that requires a SACM Component to include SACM Functions enabling it to receive information from other SACM Components.

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.

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.)

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].

SIEM: TBD

SOAR: TBD

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).

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].

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.

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. Architectural Overview

The generic approach proposed herein recognizes the need to obtain information from existing and future state collection systems, and makes every attempt to respect [RFC7632] and [RFC8248]. At the foundation of any architecture are entities, or components, that need to communicate. They communicate by sharing information, where, in a given flow, one or more components are consumers of information and one or more components are providers of information. Different roles within a cooperative ecosystem may act as both Producers and Consumers of SACM-relevant information.

```
          +-------+--------+
          | SACM Component |
          |   (Producer)   |
          +-------+--------+
                  |
                  |
   +--------------v-----------------+
   |      Integration Service       |
   +--------------+-----------------+
                  |
                  |
          +-------v--------+
          | SACM Component |
          |   (Consumer)   |
          +----------------+
```

Figure 1: Basic Architectural Structure

3.1. Producer

A Producer can be described as an abstraction that refers to an entity capable of sending SACM-relevant information to one or many Consumers. In general, information (a "payload") is produced to a particular topic, subscribed to by one or more Consumers. Producers need not be concerned about any specifics of the payload it is providing to a given topic. A Producer may, for example, publish posture collection instructions to collector topics.

3.2. Consumer

A Consumer can be described as an abstraction that refers to an entity capable of receiving SACM-relevant information from one or many Producers. A Consumer acts as a subscriber to a given topic (or set of topics), enabling it to receive event notifications when a Producer provides a payload to that topic or topics. Consumers receive payloads and act upon them according to their capabilities. A Consumer may, for example, subscribe to a posture collection topic to receive and act upon, collection instructions.

3.3. Integration Service

The Integration Service acts as the broker between Producers and Consumers; acting as the destination for Producers to publish payloads, and as the source for Consumers subscribing to those payloads.

SACM Components are intended to interact with other SACM Components. These interactions can be thought of, at the architectural level, as the combination of interfaces with their supported operations. Each interaction will convey a classified payload of information. This classification of payload information allows Consumers to subscribe to only the classifications to which they are capable of handling. The payload information should contain subdomain-specific characteristics and/or instructions.

3.4. Payload/Message

The payload (sometimes referred to as a "message" or "message payload") is the unit of data involved in any given interaction between two SACM components. The payload MAY be used to convey the semantic meaning of the operation to be performed. Protocols such as [RFC6120] achieves this meaning through XML namespace identification within a "<message/>" or "<iq/>" stanza. Topic-centric protocols such as [MQTT] convey the meaning of payloads through topic naming techniques. Both methods require connected components to verify message payloads according to their respective capabilities.

With respect to the Integration Service, the payload is simply an array of bytes, so the data contained within it is not required to convey a specific format or meaning to the Integration Service. The serialization of the payload combined with the payload categorization provides meaning within the SACM context.

3.5. Payload Categorization

Within the SACM ecosystem, categorization of payloads and their transport provide the context through which various capabilities are achieved. Two types of payload categorization can be described.

3.5.1. Topic-centric

Topic-centric payload categorization allows for a broad spectrum of payloads by characterizing those payloads through the Integration Service topic. In this categorization, the topic name becomes a label attached to the payload to which the Integration Service matches against known subscriptions. The topic becomes the operational context for the payload. Topic-centric categorization allows for any payload to be sent to any topic, but requires that SACM consumers parse the payloads to determine whether or not they have the capability to act on those payloads.

When interacting using a topic-centric payload categorization, topic naming conventions SHOULD provide an adequate amount of information to be deterministic regarding the purpose of the interaction. For example, a topic named "/notification/collection/oval" would indicate that (a) the topic is a broadcast/notification (publish/subscribe) topic, (b) subscribers to this topic are performing a "collection" action, and (c) the payloads published to the topic are represented using the OVAL serialization format.

3.5.2. Payload-centric

Payload-centric categorization encapsulates the intent of an interaction within the message payload itself, using an identifying token, tag, or namespace identifier. This method allows for the limitation of message types, and therefore increases the extensibility of message payloads.

Payload-centric categorization allows for modularization and specification of extensions, and for plugin-based support of capabilities based the categorization. XMPP is an example of utilization of payload-centric categorization, allowing only three distinct "stanzas" ("<message/>", "<presence/>", and "<iq/>"), using payloads defined by the various extension protocols maintained by the XMPP standards foundation.

3.6. Capabilities

SACM components interact with each other based on their capacity to perform specific actions. In advertising its capabilities, a SACM component indicates its competence to understand message payloads, perform any payload translation or normalization, and act upon that message. For example, an Orchestration component receives a message to initiate posture attribute collection. The Orchestrator may then normalize those instructions to a particular collection system’s serialization. The normalized instructions are then published to the Integration Service, notifying the appropriate subscribers.

Capabilities are described using Uniform Resource Names (URNs), which will be maintained and enhanced via IANA tables (IANA Considerations). Using topic-centric categorization of message payloads, capability URNs SHOULD be associated with Integration Service topics to which publishers, subscribers, and service handlers, will interact. Topic naming conventions are considered implementation details and are not considered for standardization. Given a payload-centric categorization of message payloads, capability URNs SHOULD be used as the identifying token, tag, or namespace in order to distinguish specific payloads.

3.7. Interaction Categories

Two categories of interactions SHOULD be supported by the Integration Service: broadcast and directed. Broadcast interactions are asynchronous by default, and directed interactions may be invoked either synchronously or asynchronously.

3.7.1. Broadcast

A broadcast interaction, commonly referred to as publish/subscribe, allows for a wider distribution of a message payload. When a payload is published to the Integration Service, all subscribers to that payload are alerted and may consume the message payload. This category of interaction can also be described as a "unicast" interaction when only a single subscriber exists. An example of a broadcast interaction could be to publish Linux OVAL objects to a posture collection topic. Subscribing consumers receive the notification, and proceed to collect endpoint configuration posture based on the supplied message payload.

3.7.2. Directed

The intent of a directed interaction is to enable point-to-point communications between a producer and consumer, through the standard interfaces provided by the Integration Service. The provider component indicates which consumer is intended to receive the payload, and the Integration Service routes the payload directly to that consumer. Two "styles" of directed interaction exist, differing only by the response from the consumer.

3.7.2.1. Synchronous

Synchronous, request/response style interaction requires that the requesting component block and wait for the receiving component to respond, or to time out when that response is delayed past a given time threshold. A synchronous interaction example may be querying a CMDB for posture attribute information in order to perform an evaluation.

3.7.2.2. Asynchronous

An asynchronous interaction involves the payload producer directing the message to a consumer, but not blocking or waiting for an immediate response. This style of interaction allows the producer to continue on to other activities without the need to wait for responses. This style is particularly useful when the interaction payload invokes a potentially long-running task, such as data collection, report generation, or policy evaluation. The receiving component may reply later via callbacks or further interactions, but it is not mandatory.

4. SACM Role-based Architecture

Within the cooperative SACM ecosystem, a number of roles act in coordination to provide relevant policy/guidance, perform data collection, storage, evaluation, and support downstream analytics and reporting.

```
 +-------------------------------------------+
 |                 Manager                   |
 +-------------------^-----------------------+
                     |
 +-----------------+ |  +--------------------+
 | Orchestrator(s) | |  |  Repository(-ies)  |
 +---------^-------+ |  +----------^---------+
           |         |             |                +--------------------+
           |         |             |                |  Downstream Uses   |
           |         |             |                | +----------------+ |
 +---------v---------v-------------v---------+      | |   Analytics    | |
 |             Integration Service           <------> +----------------+ |
 +-----------^--------------------------^----+      | +----------------+ |
             |                          |           | |   Reporting    | |
             |                          |           | +----------------+ |
 +-----------v-------------------+      |           +--------------------+
 |  Collection Sub-Architecture  |      |
 +-------------------------------+      |
                                        |
                        +---------------v---------------+
                        |  Evaluation Sub-Architecture  |
                        +-------------------------------+
```

Figure 2: Notional Role-based Architecture

As shown in Figure 2, the SACM role-based architecture consists of some basic SACM Components communicating using an integration service. The integration service is expected to maximally align with the requirements described in [RFC8248], which means that the integration service will support brokered (i.e. point-to-point) and proxied data exchange.

4.1. Architectural Roles/Components

This document suggests a variety of players in a cooperative ecosystem; known as SACM Components. SACM Components may be composed of other SACM Components, and each SACM Component plays one, or more, of several roles relevant to the ecosystem. Roles may act as providers of information, consumers of information, or both provider and consumer. Figure 2 depicts a number of SACM components which are architecturally significant and therefore warrant discussion and clarification. Each role depicted in Figure 2 represents the interface to the component(s) fulfilling that role, not necessarily any specific implementation. For example, the "Repository" figure represents the interface to persistent storage, and not any particular persistent storage mechanism.

4.1.1. Manager

The Manager acts as the control plane for the SACM ecosystem; a sort of high level component capable of coordinating the actions, notifications, and events between components. The manager controls the administrative interfaces with the various components of the ecosystem, acting as the central point to which all other components will register and advertise their capabilities. It is the responsibility of the manager to control a component’s access to the ecosystem, maintain an inventory of components attached to the ecosystem, and to initiate the various workflows involved in the collection and/or evaluation of posture attributes.

The manager should maintain the master set of capabilities that can be supported within the ecosystem. These are the various collection, evaluation, and persistence capabilities with which components may register. The manager MAY be responsible for assigning topics for each of the capabilities that are supported, as registering components subsequently subscribe to, or configure service handlers for, those topics.

The manager may act as the user interface to the ecosystem, providing user dashboards, inventories, component management, or operational controls within the boundary of responsibility.

4.1.2. Orchestrator(s)

Orchestration components provide the manager with resources for delegating work across the SACM ecosystem. Orchestrators are responsible for receiving messages from the manager, e.g. posture attribute collection instructions, and routing those messages to the appropriate "actions". For example, an orchestrator may support the capability of translating posture collection instructions using the

Open Vulnerability and Assessment Language (OVAL) and providing those instructions to OVAL collectors. An orchestrator may support the capability of initiating policy evaluation. Where the Manager is configured to ask a particular set of questions, those questions are delegated to Orchestrators, who are then capable of asking those questions using specific dialects.

4.1.3. Repositories

Figure 2 only includes a single reference to "Repository(-ies)", but in practice, a number of separate data repositories may exist, including posture attribute repositories, policy repositories, local vulnerability definition data repositories, and state assessment results repositories. The diagrammed notion of a repository within the SACM context represents an interface in which payloads are provided (based on the capabilities of the producer), normalized, and persisted.

These data repositories may exist separately or together in a single representation, and the design of these repositories may be as distinct as their intended purpose, such as the use of relational database management systems (RDBMS), filesystem-based storage, or graph/map implementations. Each implementation of a SACM repository should focus on the relationships between data elements and implement the SACM information and data model(s).

4.1.4. Integration Service

If each SACM component represents a set of capabilities, then the Integration Service represents the "fabric" by which SACM components are woven together. The Integration Service acts as a message broker, combining a set of common message categories and infrastructure to allow SACM components to communicate using a shared set of interfaces. The Integration Service’s brokering capabilities enable the exchange of various information payloads, orchestration of component capabilities, message routing and reliable delivery. The Integration Service minimizes the dependencies from one system to another through the loose coupling of applications through messaging. SACM components will "attach" to the Integration Service either through native support for the integration implementation, or through the use of "adapters" which provide a proxied attachment.

The Integration Service should provide mechanisms for both synchronous and asynchronous request/response-style messaging, and a publish/subscribe mechanism to implement an event-based architecture. It is the responsibility of the Integration Service to coordinate and manage the sending and receiving of messages. The Integration Service should allow components to directly connect and produce or consume messages, or connect via message translators which can act as a proxy, transforming messages from a component format to one implementing a SACM data model.

The Integration Service MUST provide routing capabilities for payloads between producers and consumers. The Integration Service MAY provide further capabilities within the payload delivery pipeline. Examples of these capabilities include, but are not limited to, intermediate processing, message transformation, type conversion, validation, or other enterprise integration patterns.

4.2. Downstream Uses

As depicted by Figure 2, a number of downstream uses exist in the cooperative ecosystem. Each notional SACM component represents distinct sub-architectures which will exchange information via the integration services, using interactions described in this draft.

4.2.1. Reporting

The Reporting component represents capabilities outside of the SACM architecture scope dealing with the query and retrieval of collected posture attribute information, evaluation results, etc. in various display formats that are useful to a wide range of stakeholders.

4.2.2. Analytics

The Analytics component represents capabilities outside of the SACM architecture scope dealing with the discovery, interpretation, and communication of any meaningful patterns of data in order to inform effective decision making within the organization.

4.3. Sub-Architectures

Figure 2 shows two components representing sub-architectural roles involved in a cooperative ecosystem of SACM components for the purpose of posture assessment: Collection and Evaluation.

4.3.1. Collection Sub-Architecture

The Collection sub-architecture is, in a SACM context, the mechanism by which posture attributes are collected from applicable endpoints and persisted to a repository, such as a configuration management database (CMDB). Control plane functions initiated by the Manager will coordinate the necessary orchestration components, who will choreograph endpoint data collection via defined interactions, using the Integration Service as a message broker. Instructions to perform endpoint data collection are directed to a Posture Collection Service capable of performing collection activities utilizing any number of protocols, such as SNMP, NETCONF/RESTCONF, SCAP, SSH, WinRM, packet capture, or host-based. Instructions are orchestrated with the appropriate Posture Collection Services using serializations supported according to the collector’s capabilities.

```
     +----------------------------------------------------------+
     |                       Manager                            |
     +-----------+----------------------------------------------+
                 |
             Orchestrate
             Collection
                 |
     +-----------v-------------+ +------------------------------+
     |      Orchestrator(s)    | | Posture Attribute Repository |
     +-----------+-------------+ +--------------^---------------+
                 |                              |
              Perform                           |
             Collection                  Collected Data
                 |                              ^
                 |                              |
     +-----------v------------------------------+---------------+
     |                    Integration Service                   |
     +----+------------------^------------------------------^---+
          |                  |           |                  |
          v                  +           v                  |
       Perform           Collected    Perform           Collected
      Collection           Data      Collection           Data
          |                  ^           |                  ^
          |                  |           |                  |
     +----v-----------------------+ +------------------------------+
     | Posture Collection Service | |    |     Endpoint     |      |
     +---^------------------------+ | +--v------------------+----+ |
         |                   |      | |Posture Collection Service| |
         |                   |      | +--------------------------+ |
       Events             Queries   +------------------------------+
         ^                   |          (PCS resides on Endpoint)
         |                   |
     +---+-------------------v----+
     |          Endpoint          |
     +----------------------------+
   (PCS does not reside on Endpoint)
```

Figure 3: Decomposed Collection Sub-Architecture

4.3.1.1. Posture Collection Service

The Posture Collection Service (PCS) is a SACM component responsible for the collection of posture attributes from an endpoint or set of endpoints. A single PCS MAY be responsible for management of posture attribute collection from many endpoints. The PCS will interact with the Integration Service to receive collection instructions, and to provide collected posture attributes for persistence to one or more Posture Attribute Repositories. Collection instructions may be supplied in a variety of forms, including subscription to a publish/subscribe topic to which the Integration Service has published instructions, or via request/response-style messaging (either synchronous or asynchronous).

Four classifications of posture collections MAY be supported.

4.3.1.1.1. Ad-Hoc

Ad-Hoc collection is defined as a single colletion of posture attributes, collected at a particular time. An example of ad-hoc collection is the single collection of a specific registry key.

4.3.1.1.2. Continuous/Scheduled

Continuous/Scheduled collection is defined as the ongoing, periodic collection of posture attributes. An example of scheduled collection is the collection of a specific registry key value every day at a given time.

4.3.1.1.3. Observational

This classification of collection is triggered by the observation, external to an endpoint, of information asserting posture attribute values for that endpoint. An example of observational collection is examination of netflow data for particular packet captures and/or specific information within those captures.

4.3.1.1.4. Event-based

Event-based collection may be triggered either internally or externally to the endpoint. Internal event-based collection is triggered when a posture attribute of interest is added, removed, or modified on an endpoint. This modification indicates a change in the current state of the endpoint, potentially affecting its adherence to some defined policy. Modification of the endpoint’s minimum password length is an example of an attribute change which could trigger collection.

External event-based collection can be described as a collector being subscribed to an external source of information, receiving events from that external source on a periodic or continuous basis. An example of event-based collection is subscription to YANG Push notifications.

4.3.1.2. Endpoint

Building upon [I-D.ietf-sacm-terminology], the SACM Collection Sub-Architecture augments the definition of an Endpoint as a component within an organization’s management domain from which a Posture Collection Service will collect relevant posture attributes.

4.3.1.3. Posture Attribute Repository

The Posture Attribute Repository is a SACM component responsible for the persistent storage of posture attributes collected via interactions between the Posture Collection Service and Endpoints.

4.3.1.4. Posture Collection Workflow

Posture collection may be triggered from a number of components, but commonly begin either via event-based triggering on an endpoint or through manual orchestration, both illustrated in Figure 3 above. Once orchestration has provided the directive to perform collection, posture collection services consume the directives. Posture collection is invoked for those endpoints overseen by the respective posture collection services. Collected data is then provided to the Integration Service, with a directive to store that information in an appropriate repository.

4.3.2. Evaluation Sub-Architecture

The Evaluation Sub-Architecture, in the SACM context, is the mechanism by which policy, expressed in the form of expected state, is compared with collected posture attributes to yield an evaluation result, that result being contextually dependent on the policy being evaluated.

```
+------------------+
|     Manager      |
+-------+----------+
        |
   Orchestrate        +------------------+
    Evaluation        |    Collection    |    +-------------------------------+
        |             | Sub+Architecture |    | Evaluation Results Repository |
 +------v----------+  +--------^---------+    +-----------------^-------------+
 | Orchestrator(s) |           |                                |
 +------+----------+     (Potentially)                          |
        |                   Perform                 Store Evaluation Results
     Perform               Collection                           |
    Evaluation                 |                                |
        |                      |                                |
 +------v----------------------v--------------------------------+-------------+
 |                             Integration Service                            |
 +--------^----------------------^-----------------------^--------------------+
          |                      |                       |
          |                      |                       |
          |               Retrieve Posture            Perform
   Retrieve Policy           Attributes              Evaluation
          |                      |                       |
          |                      |                       |
   +------v-----+          +-----v------+       +--------v-------------------+
   |   Policy   |          |  Posture   |       | Posture Evaluation Service |
   | Repository |          | Attribute  |       +----------------------------+
   +------------+          | Repository |
                           +------------+
```

Figure 4: Decomposed Evaluation Sub-Architecture

4.3.2.1. Posture Evaluation Service

The Posture Evaluation Service (PES) represents the SACM component responsible for coordinating the policy to be evaluated and the collected posture attributes relevant to that policy, as well as the comparison engine responsible for correctly determining compliance with the expected state.

4.3.2.2. Policy Repository

The Policy Repository represents a persistent storage mechanism for the policy to be assessed against collected posture attributes to determine if an endpoint meets the desired expected state. Examples of information contained in a Policy Repository would be Vulnerability Definition Data or configuration recommendations as part of a CIS Benchmark or DISA STIG.

4.3.2.3. Evaluation Results Repository

The Evaluation Results Repository persists the information representing the results of a particular posture assessment, indicating those posture attributes collected from various endpoints which either meet or do not meet the expected state defined by the assessed policy. Consideration should be made for the context of individual results. For example, meeting the expected state for a configuration attribute indicates a correct configuration of the endpoint, whereas meeting an expected state for a vulnerable software version indicates an incorrect configuration.

4.3.2.4. Posture Evaluation Workflow

Posture evaluation is orchestrated through the Integration Service to the appropriate Posture Evaluation Service (PES). The PES will, using interactions defined by the applicable taxonomy, query both the Posture Attribute Repository and the Policy Repository to obtain relevant state data for comparison. If necessary, the PES may be required to invoke further posture collection. Once all relevant posture information has been collected, it is compared to expected state based on applicable policy. Comparison results are then persisted to an evaluation results repository for further downstream use and analysis.

5. Ecosystem Interactions

Ecosystem interactions describe the various functions between SACM components, including manager requirements, the onboarding of components, capability advertisement, administrative actions, and status updates, among others. The Manager component acts as the administrative "lead" for the SACM ecosystem, and must maintain records of registered components, manage capabilities, and more.

5.1. Manager

The Manager, being a specialized role in the architecture, enables the onboarding and capability management of the various SACM component roles. The Manager must support the set of capabilities needed to operate the SACM ecosystem.

With this in mind, the Manager must first authenticate to the Integration Service. Once authentication has succeeded, the Manager MUST establish a service handler capable of performing SACM component registration/onboarding activities (Component Registration Operation). The Manager MUST also establish a subscription to an ecosystem-wide status notification mechanism, in order to receive published status updates from other SACM components.

The following requirements exist for the Manager to establish service handlers supporting the component registration taxonomy (Component Registration Operation):

* The Manager MUST enable the capability to receive onboarding requests,

* The Manager MUST have the capability to generate, manage, and persist unique identifiers for all registered components,

* The Manager MUST maintain the relationships between capabilities and payload categorizations (such as topic names or specific payload identifiers),

* The Manager MUST have the capability to inventory and manage its "roster" (the list of registered components),

* The Manager MUST have the capability to manage its roster's advertised capabilities, including those endpoints to which those capabilities apply.

* In addition to supporting component registration, the Manager is responsible for many of the operational functions of the architecture, including initiating collection or evaluation, queries for repository data, or the assembly of information for downstream use.

* The Manager MUST support making directed requests to registered components over the component's administrative interface. Administrative interface functions are described by their taxonomy, below.

* The Manager MUST support each of the interaction categories as described above.

5.2. Component Registration

Component registration describes how an individual component becomes part of the SACM ecosystem; authenticating to the Integration Service, registering and establishing its administrative interface with, the Manager.

The component onboarding workflow involves multiple steps:

* The component first authenticates to the Integration Service.

- The component initiates registration with the Manager, per the component registration operation (Component Registration Operation).

- The component handles the response from the Manager to configure a service handler allowing the component to receive directed messages over the administrative interface with the Manager.

5.3. Administrative Interface

The administrative interface represents a direct communication channel between the Manager and any registered Component. This interface allows the Manager to make directed requests to a component in order to perform specific actions.

5.3.1. Capability Advertisement Handshake

Capability Advertisement is the mechanism by which components initially indicate their capabilities to the Manager. This handshake is completed using the administrative interface with the Manager. It becomes the Manager’s responsibility to persist component/capability relationships, and to provide the component the information necessary to receive and process message payloads specific to the supported capabilities.

5.3.2. Health Check

The administrative "health check" is a mechanism by which the Manager queries for the "liveness" of its roster of components, and to possibly alert users or other systems when components are no longer present. The Manager MAY enable a periodic message to each component to determine if that component is still listening to the Administrative Interface. The Health Check interaction MAY include a request for "Capability Refresh", to reinitiate the Capability Advertisement Handshake. This interaction is similar to the "Heartbeat" interaction, but is initiated by the Manager.

5.3.3. Heartbeat

The administrative "heartbeat" is a mechanism by which a Component indicates to the Manager that the Component remains connected to the ecosystem. The Heartbeat differs from the Health Check interaction in that the Component initiates the interaction, and that no response from the Manager is required.

5.3.4. Capability-specific Requests

Any number of capability-specific requests can be enabled through the administrative interface that allow the Manager to direct actions to be performed by a specific component. Utilizing the interface from a component to the Manager, this interface can be used to indicate a component has come back online, or to provide an updated capability advertisement, potentially resulting in updates to subscriptions or service handlers.

5.4. Status Notifications

A generic status notifications mechanism SHOULD be configured to which (a) the Manager is subscribed, and (b) all onboarded components can publish. Status notifications may be used by the Manager to update user interfaces, to provide notification of the start, finish, success or failure of ecosystem operations, or as events to trigger subsequent activities.

5.5. Component Interactions

Component interactions describe functionality between components relating to collection, evaluation, or other downstream processes. The following component interactions begin with the Manager providing a set of instructions to an Orchestrator or set of Orchestrators that have registered with the SACM ecosystem indicating the appropriate capabilities, such as collection or evaluation. Subscribing Orchestrator(s) MAY translate, manipulate, filter, augment, or otherwise transform the Manager’s instructions into content supported through the Orchestrator’s capabilities.

5.5.1. Initiate Ad-Hoc Collection

The Orchestrator supplies a payload of collection instructions to a Posture Collection Service either through direct or broadcast mechanisms. The receiving PCS components perform the required collection based on their capabilities. Each PCS then forms a payload of collected posture attributes (including endpoint identifying information) and provides that payload to the Posture Attribute Repository interface, for persistence.

5.5.2. Coordinate Periodic Collection

Similar to ad-hoc collection, the Orchestrator supplies a payload of collection instructions similar to those of ad-hoc collection. Additional information elements containing collection identification and periodicity are included.

5.5.2.1. Schedule Periodic Collection

To enable operations on periodic collection, the scheduling payload MUST include both a unique identifier for the set of collection instructions, as well as a periodicity expression to enable the collection schedule. An optional "immediate collection" flag will indicate to the collection component that, upon receipt of the collection instructions, a collection will automatically be initiated prior to engagement of the scheduled collection.

5.5.2.2. Cancel Periodic Collection

The Orchestrator disables the periodic collection of posture attributes by supplying collector(s) the unique identifier of previously scheduled collection instructions. An optional "final collection" flag will indicate to the collection component that, upon receipt of the cancellation instructions, a final ad-hoc collection is to take place.

5.5.3. Coordinate Observational/Event-based Collection

In these scenarios, the Posture Collection Service acts as the "observer". Interactions with the observer could specify a time period of observation and potentially information intended to filter observed posture attributes to aid the PCS in determining those attributes that are applicable for collection and persistence to the Posture Attribute Repository.

5.5.3.1. Initiate Observational/Event-based Collection

The Orchestrator supplies a payload of instructions to a topic or set of topics to which Posture Collection Services (observers) are subscribed. This payload could include specific instructions based on the observer’s capabilities to determine specific posture attributes to observe and collect.

5.5.3.2. Cancel Observational/Event-based Collection

The Orchestrator supplies a payload of instructions to a topic or set of topics to which Posture Collection Services are subscribed. The receiving PCS components cancel the identified observational/event-based collection executing on those PCS components.

5.5.4. Persist Collected Posture Attributes

Following successful collection, Posture Collection Services (PCS) will supply the payload of collected posture attributes to the interface(s) supporting the persistent storage of those attributes to the Posture Attribute Repository. Information in this payload should include identifying information of the computing resource(s) for which attributes were collected.

5.5.5. Initiate Ad-Hoc Evaluation

The Orchestrator supplies a payload of evaluation instructions to a Posture Evaluation Services (PES) either through direct or broadcast mechanisms. The receiving PES components perform the required evaluation based on their capabilities. The PES generates a payload of posture evaluation results and publishes that payload to the Evaluation Results Repository interface, for persistence.

5.5.6. Coordinate Periodic Evaluation

Similar to ad-hoc evaluation, the Orchestrator supplies a payload of evaluation instructions similar to those of ad-hoc evaluation. Additional information elements containing evaluation identification and periodicity are included.

5.5.6.1. Schedule Periodic Evaluation

To enable operations on periodic evaluation, the scheduling payload MUST include both a unique identifier for the set of evaluation instructions, as well as a periodicity expression to enable the evaluation schedule. An optional "immediate evaluation" flag will indicate to the Posture Evaluation Service (PES) that, upon receipt of the evaluation instructions, an evaluation will automatically be initiated prior to engagement of the scheduled evaluation.

5.5.6.2. Cancel Periodic Evaluation

The Orchestrator disables the periodic evaluation of posture attributes by supplying Posture Evaluation Service(s) the unique identifier of previously scheduled evaluation instructions. An optional "final evaluation" flag will indicate to the PES that, upon receipt of the cancellation instructions, a final ad-hoc evaluation is to take place.

5.5.7. Coordinate Change-based Evaluation

A more fine-grained approach to periodic evaluation may be enabled through the triggering of Posture Evaluation based on changes to posture attribute values at the time of their collection and persistence to the Posture Attribute Repository.

5.5.7.1. Identify Attributes

The Orchestrator enables change-based evaluation through a payload published to Posture Attribute Repository component(s). This payload includes appropriate information elements describing the posture attributes on which changes in value will trigger posture evaluation.

5.5.7.2. Cancel Change-based Evaluation

An Orchestrator may disable change-based evaluation through a payload published to Posture Attribute Repository component(s), including those information elements necessary to identify those posture attributes for which change-based evaluation no longer applies.

5.5.8. Queries

Queries should allow for a "freshness" time period, allowing the requesting entity to determine if/when posture attributes must be re-collected prior to performing evaluation. This freshness time period can be "zeroed out" for the purpose of automatically triggering re-collection regardless of the most recent collection.

6. Operations

The following sections describe a number of operations required to enable a cooperative ecosystem of posture attribute collection and evaluation functions.

6.1. Component Registration

Component registration describes how an individual component becomes part of the SACM ecosystem; registering with the Manager, and establishing the administrative interface.

* Interaction Type: Directed (Request/Response)

* Source Component: Any component wishing to join the ecosystem, such as Posture Collection Services, Repository Interfaces, Posture Evaluation Services and more.

* Target Component(s): Manager

6.1.1. Request Payload

When a component onboards with the ecosystem, it must identify itself to the Manager, using either descriptive information or an already-existing component unique identifier.

```
component-registration-request:
  {:component-identification:}

component-identification:
  component-unique-identifier (if re-establishing communication)
    #-OR-#
  component-type {:component-type:}
  component-name
  component-description (optional)

component-type:
  enumeration:
    - posture-collection-service
    - posture-evaluation-service
    - repository-interface
    - orchestrator
    - others?
```

When registering for the first time, the component will send identifying information including the component type and a name. If the component is re-establishing communications, for example after a restart of the component or deployment of a new version, the component only needs to supply its previously generated (and persisted) [component-unique-identifier].

6.1.2. Request Processing

When the Manager receives the component’s request for onboarding, it will:

* Generate a unique identifier, "[component-unique-identifier]", for the onboarding component,

* Persist identifying information, including the "[component-unique-identifier]" to its component inventory, enabling an up-to-date roster of components being managed,

* Establish the administrative interface to the onboarded component by enabling a service handler to listen for directed messages from the component.

6.1.3. Response Payload

The Manager will respond to the component with a payload including the component's unique identifier. At this point, the Manager is aware of the component's existence in the ecosystem, and the component can self-identify by virtue of receiving its unique identifier.

```
component-registration-response:
  component-unique-identifier: [component-unique-identifier]
```

6.1.4. Response Processing

Successful receipt of the Manager's response, including the "[component-unique-identifier]", indicates the component is onboarded to the ecosystem. Using the response payload, the component can then establish it's end of the administrative interface with the Manager. The component must then persist it's unique identifier for use when re-establishing communication with the Manager after failure recovery or restart.

6.2. Administrative Interface

A number of functions may take place which, instead of being published to multiple subscribers, may require direct interaction between the Manager and a registered component (and vice-versa). During component onboarding, this direct channel, known as the Administrative Interface, is established first by the Manager and subsequently complemented by the component onboarding the SACM ecosystem. Three operations are defined for the administrative interface, but any number of application or capability-specific operations MAY be enabled using the directed messaging provided by this interface.

6.2.1. Capability Advertisement Handshake

Capability advertisement represents the ability of any registered component to inform the Manager of that component's capacity for performing certain operations. For example, a Posture Collection Service component may advertise its capability to perform collection using a particular collection system/serialization. This capability advertisement is important for the Manager to persist in order for the Manager to correctly classify components registered within the SACM ecosystem, and therefore provide the ability to publish messages to components in accordance with their capabilities.

* Interaction Type: Directed (Request/Response)

- Source Component: Any registered component, such as Posture Collection Services, Repository Interfaces, Posture Evaluation Services and more.

- Target Component(s): Manager

6.2.1.1. Request Payload

The component's capability advertisement request payload will include a list of "Capability URNs" (TBD IANA SECTION) that represent it's supported operational capabilities.

```
capability-advertisement:
  capabilities:
    capability-urn: [urn]
    capability-urn: [urn]
    capability-urn: [urn]
    ...
```

6.2.1.2. Request Processing

Upon receipt of the component's capability advertisement, the Manager SHOULD:

- Persist the component's capabilities to the Manager's inventory

- Coordinate, based on the supplied capabilities, the service handlers (for directed messages) and/or event listeners (for broadcast messages) to which the component should support.

6.2.1.3. Response Payload

The response payload delivered to the component should include the appropriate service handling/event listening information required for the component to handle further interactions based on each advertised capability. If a capability was not registered successfully, appropriate error messages SHOULD be supplied to inform the component of the failure(s).

```
capability-advertisement-response:
  capabilities:
    capability:
      capability-urn: [urn]
      registration-status: (success | failure)
      service-handler-or-event-listener: [info]
      messages: [messages]
    capability:
      capability-urn: [urn]
      registration-status: (success | failure)
      service-handler-or-event-listener: [info]
      messages: [messages]
```

6.2.1.4. Response Processing

Once the component has received the response to its capability advertisement, it should configure the capability-specific service handler(s) or event listener(s). Once these handlers/listeners have been configured, the component is considered fully onboarded to the SACM ecosystem.

6.2.2. Health Check

As time passes, it is important that the Manager maintains knowledge of all registered component's current operational status. The health check operation describes the efforts taken by the Manager to maintain the most up-to-date inventory of it's component roster, and to potentially trigger events to users or outside systems (e.g. a SIEM or SOAR) indicating unavailable components.

* Interaction Type: Directed (Request/Response)

* Source Component: Manager

* Target Component(s): Any registered component, such as Posture Collection Services, Repository Interfaces, Posture Evaluation Services and more.

6.2.2.1. Request Payload

The request for the health check is a simple "ping".

```
health-check-request:
  action: ping
```

6.2.2.2. Request Processing

When the target component receives the health check request, the target component need only respond that it is operational and connected to the integration service. This is a simple "Hello component, are you listening? Yes, I am" interaction. The health check request from the Manager should be made with an appropriately small timeout indicator; only an operational component will be able to respond to the request, so if that component is offline and cannot respond, the Manager should not be kept waiting for an extended amount of time.

6.2.2.3. Response Payload

When responding to the health check request, the response payload will simply indicate success: ˜˜˜˜˜˜ health-check-response: response: success ˜˜˜˜˜˜

6.2.2.4. Response Processing

Upon receipt of the "health-check-response" payload, the Manager will update its inventory of currently operational components with the timestamp of the receipt. Manager implementations may raise alerts, inform users, or take other actions when health checks are unsuccessful, at their discretion.

6.2.3. Heartbeat

As time passes and SACM ecosystem components which have previously registered are brought offline (perhaps for maintenance or redeployment) and back online, it is important that registered components maintain administrative contact with the Manager. The heartbeat operation describes the efforts taken by a registered component to determine the status of contact with the Manager, and to take appropriate action if such contact cannot be made.

* Interaction Type: Directed (Request/Response)

* Source Component: Any registered component, such as Posture Collection Services, Repository Interfaces, Posture Evaluation Services and more.

* Target Component(s): Manager

6.2.3.1. Request Payload

The request payload simply defines the hearbeat action:

```
heartbeat-request:
  action: pulse
```

6.2.3.2. Request Processing

When the Manager receives the heartbeat request, it need only respond that it is operational and connected to the integration service. This is a simple "Hello Manager, are you listening? Yes, I am" interaction. The heartbeat request from the component should be made with an appropriately small timeout indicator; only an operational Manager will be able to respond to the request, so if it is offline and cannot respond, the component should not be kept waiting for an extended amount of time.

6.2.3.3. Response Payload

When responding to the heartbeat request, the response payload will simply indicate success: ˜˜˜˜˜˜ heartbeat-response: response: success ˜˜˜˜˜˜

6.2.3.4. Response Processing

Upon receipt of the "heartbeat-response" payload, the component may reset its heartbeat timer and continue normal operations, awaiting incoming message payloads. Component implementations may raise alerts, inform users, or take other actions when heartbeat requests are unsuccessful (potentially indicating a downed Manager), at their discretion.

6.3. Status Notification

From time to time during the performance of any given operation, a component may need to supply status information to the Manager (or any other concerned component), for use in display to users, or to trigger other events within the SACM ecosystem. The status notification operation is designed to allow any component to broadcast status message payloads to any subscribers with the need to know. For example, a collection component could broadcast to the Manager that it has initiated collection, subsequent collection progress updates, and finally completion or error conditions.

* Interaction Type: Broadcast (Publish/Subscribe)

* Source Component: Any registered component, such as Posture Collection Services, Repository Interfaces, Posture Evaluation Services and more.

- Target Component(s): Typically the Manager, but any registered component may subscribe to status notifications.

6.3.1. Request Payload

At a minimum, the payload broadcast for a status notification MUST include the status message and the publishing component’s "component-unique-identifier". Further identifying information, such as status codes, operation indicators, etc., MAY be provided by implementing components.

```
status-notification:
  publisher: [component-unique-identifier]
  message: [message]
  [additional information]
```

6.3.2. Request Processing

When subscribers are notified of the status message, respective components may act upon them in component/application-specific ways, including persisting those messages to repositories, forwarding to log aggregation tools, displaying on user interfaces, and so on. Potential for use of component status notifications is only limited by application implementations.

6.3.3. Response Payload

N/A

6.3.4. Response Processing

N/A

6.4. Initiate Ad-Hoc Collection

The Ad-hoc collection workflow MAY be initiated by the Manager, via user interaction, or through a Posture Evaluation Service, and represents a single, point-in-time operation to collect posture attributes from applicable endpoints. The SACM Producer initiates a message payload, either through directed channels (such as the administrative interface) or through broadcast notifications to multiple subscribers, to Orchestrator components. Orchestrators MAY manipulate the Manager’s collection instructions according to various collection capabilities, prior to providing those instructions to Posture Collection Service (PCS) components. Once collection instructions are received by the PCS, it will collect the requested posture attributes from the designated endpoints, using its advertised collection capabilities. The following diagram illustrates this workflow with the Manager as the initiating SACM Producer:

```
 +-----------+                      +------------------------------+
 |  Manager  |                      |  Posture Collection Service  |
 +-+-----^---+                      +---^----------+----------+----+
   |     |                              |          |          |
 1.|    (S)                           4.|         (S)       5.|
   |     |                              |          |          |
   |     |                              |          |          |
 +-v-----+-----------------------------------------v----------v----+
 |                       Integration Service                       |
 +-+-----^------^----------------------------------^----------+----+
   |     |      |                                  |          |
 2.|    (S)     |3.                               (S)         |6.
   |     |      |                                  |          |
   |     |      |                                  |          |
 +-v-----+------+-+               +----------------+----------v----+
 |  Orchestrator  |               |  Posture Attribute Repository  |
 +----------------+               +--------------------------------+
```

1. The Manager initiates a request to one or more Orchestrators to perform collection,

2. The Orchestrator receives collection instructions and potentially manipulates them according to one or more collection capabilities,

3. The Orchestrator publishes a notification to subscribed Posture Collection Service components, indicating the posture attributes to be collected,

4. The Posture Collection Service receives the collection instructions and performs the actual collection of posture attributes from an endpoint or endpoints.

5. The Posture Collection Service publishes a notification(s) containing the collected posture attributes to be persisted to the Posture Attribute Repository,

6. The Posture Attribute Repository persists the collected posture attributes, potentially performing normalization of the data as part of its process.

Interactions labeled (S) indicate the capability of each component to publish status notifications, subscribed to by the Manager.

6.4.1. SACM Producer to Orchestrator

The Ad-hoc collection workflow MAY be initiated by a number of SACM components, such as the Manager, a Posture Evaluation Service, or other events outside the scope of this document.

* Interaction Type: Directed (Request/Response) or Broadcast (Publish/Subscribe)

* Source Component: Various

* Target Component(s): Orchestrator

6.4.1.1. Request Payload

A request to orchestrate posture attribute collection MUST include enough information to describe those attributes being collected, and MAY include endpoint targeting information.

collection-instructions:
TBD

6.4.1.2. Request Processing

When the Orchestrator receives the collection instructions, it may be required to manipulate them according to the capabilities it’s collector(s) support. For example, generic collection instructions could be transformed to the appropriate OVAL serialization for collection via OVAL-compliant collectors.

6.4.1.3. Response Payload

Orchestrators have the option to provide broadcast status update messages to indicate success, failure, or other error messages when receiving posture collection orchestration payloads.

6.4.1.4. Response Processing

N/A

6.4.2. Orchestrator to Posture Collection Service

Once the Orchestrator has received collection instructions from the initiating SACM component, and has performed any manipulation of the instructions to conform to it’s capabilities, it will provide those instructions to relevant Posture Collection Services.

- Interaction Type: Directed (Request/Response) or Broadcast (Publish/Subscribe)

- Source Component: Orchestrator

- Target Component(s): Posture Collection Service

6.4.2.1. Request Payload

The payload exchanged between the Orchestrator and it's associated Posture Collection Services will be collection instructions adhering to a data model supported by the PCS based on its advertised capabilities.

```
collection-instructions:
  TBD
```

6.4.2.2. Request Processing

Upon receipt of the payload containing collection instructions, the Posture Collection Service should parse and validate them, indicating any errors in the process. If the payload does not conform to any serialization or data model to which the PCS' capabilities correspond, status messages indicating such nonconformance SHOULD be provided to both the Orchestrator and the initiating SACM producer.

Once successfully parsed and validated, the PCS MUST perform collection of posture attributes according to the collection instructions, from any endpoint to which the PCS has access, or from the list of endpoints described in any targeting information included in the collection instructions.

6.4.2.3. Response Payload

Posture Collection Service components will respond using the generic status update mechanisms to indicate success, failure, or any errors that occur. Errors may occur parsing collection instructions, verifying them, targeting indicated endpoints, or from the act of collecting the indicated posture attributes.

6.4.2.4. Response Processing

Any messages received by components regarding the success, failure, or errors involved in the collection of posture attributes MAY be processed according to the receiving components' capabilities.

6.4.3. Posture Collection Service to Posture Attribute Repository

Upon completion of posture attribute collection, the PCS constructs the payload of collected attributes based on its advertised capabilities, e.g. OVAL system characteristics. This payload is provided to either a specific posture attribute repository via directed messages or to subscribed repository interfaces via broadcast messages.

* Interaction Type: Directed (Request/Response) or Broadcast (Publish/Subscribe)

* Source Component: Posture Collection Service

* Target Component(s): Posture Attribute Repository

6.4.3.1. Request Payload

The payload supplied by the Posture Collection Service SHOULD conform to information and data models supported by its advertised capabilities. These data models, at a minimum, SHOULD include name/value pairs for each collected attribute.

```
collection-results:
  [
    attribute-name,
    attribute-value
  ]
```

6.4.3.2. Request Processing

As the Posture Attribute Repository interface receives the payload of collected posture attributes, some data normalization MAY occur in order to persist the information most efficiently based on the persistence technology. This normalization is dependent on the implementation of the repository interface as well as the persistence technology. For example, OVAL system characteristics, an XML payload, could be normalized to a property graph representation when persisted to a Neo4j database.

6.4.3.3. Response Payload

Once the Posture Attribute Repository has received, it MAY respond to the Posture Collection Service that it has successfully received the collected posture attributes. This response would only be applicable when receiving payloads via directed requests. If payloads are received via broadcast interactions, there may not be a PCS to which a response can be sent. The Posture Attribute Repository MAY utilize the generic status update interactions to provide response messages to appropriate subscribers.

6.4.3.4. Response Processing

Any messages received by components regarding the success, failure, or errors involved in the persistence of collected posture attributes MAY be processed according to the receiving components' capabilities. For example, a generic status update message could be processed by a Manager component, correlated with the initial posture collection instructions in order to "close the loop" on the posture attribute collection workflow.

6.5. Initiate Ad-Hoc Evaluation

### Manager to Orchestrator ### Orchestrator to Evaluator ### Evaluator to Posture Evaluation Repository

7. Privacy Considerations

[TBD]

8. Security Considerations

[TBD]

9. IANA Considerations

[TBD] Some boilerplate code...

9.1. Component Types

URI: "urn:ietf:sacm:component-type" Description: The allowed enumeration of the various component types permitted to utilize the SACM ecosystem.

- Manager

- Orchestrator

- Collector

- Evaluator

- Repository Interface

- [MORE]

9.2. Component Capabilities

### Health Check A URN representing a component’s capability to initiate Health Check operations and to process any provided response payloads.

URN: "urn:ietf:sacm:capability:action:health-check"

9.2.1. Heartbeat

A URN representing a component’s capability to initiate Heartbeat operations and to process any provided response payloads.

URN: "urn:ietf:sacm:capability:action:heartbeat"

9.2.2. Status Notification (Publish)

A URN representing a component’s capability to publish status notifications.

URN: "urn:ietf:sacm:capability:publish:status-notification"

9.2.3. Status Notification (Subscribe)

A URN representing a component’s capability to subscribe to status notification events.

URN: "urn:ietf:sacm:capability:subscribe:status-notification"

10. References

10.1. Normative References

[I-D.ietf-sacm-ecp]
Haynes, D., Fitzgerald-McKay, J., and L. Lorenzin, "Endpoint Posture Collection Profile", Work in Progress, Internet-Draft, draft-ietf-sacm-ecp-05, 21 June 2019, <https://www.ietf.org/archive/id/draft-ietf-sacm-ecp-05.txt>.

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

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC8412] Schmidt, C., Haynes, D., Coffin, C., Waltermire, D., and J. Fitzgerald-McKay, "Software Inventory Message and Attributes (SWIMA) for PA-TNC", RFC 8412, DOI 10.17487/RFC8412, July 2018, <https://www.rfc-editor.org/info/rfc8412>.

[RFC8600] Cam-Winget, N., Ed., Appala, S., Pope, S., and P. Saint-Andre, "Using Extensible Messaging and Presence Protocol (XMPP) for Security Information Exchange", RFC 8600, DOI 10.17487/RFC8600, June 2019, <https://www.rfc-editor.org/info/rfc8600>.

10.2. Informative References

[CISCONTROLS] "CIS Controls v7.1", n.d., <https://www.cisecurity.org/controls>.

[draft-birkholz-sacm-yang-content] Birkholz, H. and N. Cam-Winget, "YANG subscribed notifications via SACM Statements", n.d., <https://tools.ietf.org/html/draft-birkholz-sacm-yang-content-01>.

[HACK100] "IETF 100 Hackathon - Vulnerability Scenario EPCP+XMPP", n.d., <https://www.github.com/sacmwg/vulnerability-scenario/ietf-hackathon>.

[HACK101] "IETF 101 Hackathon - Configuration Assessment XMPP", n.d., <https://www.github.com/CISecurity/Integration>.

[HACK102] "IETF 102 Hackathon - YANG Collection on Traditional Endpoints", n.d., <https://www.github.com/CISecurity/YANG>.

[HACK103] "IETF 103 Hackathon - N/A", n.d., <https://www.ietf.org/how/meetings/103/>.

[HACK104] "IETF 104 Hackathon - A simple XMPP client", n.d., <https://github.com/CISecurity/SACM-Architecture>.

[HACK105] "IETF 105 Hackathon - A more robust XMPP client including collection extensions", n.d., <https://github.com/CISecurity/SACM-Architecture>.

[HACK99] "IETF 99 Hackathon - Vulnerability Scenario EPCP", n.d., <https://www.github.com/sacmwg/vulnerability-scenario/ietf-hackathon>.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D. R., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", Work in Progress, Internet-Draft, draft-ietf-i2nsf-terminology-08, 5 July 2019, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-terminology-08.txt>.

[I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., Cam-Winget, N., and A. Montville, "Security Automation and Continuous Monitoring (SACM) Terminology", Work in Progress, Internet-Draft, draft-ietf-sacm-terminology-16, 14 December 2018, <https://www.ietf.org/archive/id/draft-ietf-sacm-terminology-16.txt>.

[MQTT] "MQTT", n.d., <https://mqtt.org/mqtt-specification/>.

[NIST800126]
Waltermire, D., Quinn, S., Booth, H., Scarfone, K., and D. Prisaca, "SP 800-126 Rev. 3 - The Technical Specification for the Security Content Automation Protocol (SCAP) - SCAP Version 1.3", February 2018, <https://csrc.nist.gov/publications/detail/sp/800-126/rev-3/final>.

[NISTIR7694]
Halbardier, A., Waltermire, D., and M. Johnson, "NISTIR 7694 Specification for Asset Reporting Format 1.1", n.d., <https://csrc.nist.gov/publications/detail/nistir/7694/final>.

[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>.

[RFC5023] Gregorio, J., Ed. and B. de hOra, Ed., "The Atom Publishing Protocol", RFC 5023, DOI 10.17487/RFC5023, October 2007, <https://www.rfc-editor.org/info/rfc5023>.

[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>.

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

[RFC8248] Cam-Winget, N. and L. Lorenzin, "Security Automation and Continuous Monitoring (SACM) Requirements", RFC 8248, DOI 10.17487/RFC8248, September 2017, <https://www.rfc-editor.org/info/rfc8248>.

[RFC8322] Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange (ROLIE)", RFC 8322, DOI 10.17487/RFC8322, February 2018, <https://www.rfc-editor.org/info/rfc8322>.

[XMPPEXT] "XMPP Extensions", n.d., <https://xmpp.org/extensions/>.

Appendix A. Security Domain Workflows

This section describes three primary information security domains from which workflows may be derived: IT Asset Management, Vulnerability Management, and Configuration Management.

A.1. IT Asset Management

Information Technology asset management is easier said than done. The [CISCONTROLS] have two controls dealing with IT asset management. Control 1, Inventory and Control of Hardware Assets, states, "Actively manage (inventory, track, and correct) all hardware devices on the network so that only authorized devices are given access, and unauthorized and unmanaged devices are found and prevented from gaining access." Control 2, Inventory and Control of Software Assets, states, "Actively manage (inventory, track, and correct) all software on the network so that only authorized software is installed and can execute, and that unauthorized and unmanaged software is found and prevented from installation or execution."

In spirit, this covers all of the processing entities on your network (as opposed to things like network cables, dongles, adapters, etc.), whether physical or virtual, on-premises or in the cloud.

A.1.1. Components, Capabilities and Workflow(s)

TBD

A.1.1.1. Components

TBD

A.1.1.2. Capabilities

An IT asset management capability needs to be able to:

* Identify and catalog new assets by executing Target Endpoint Discovery Tasks

* Provide information about its managed assets, including uniquely identifying information (for that enterprise)

* Handle software and/or hardware (including virtual assets)

* Represent cloud hybrid environments

A.1.1.3. Workflow(s)

TBD

A.2. Vulnerability Management

Vulnerability management is a relatively established process. To paraphrase the [CISCONTROLS], continuous vulnerability management is the act of continuously acquiring, assessing, and taking subsequent action on new information in order to identify and remediate vulnerabilities, therefore minimizing the window of opportunity for attackers.

A vulnerability assessment (i.e. vulnerability detection) is performed in two steps:

* Endpoint information collected by the endpoint management capabilities is examined by the vulnerability management capabilities through Evaluation Tasks.

- If the data possessed by the endpoint management capabilities is insufficient, a Collection Task is triggered and the necessary data is collected from the target endpoint.

Vulnerability detection relies on the examination of different endpoint information depending on the nature of a specific vulnerability. Common endpoint information used to detect a vulnerability includes:

- A specific software version is installed on the endpoint

- File system attributes

- Specific state attributes

In some cases, the endpoint information needed to determine an endpoint’s vulnerability status will have been previously collected by the endpoint management capabilities and available in a Repository. However, in other cases, the necessary endpoint information will not be readily available in a Repository and a Collection Task will be triggered to perform collection from the target endpoint. Of course, some implementations of endpoint management capabilities may prefer to enable operators to perform this collection even when sufficient information can be provided by the endpoint management capabilities (e.g. there may be freshness requirements for information).

A.2.1. Components, Capabilities and Workflow(s)

TBD

A.2.1.1. Components

TBD

A.2.1.2. Capabilities

TBD

A.2.1.3. Workflow(s)

TBD

A.3. Configuration Management

Configuration management involves configuration assessment, which requires state assessment. The [CISCONTROLS] specify two high-level controls concerning configuration management (Control 5 for non-network devices and Control 11 for network devices). As an aside, these controls are listed separately because many enterprises have different organizations for managing network infrastructure and workload endpoints. Merging the two controls results in the following paraphrasing: Establish, implement, and actively manage (track, report on, correct) the security configuration of systems using a rigorous configuration management and change control process in order to prevent attackers from exploiting vulnerable services and settings.

Typically, an enterprise will use configuration guidance from a reputable source, and from time to time they may tailor the guidance from that source prior to adopting it as part of their enterprise standard. The enterprise standard is then provided to the appropriate configuration assessment tools and they assess endpoints and/or appropriate endpoint information.

A preferred flow follows:

- Reputable source publishes new or updated configuration guidance

- Enterprise configuration assessment capability retrieves configuration guidance from reputable source

- Optional: Configuration guidance is tailored for enterprise-specific needs

- Configuration assessment tool queries asset inventory repository to retrieve a list of affected endpoints

- Configuration assessment tool queries configuration state repository to evaluate compliance

- If information is stale or unavailable, configuration assessment tool triggers an ad hoc assessment

The SACM architecture needs to support varying deployment models to accommodate the current state of the industry, but should strongly encourage event-driven approaches to monitoring configuration.

A.3.1. Components, Capabilities and Workflow(s)

This section provides more detail about the components and capabilities required when considering the aforementioned configuration management workflow.

A.3.1.1. Components

The following is a minimal list of SACM Components required to implement the aforementioned configuration assessment workflow.

* Configuration Policy Feed: An external source of authoritative configuration recommendations.

* Configuration Policy Repository: An internal repository of enterprise standard configurations.

* Configuration Assessment Orchestrator: A component responsible for orchestrating assessments.

* Posture Attribute Collection Subsystem: A component responsible for collection of posture attributes from systems.

* Posture Attribute Repository: A component used for storing system posture attribute values.

* Configuration Assessment Evaluator: A component responsible for evaluating system posture attribute values against expected posture attribute values.

* Configuration Assessment Results Repository: A component used for storing evaluation results.

A.3.1.2. Capabilities

Per [RFC8248], solutions MUST support capability negotiation. Components implementing specific interfaces and operations (i.e. interactions) will need a method of describing their capabilities to other components participating in the ecosystem; for example, "As a component in the ecosystem, I can assess the configuration of Windows, MacOS, and AWS using OVAL".

A.3.1.3. Configuration Assessment Workflow

This section describes the components and interactions in a basic configuration assessment workflow. For simplicity, error conditions are recognized as being necessary and are not depicted. When one component messages another component, the message is expected to be handled appropriately unless there is an error condition, or other notification, messaged in return.

```
+-------------+  +----------------+  +------------------+  +------------+
| Policy Feed |  |  Orchestrator  |  |    Evaluation    |  | Evaluation |
+------+------+  +-------+--------+  | Sub-Architecture |  |   Results  |
       |                 |           +---^----------+---+  | Repository |
       |                 |               |          |      +------^-----+
       |                 |               |          |             |
     1.|               3.|             8.|        9.|         10.|
       |                 |               |          |             |
       |                 |               |          |             |
+------v-----------------v---------------+----------v-------------+-----+
|                           Integration Service                         |
+-----+----------------------------------+----------^---------+------^--+
      |                                  |          |         |      |
      |                                  |          |         |      |
    2.|                                4.|        5.|       6.|    7.|
      |                                  |          |         |      |
      |                                  |          |         |      |
+-----v------+                       +---v----------+---+  +--v------+--+
|   Policy   |                       |    Collection    |  |  Posture   |
| Repository |                       | Sub-Architecture |  | Attribute  |
+------------+                       +------------------+  | Repository |
                                                           +------------+
```

Figure 5: Configuration Assessment Component Interactions

Figure 5 depicts configuration assessment components and their interactions, which are further described below.

1. A policy feed provides a configuration assessment policy payload to the Integration Service.

2. The Policy Repository, a consumer of Policy Feed information, receives and persists the Policy Feed’s payload.

3. Orchestration component(s), either manually invoked, scheduled, or event-based, publish a payload to begin the configuration assessment process.

4. If necessary, Collection Sub-Architecture components may be invoked to collect neeeded posture attribute information.

5. If necessary, the Collection Sub-Architecture will provide collected posture attributes to the Integration Service for persistence to the Posture Attribute Repository.

6. The Posture Attribute Repository will consume a payload querying for relevant posture attribute information.

7. The Posture Attribute Repository will provide the requested information to the Integration Service, allowing further orchestration payloads requesting the Evaluation Sub-Architecture perform evaluation tasks.

8. The Evaluation Sub-Architecture consumes the evaluation payload and performs component-specific state comparison operations to produce evaluation results.

9. A payload containing evaluation results are provided by the Evaluation Sub-Architecture to the Integration Service

10. Evaluation results are consumed by/persisted to the Evaluation Results Repository

In the above flow, the payload information is expected to convey the context required by the receiving component for the action being taken under different circumstances. For example, a directed message sent from an Orchestrator to a Collection sub-architecture might be telling that Collector to watch a specific posture attribute and report only specific detected changes to the Posture Attribute Repository, or it might be telling the Collector to gather that posture attribute immediately. Such details are expected to be handled as part of that payload, not as part of the architecture described herein.

Authors' Addresses

Adam W. Montville
Center for Internet Security
31 Tech Valley Drive
East Greenbush, NY 12061
United States of America

Email: adam.montville.sdo@gmail.com

Bill Munyan
Center for Internet Security
31 Tech Valley Drive
East Greenbush, NY 12061
United States of America

Email: bill.munyan.ietf@gmail.com