SACM Working Group   A. Montville
Internet-Draft   B. Munyan
Intended status: Standards Track   CIS
Expires: September 29, 2018   March 28, 2018

Security Automation and Continuous Monitoring (SACM) Architecture
draft-mandm-sacm-architecture-01

Abstract

This memo documents an exploration of a possible Security Automation and Continuous Monitoring (SACM) architecture. This work is built upon [I-D.ietf-mile-xmpp-grid], and 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].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The purpose of this draft is to document and track the outcome of solution discovery, with the intent of eventually describing an emerged architecture. We have initially built our partial solution upon [I-D.ietf-mile-xmpp-grid] and [I-D.ietf-sacm-ecp], and believe these approaches complement each other to more completely meet the spirit of [RFC7632] and requirements found in [RFC8248].

This solution gains the most advantage by supporting a variety of collection mechanisms. In this sense, our solution ideally intends to enable a cooperative ecosystem of tools from disparate sources with minimal operator configuration. The solution described in this document seeks to accommodate these recognitions by first defining a

generic abstract architecture, then making that solution somewhat more concrete.

Keep in mind that, at this point, the draft is tracking ongoing work being performed primarily around and during IETF hackathons. The list of hackathon efforts follows:

- [HACK99]: TODO: Provide description.

- [HACK100]: TODO: Provide description.

- [HACK101]: TODO: Provide description.

1.1. Open Questions

The following is a list of open questions we still have about the path forward with this exploration:

- What are the specific components participating in a SACM Domain?

- What are the capabilities we can expect these components to contain?

  * How can we classify these capabilities?

  * How do we define an extensible capability taxonomy (perhaps using IANA tables)?

- What are the present-day workflows we expect an operational enterprise to carry out?

  * Can we prioritize these workflows in some way that helps us progress sensibly?

  * How can these workflows be improved?

  * Is it a straight path to improvement?

- Should workflows be documented in this draft or separate drafts?

- Should interfaces be documented in workflow drafts or separate drafts (or even this draft)?

1.2. 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

This draft defers to [I-D.ietf-sacm-terminology] for terms and definitions.

3. Architectural Discovery

The generic approach proposed herein recognizes the need to pull information from existing state collection mechanisms, 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.

```
   +----------+      +-------+   +------------+
   |Repository|      |Policy|   |Orchestrator|
   +----^-----+      +--^----+   +----^-------+       +----------------+
     A  |            B  |          C |               | Downstream Uses|
        |               |            |               | +-----------+  |
   +----v---------------v------------v-------+       | |Evaluations|  |
   |             Message Transfer            <-------> +-----------+  |
   +----------------^------------------------+     D | +---------+    |
                    |                                | |Analytics|    |
                    |                                | +---------+    |
            +-------v---------                       | +---------+    |
            | Transfer System |                      | |Reporting|    |
            |    Connector    |                      | +---------+    |
            +-------^---------+                      +----------------+
                    |
                    |
            +-------v-------+
            |   Collection  |
            |     System    |
            +---------------+
```

Figure 1: Notional Architecture

As shown in Figure 1, the notional SACM architecture consists of some basic SACM Components using a message transfer system to communicate. While not depicted, the message transfer system is expected to maximally align with the requirements described in [RFC8248], which means that the message transfer system will support brokered (i.e. point-to-point) and proxied data exchange.

Additionally, component-specific interfaces (i.e. such as A, B, C, and D in Figure 1) are expected to be specified logically then bound to one or more specific implementations. This should be done for each capability related to the given SACM Component.

3.1. SACM Roles

This document suggests a variety of players in a cooperative ecosystem - we call these players SACM Components. SACM Components may be composed of other SACM Components, and each SACM Component plays one of several roles relevant to the ecosystem. Generally each role is either a consumer of information or a provider of information. The "Components, Capabilities, Interfaces, and Workflows" section provides more details about SACM Components that play these types of roles.

3.2. Exploring An XMPP-based Solution

In Figure 2, we have a more detailed view of the architecture - one that fosters the development of a pluggable ecosystem of cooperative tools. Existing collection mechanisms (ECP/SWIMA included) can be brought into this architecture by specifying the interface of the collector and creating the XMPP-Grid Connector binding for that interface.

Additionally, while not directly depicted in Figure 2, this architecture does allows point-to-point interfaces. In fact, [I-D.ietf-mile-xmpp-grid] provides brokering capabilities to facilitate such point-to-point data transfers). Additionally, each of the SACM Components depicted in Figure 2 may be a provider, a consumer, or both, depending on the workflow in context.

```
  +----------+      +------+   +------------+
  |Repository|      |Policy|   |Orchestrator|
  +----^-----+      +--^---+   +----^-------+
       |               |            |
       |               |            |
  +----v---------------v------------v-----------------+     +-----------------+
  |                     XMPP-Grid+                    <-----> Downstream Uses |
  +-----^-------------^-------------^-------------^---+     +-----------------+
        |             |             |             |
        |             |             |             |
   +----v----+   +----v----+   +----v----+   +----v----+
   |XMPP-Grid|   |XMPP-Grid|   |XMPP-Grid|   |XMPP-Grid|
/~~|Connector|~~~|Connector|~~~|Connector|~~~|Connector|~~\
|  +----^----+   +----^----+   +----^----+   +----^----+  |
|       |             |             |             |       |
|  +----v----+   +----v-----+  +----v----+   +----v----+  |
|  |ECP/SWIMA|   |Datastream|  |YANG Push|   |  IPFIX  |  |
|  +---------+   +----------+  +---------+   +---------+  |
|                      Collectors                         |
\~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~/
```

Figure 2: Detailed Architecture

At this point, [I-D.ietf-mile-xmpp-grid] specifies fewer features than SACM requires, and there are other XMPP extensions (XEPs) we need to consider to meet the needs of [RFC7632] and [RFC8248]. In Figure 2 we therefore use "XMPP-Grid+" to indicate something more than [I-D.ietf-mile-xmpp-grid] alone, even though we are not yet fully confident in the exact set of XMPP-related extensions we will require. The authors propose work to extend (or modify) [I-D.ietf-mile-xmpp-grid] to include additional XEPs - possibly the following:

o Entity Capabilities (XEP-0115): May be used to express the specific capabilities that a particular client embodies.

o Form Discovery and Publishing (XEP-0346): May be used for datastream examples requiring some expression of a request followed by an expected response.

o Ad Hoc Commands (XEP-0050): May be usable for simple orchestration (i.e. "do assessment").

o File Repository and Sharing (XEP-0214): Appears to be needed for handling large amounts of data (if not fragmenting).

o Publishing Stream Initiation Requests (XEP-0137): Provides ability to stream information between two XMPP entities.

- PubSub Collection Nodes (XEP-0248): Nested topics for specialization to the leaf node level.

- Security Labels In Pub/Sub (XEP-0314): Enables tagging data with classification categories.

- PubSub Since (XEP-0312): Persists published items, which may be useful in intermittent connection scenarios

- PubSub Chaining (XEP-0253): Federation of publishing nodes enabling a publish node of one server to be a subscriber to a publishing node of another server

- Easy User Onboarding (XEP-401): Simplified client registration

4. Components, Capabilities, Interfaces, and Workflows

The SACM Architecture consists of a variety of SACM Components, and named components are intended to embody one or more specific capabilities. Interacting with these capabilities will require at least two levels of interface specification. The first is a logical interface specification, and the second is at least one binding to a specific transfer mechanism. At this point, we have been experimenting with XMPP as a transfer mechanism.

The following subsections describe some of the components, capabilities, and interfaces we may expect to see participating in a SACM Domain.

4.1. Components

The following is a list of suggested SACM Component classes and specializations.

- Repository

  - Vulnerability Information Repository

  - Asset Inventory Repository

    - Software Inventory Repository

    - Device Inventory Repository

  - Configuration Policy Repository

  - Configuration State Repository

- o Collector
  - Vulnerability State Collector
  - Asset Inventory Collector
    - Software Inventory Collector
    - Device Inventory Collector
  - Configuration State Collector
- o Evaluator
  - Vulnerability State Evaluator
  - Asset Inventory Evaluator
    - Software Inventory Evaluator
    - Device Inventory Evaluator
  - Configuration State Evaluator
- o Orchestrator
  - Vulnerability Management Orchestrator
  - Asset Management Orchestrator
    - Software Inventory Evaluator
    - Device Inventory Evaluator
  - Configuration Management Orchestrator

4.2. Capabilities

Repositories will have a need for fairly standard CRUD operations and query by attribute operations. Collector interfaces may enable ad hoc assessment (on-demand processing), state item watch actions (i.e. watch a particular item for particular change), persisting other behaviors (i.e. setting some mandatory reporting period). Evaluators may have their own set of interfaces, and an Assessor would represent both Collector and Evaluation interfaces, and may have additional concerns added to an Assessor Interface.

Not to be overlooked, whatever solution at which we arrive must, per [RFC8248], MUST support capability negotiation. While not explicitly treated here, each interface will understand specific serializations, and other component needs to express those serializations to other components.

4.3. Interfaces

Interfaces should be derived directly from identified workflows, several of which are described in this document.

4.4. (Candidate) Workflows

The workflows described in this document should be considered as candidate workflows - informational for the purpose of discovering the necessary components and specifying their interfaces.

4.4.1. Vulnerability Management

TODO: Pull in some vulnerability management scenario text.

4.4.2. Configuration Management

TODO: Describe configuration management workflow (from policy creation to implementation to routine assessment).

4.4.3. IT Asset Management

TODO: Describe some ideas surrounding the notion of managing technology assets. For example, we may consider software inventory for:

- Agent-based devices

- Non-agent based devices

- Virtual/Cloud environments (public/private) including containers

- Mobile devices

- Devices that are intermittently connected

Ideally, this would provide hardware identification as well.

5. Privacy Considerations

TODO

6. Security Considerations

TODO

7. IANA Considerations

IANA tables can probably be used to make life a little easier. We would like a place to enumerate:

- Capability/operation semantics

- SACM Component implementation identifiers

- SACM Component versions

- Associations of SACM Components (and versions) to specific Capabilities

8. References

8.1. Normative References

[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>.

8.2. Informative References

[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 ECP+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>.

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

[I-D.ietf-mile-rolie]
Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange", draft-ietf-mile-rolie-16 (work in progress), December 2017.

[I-D.ietf-mile-xmpp-grid]
Cam-Winget, N., Appala, S., Pope, S., and P. Saint-Andre, "Using XMPP for Security Information Exchange", draft-ietf-mile-xmpp-grid-05 (work in progress), February 2018.

[I-D.ietf-sacm-ecp]
Haynes, D., Fitzgerald-McKay, J., and L. Lorenzin, "Endpoint Compliance Profile", draft-ietf-sacm-ecp-01 (work in progress), January 2018.

[I-D.ietf-sacm-nea-swid-patnc]
Schmidt, C., Haynes, D., Coffin, C., Waltermire, D., and J. Fitzgerald-McKay, "Software Inventory Message and Attributes (SWIMA) for PA-TNC", draft-ietf-sacm-nea-swid-patnc-01 (work in progress), March 2017.

[I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., Cam-Winget, N., and A. Montville, "Security Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-14 (work in progress), December 2017.

[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>.

[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>.

[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>.

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

Appendix A. Mapping to RFC8248

This section provides a mapping of XMPP and XMPP Extensions to the relevant requirements from [RFC8248]. In the table below, the ID and Name columns provide the ID and Name of the requirement directly out of [RFC8248]. The Supported By column may contain one of several values:

- N/A: The requirement is not applicable to this architectural exploration

- Architecture: This architecture (possibly assuming some components) should meet the requirement

- XMPP: The set of XMPP Core specifications and the collection of applicable extensions, deployment, and operational considerations.

- XMPP-Core: The requirement is satisfied by a core XMPP feature

- XEP-nnnn: The requirement is satisfied by a numbered XMPP extension (see [XMPPEXT])

- Operational: The requirement is an operational concern or can be addressed by an operational deployment

- Implementation: The requirement is an implementation concern

If there is no entry in the Supported By column, then there is a gap that must be filled.

ID	Name	Supported By
G-001	Solution Extensibility	XMPP-Core
G-002	Interoperability	XMPP

G-003	Scalability	XMPP
G-004	Versatility	XMPP-Core
G-005	Information Extensibility	XMPP-Core
G-006	Data Protection	Operational
G-007	Data Partitioning	Operational
G-008	Versioning and Backward Compatibility	XEP-0115/0030
G-009	Information Discovery	XEP-0030
G-010	Target Endpoint Discovery	XMPP-Core
G-011	Push and Pull Access	XEP-0060/0312
G-012	SACM Component Interface	N/A
G-013	Endpoint Location and Network Topology	
G-014	Target Endpoint Identity	XMPP-Core
G-015	Data Access Control	
ARCH-001	Component Functions	XMPP
ARCH-002	Scalability	XMPP-Core
ARCH-003	Flexibility	XMPP-Core
ARCH-004	Separation of Data and Management Functions	
ARCH-005	Topology Flexibility	XMPP-Core
ARCH-006	Capability Negotiation	XEP-0115/0030
ARCH-007	Role-Based Authorization	XMPP-Core
ARCH-008	Context-Based Authorization	
ARCH-009	Time Synchronization	Operational
IM-001	Extensible Attribute Vocabulary	N/A

IM-002	Posture Data Publication	N/A
IM-003	Data Model Negotiation	N/A
IM-004	Data Model Identification	N/A
IM-005	Data Lifetime Management	N/A
IM-006	Singularity and Modularity	N/A
DM-001	Element Association	N/A
DM-002	Data Model Structure	N/A
DM-003	Search Flexibility	N/A
DM-004	Full vs. Partial Updates	N/A
DM-005	Loose Coupling	N/A
DM-006	Data Cardinality	N/A
DM-007	Data Model Negotiation	N/A
DM-008	Data Origin	N/A
DM-009	Origination Time	N/A
DM-010	Data Generation	N/A
DM-011	Data Source	N/A
DM-012	Data Updates	N/A
DM-013	Multiple Collectors	N/A
DM-014	Attribute Extensibility	N/A
DM-015	Solicited vs. Unsolicited Updates	N/A
DM-016	Transfer Agnostic	N/A
OP-001	Time Synchronization	
OP-002	Collection Abstraction	
OP-003	Collection Composition	

OP-004	Attribute-Based Query	
OP-005	Information-Based Query with Filtering	
OP-006	Operation Scalability	
OP-007	Data Abstraction	
OP-008	Provider Restriction	
T-001	Multiple Transfer Protocol Support	Architecture
T-002	Data Integrity	Operational
T-003	Data Confidentiality	Operational
T-004	Transfer Protection	
T-005	Transfer Reliability	
T-006	Transfer-Layer Requirements	
T-007	Transfer Protocol Adoption	Architecture

Appendix B. Example Components

B.1. Policy Services

Consider a policy server conforming to [I-D.ietf-mile-rolie]. [I-D.ietf-mile-rolie] describes a RESTful way based on the ATOM Publishing Protocol ([RFC5023]) to find specific data collections. While this represents a specific binding (i.e. RESTful API based on [RFC5023]), there is a more abstract way to look at ROLIE.

ROLIE provides notional workspaces and collections, and provides the concept of information categories and links. Strictly speaking, these are logical concepts independent of the RESTful binding ROLIE specifies. In other words, ROLIE binds a logical interface (i.e. GET workspace, GET collection, SET entry, and so on) to a specific mechanism (namely an ATOM Publication Protocol extension).

It is not inconceivable to believe there could be a different interface mechanism, or a connector, providing these same operations using XMPP-Grid as the transfer mechanism.

Even if a [I-D.ietf-mile-rolie] server were external to an organization, there would be a need for a policy source inside the

organization as well, and it may be preferred for such a policy source to be connected directly to the ecosystem's communication infrastructure.

B.2. Software Inventory

The SACM working group has accepted work on the Endpoint Compliance Profile [I-D.ietf-sacm-ecp], which describes a collection architecture and may be viewed as a collector coupled with a collection-specific repository.

```
                                 Posture Manager              Endpoint
                Orchestrator    +---------------+        +---------------+
                +--------+      |               |        |               |
                |        |      | +-----------+ |        | +-----------+ |
                |        |<---->| | Posture   | |        | | Posture   | |
                |        | pub/ | | Validator | |        | | Collector | |
                |        | sub  | +-----------+ |        | +-----------+ |
                +--------+      |      |        |        |      |        |
                                |      |        |        |      |        |
Evaluator       Repository      |      |        |        |      |        |
+------+        +--------+      | +-----------+ |<-------| +-----------+ |
|      |        |        |      | | Posture   | | report | | Posture   | |
|      |        |        |      | | Collection| |        | | Collection| |
|      |<-----> |        |<-----| | Manager   | | query  | | Engine    | |
|      |request/|        | store| +-----------+ |------->| +-----------+ |
|      |respond |        |      |               |        |               |
|      |        |        |      |               |        |               |
+------+        +--------+      +---------------+        +---------------+
```

Figure 3: ECP Collection Architecture

In Figure 3, any of the communications between the Posture Manager and ECP components to its left could be performed directly or indirectly using a given message transfer mechanism. For example, the pub/sub interface between the Orchestrator and the Posture Manager could be using a proprietary method or using [I-D.ietf-mile-xmpp-grid] or some other pub/sub mechanism. Similarly, the store connection from the Posture Manager to the Repository could be performed internally to a given implementation, via a RESTful API invocation over HTTPS, or even over a pub/sub mechanism.

Our assertion is that the Evaluator, Repository, Orchestrator, and Posture Manager all have the potential to represent SACM Components with specific capability interfaces that can be logically specified,

then bound to one or more specific transfer mechanisms (i.e. RESTful API, [I-D.ietf-mile-rolie], [I-D.ietf-mile-xmpp-grid], and so on).

B.3. Datastream Collection

[NIST800126], also known as SCAP 1.3, provides the technical specifications for a "datastream collection". The specification describes the "datastream collection" as being "composed of SCAP data streams and SCAP source components". A "datastream" provides an encapsulation of the SCAP source components required to, for example, perform configuration assessment on a given endpoint. These source components include XCCDF checklists, OVAL Definitions, and CPE Dictionary information. A single "datastream collection" may encapsulate multiple "datastreams", and reference any number of SCAP components. Datastream collections were intended to provide an envelope enabling transfer of SCAP data more easily.

The [NIST800126] specification also defines the "SCAP result data stream" as being conformant to the Asset Reporting Format specification, defined in [NISTIR7694]. The Asset Reporting Format provides an encapsulation of the SCAP source components, Asset Information, and SCAP result components, such as system characteristics and state evaluation results.

What [NIST800126]did not do is specify the interface for finding or acquiring source datastream information, nor an interface for publishing result information. Discovering the actual resources for this information could be done via ROLIE, as described in the Policy Services section above, but other repositories of SCAP data exist as well.

B.4. Network Configuration Collection

[draft-birkholz-sacm-yang-content] illustrates a SACM Component incorporating a YANG Push client function and an XMPP-grid publisher function. [draft-birkholz-sacm-yang-content] further states "the output of the YANG Push client function is encapsulated in a SACM Content Element envelope, which is again encapsulated in a SACM statement envelope" which are published, essentially, via an XMPP-Grid Connector for SACM Components also part of the XMPP-Grid.

This is a specific example of an existing collection mechanism being adapted to the XMPP-Grid message transfer system.

Authors' Addresses

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

Email: adam.w.montville@gmail.com

Bill Munyan
Center for Internet Security
31 Tech Valley Drive
East Greenbush, NY 12061
USA

Email: bill.munyan.ietf@gmail.com