Security Automation and Continuous Monitoring (SACM) Architecture
draft-ietf-sacm-arch-02

Abstract

This memo defines a Security Automation and Continuous Monitoring (SACM) architecture. This work is built upon [RFC8600], 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].

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 27, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
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

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

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.
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. 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.

The enterprise boundary is not intended to imply a physical boundary. Rather, the enterprise boundary is intended to be inclusive of various cloud environments and vendor-provided services in addition to any physical systems the enterprise operates.

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, or more, of several roles relevant to the ecosystem. Generally each role is either a consumer of information or a provider of information.
4. Relevant Workflows

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

4.1. 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.

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

4.2. 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).

### 4.3. 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.

5. Configuration Management Components, Interactions, and Capabilities

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

5.1. 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.

5.2. Interactions

SACM Components are intended to interact with other SACM Components. These interactions can be thought of, at the level of this architectural approach, as the combination of interfaces with their supported operations.

- Store: One component stores information in another.
- Ask: A component requests information from another.
- Notify/Ask: A component notifies another component, which then asks the notifying component (or another component) for information.
- Publish/Subscribe: A component publishes information to a messaging system and a set of other components, subscribed to that information type, receive the published information.
- Tell: A component instructs another.

TODO: Consider breaking out Notify, Publish, and Subscribe into separate line items, and adding Error (a type of Notify). Then consider explaining the necessary combinations relevant to the configuration assessment workflow below.

Each interaction will convey a payload of information. The payload information is expected to contain sub-domain-specific characteristics and instructions.

5.3. 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".

6. 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.

Figure 2: Configuration Assessment Component Interactions

Figure 2 depicts configuration assessment components and their interactions, which are further described below.


2. The Orchestrator obtains collection information from the Policy Repository: TODO - add specific interaction options here.
3. The Orchestrator initiates collection to be performed by the Collection Sub-Architecture: TODO - add specific interaction options here.


5. The Orchestrator initiates the Evaluator (optionally with evaluation information gathered from the Policy Repository): TODO - add specific interaction options here
   1. The Evaluator obtains evaluation information from the Policy Repository (optionally): TODO - add specific interaction options here
   2. The Evaluator obtains relevant posture attributes from the Posture Attribute Repository: TODO - add specific interaction options here

6. Evaluation results are stored in the Evaluation Results Repository: TODO - add specific interaction options here

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, the Tell 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.

7. Privacy Considerations
   TODO

8. Security Considerations
   TODO

9. IANA Considerations
   TODO: Revamp this section after the configuration assessment workflow is fleshed out.

IANA tables can probably be used to make life a little easier. We would like a place to enumerate:
0. Capability/operation semantics
0. SACM Component implementation identifiers
0. SACM Component versions
0. Associations of SACM Components (and versions) to specific Capabilities
0. Collection sub-architecture Identification

10. References

10.1. Normative References


10.2. Informative References


Appendix A. Mapping to RFC8248

TODO: Consider removing or placing in a separate solution draft.

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.
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Supported By</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-001</td>
<td>Solution Extensibility</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>G-002</td>
<td>Interoperability</td>
<td>XMPP</td>
</tr>
<tr>
<td>G-003</td>
<td>Scalability</td>
<td>XMPP</td>
</tr>
<tr>
<td>G-004</td>
<td>Versatility</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>G-005</td>
<td>Information Extensibility</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>G-006</td>
<td>Data Protection</td>
<td>Operational</td>
</tr>
<tr>
<td>G-007</td>
<td>Data Partitioning</td>
<td>Operational</td>
</tr>
<tr>
<td>G-008</td>
<td>Versioning and Backward Compatibility</td>
<td>XEP-0115/0030</td>
</tr>
<tr>
<td>G-009</td>
<td>Information Discovery</td>
<td>XEP-0030</td>
</tr>
<tr>
<td>G-010</td>
<td>Target Endpoint Discovery</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>G-011</td>
<td>Push and Pull Access</td>
<td>XEP-0060/0312</td>
</tr>
<tr>
<td>G-012</td>
<td>SACM Component Interface</td>
<td>N/A</td>
</tr>
<tr>
<td>G-013</td>
<td>Endpoint Location and Network Topology</td>
<td></td>
</tr>
<tr>
<td>G-014</td>
<td>Target Endpoint Identity</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>G-015</td>
<td>Data Access Control</td>
<td></td>
</tr>
<tr>
<td>ARCH-001</td>
<td>Component Functions</td>
<td>XMPP</td>
</tr>
<tr>
<td>ARCH-002</td>
<td>Scalability</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>ARCH-003</td>
<td>Flexibility</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>ARCH-004</td>
<td>Separation of Data and Management Functions</td>
<td></td>
</tr>
<tr>
<td>ARCH-005</td>
<td>Topology Flexibility</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>ARCH-006</td>
<td>Capability Negotiation</td>
<td>XEP-0115/0030</td>
</tr>
<tr>
<td>ARCH-007</td>
<td>Role-Based Authorization</td>
<td>XMPP-Core</td>
</tr>
<tr>
<td>ARCH-008</td>
<td>Context-Based Authorization</td>
<td></td>
</tr>
<tr>
<td>ARCH-009</td>
<td>Time Synchronization</td>
<td>Operational</td>
</tr>
<tr>
<td>IM-001</td>
<td>Extensible Attribute Vocabulary</td>
<td>N/A</td>
</tr>
<tr>
<td>IM-002</td>
<td>Posture Data Publication</td>
<td>N/A</td>
</tr>
<tr>
<td>IM-003</td>
<td>Data Model Negotiation</td>
<td>N/A</td>
</tr>
<tr>
<td>IM-004</td>
<td>Data Model Identification</td>
<td>N/A</td>
</tr>
<tr>
<td>IM-005</td>
<td>Data Lifetime Management</td>
<td>N/A</td>
</tr>
<tr>
<td>IM-006</td>
<td>Singularity and Modularity</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-001</td>
<td>Element Association</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-002</td>
<td>Data Model Structure</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-003</td>
<td>Search Flexibility</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-004</td>
<td>Full vs. Partial Updates</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-005</td>
<td>Loose Coupling</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-006</td>
<td>Data Cardinality</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-007</td>
<td>Data Model Negotiation</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-008</td>
<td>Data Origin</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-009</td>
<td>Origination Time</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-010</td>
<td>Data Generation</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-011</td>
<td>Data Source</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-012</td>
<td>Data Updates</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-013</td>
<td>Multiple Collectors</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-014</td>
<td>Attribute Extensibility</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-015</td>
<td>Solicited vs. Unsolicited Updates</td>
<td>N/A</td>
</tr>
<tr>
<td>DM-016</td>
<td>Transfer Agnostic</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Appendix B. Example Components

TODO: Consider removing.

B.1. Policy Services

Consider a policy server conforming to [RFC8322]. [RFC8322] 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 [RFC8322] 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 Posture Collection Profile [I-D.ietf-sacm-ecp], which describes a collection architecture and may be viewed as a collector coupled with a collection-specific repository.

![Diagram of EPCP Collection Architecture]

Figure 3: EPCP Collection Architecture

In Figure 3, any of the communications between the Posture Manager and EPCP 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 [RFC8600] 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, [RFC8322], [RFC8600], 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.

Appendix C. Exploring An XMPP-based Solution

TODO: Consider removing or placing in a separate draft.

Ongoing work has been taking place around and during IETF hackathons. The list of hackathon efforts follows:

- [HACK99]: A partial implementation of a vulnerability assessment scenario involving an [I-D.ietf-sacm-ecp] implementation, a [RFC8322] implementation, and a proprietary evaluator to pull the pieces together.

- [HACK100]: Work to combine the vulnerability assessment scenario from [HACK99] with an XMPP-based YANG push model.

- [HACK101]: A fully automated configuration assessment implementation using XMPP (specifically Publish/Subscribe capabilities) as a communication mechanism.

- [HACK102]: An exploration of how we might model assessment, collection, and evaluation abstractly, and then rely on YANG expressions for the attributes of traditional endpoints.

- [HACK103]: No SACM participation at the Bangkok hackathon.

- [HACK104]: Basic XMPP-to-Concise MAP - Created an XMPP adapter that can accept basic posture attributes and translate them to Concise MAP. This hackathon only proved the concept that system characteristics information can be transported via XMPP and translated to a (very basic) concise MAP implementation.

- [HACK105]: Advanced XMPP-to-Concise MAP: Full orchestration of collection capabilities using XMPP. Collector implementations extend the core XMPP structure to allow OVAL collection instructions (OVAL objects) to inform posture attribute collection. Collected system characteristics can be provided to the Concise MAP XMPP adapter using all 3 available XMPP capabilities: Publish/Subscribe, Information Query (iq - request/response) stanzas, or direct Message stanzas. CDDL was created to map collected posture attributes to Concise MAP structure. The XMPP adapter translates the incoming system characteristics and stores the information in the MAP.

Figure 4 depicts a slightly more detailed view of the architecture (within the enterprise boundary) - one that fosters the development
of a pluggable ecosystem of cooperative tools. Existing collection mechanisms 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 4, this architecture does allow point-to-point interfaces. In fact, [RFC8600] provides brokering capabilities to facilitate such point-to-point data transfers. Additionally, each of the SACM Components depicted in Figure 4 may be a provider, a consumer, or both, depending on the workflow in context.

```
+--------------+           +--------------+
| Orchestrator |           | Repositories |
+-------------+           +-------------+
|              |                          |
+--------------+           +--------------+
|                XMPP-Grid+                 <-----| Downstream Uses |
+-----------------^------------------------+
|                                   |
+------------------------v---------------------------------+
| XMPP-Grid Connector(s)               +-----------------+
+------------------------^------------------------+
|                                   |
+-------------v------------------------+
| Collector(s) |
+-------------+
```

Figure 4: XMPP-based Architecture

[RFC8600] details a number of XMPP extensions (XEPs) that MUST be utilized to meet the needs of [RFC7632] and [RFC8248]:

- Service Discovery (XEP-0030): Service Discovery allows XMPP entities to discover information about other XMPP entities. Two kinds of information can be discovered: the identity and capabilities of an entity, such as supported features, and items associated with an entity.

- Publish-Subscribe (XEP-0060): The PubSub extension enables entities to create nodes (topics) at a PubSub service and publish information at those nodes. Once published, an event notification is broadcast to all entities that have subscribed to that node.
At this point, [RFC8600] 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 4 we therefore use "XMPP-Grid+" to indicate something more than [RFC8600] 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) [RFC8600] to include additional XEPs – possibly the following:

1. **Entity Capabilities (XEP-0115):** This extension defines the methods for broadcasting and dynamically discovering an entities’ capabilities. This information is transported via standard XMPP presence. Example capabilities that could be discovered could include support for posture attribute collection, support for specific types of posture attribute collection such as EPCP, SWIMA, OVAL, or YANG. Other capabilities are still to be determined.

2. **Ad Hoc Commands (XEP-0050):** This extension allows an XMPP entity to advertise and execute application-specific commands. Typically the commands contain data forms (XEP-0004) in order to structure the information exchange. This extension may be usable for simple orchestration (i.e. "do assessment").

3. **HTTP File Upload (XEP-0363):** The HTTP File Upload extension allows for large data sets to be published to a specific path on an HTTP server, and receive a URL from which that file can later be downloaded again. XMPP messages and IQs are meant to be compact, and large data sets, such as collected posture attributes, may exceed a message size threshold. Usage of this XEP allows those larger data sets to be persisted, thus necessitating only the download URL to be passed via XMPP messages.

4. **Personal Eventing Protocol (XEP-0163):** The Personal Eventing Protocol can be thought of as a virtual PubSub service, allowing an XMPP account to publish events only to their roster instead of a generic PubSub topic. This XEP may be useful in the cases when collection requests or queries are only intended for a subset of endpoints and not an entire subscriber set.

5. **File Repository and Sharing (XEP-0214):** This extension defines a method for XMPP entities to designate a set of file available for retrieval by other users of their choosing, and is based on PubSub Collections.

6. **Easy User Onboarding (XEP-401):** The goal of this extension is simplified client registration, and may be useful when adding new endpoints or SACM components to the ecosystem.
Bidirectional-streams Over Synchronous HTTP (BOSH) (XEP-0124): BOSH emulates the semantics of a long-lived, bidirectional TCP connection between two entities (aka "long polling"). Consider a SACM component that is updated dynamically, i.e. an internal vulnerability definition repository ingesting data from a Feed/Repository of External Data, and a second SACM component such as an Orchestrator. Using BOSH, the Orchestrator can effectively continuously poll the vulnerability definition repository for changes/updates.

PUBSUB Collection Nodes (XEP-0248): Effectively an extension to XEP-0060 (Publish-Subscribe), PUBSUB Collections aim to simplify an entities’ subscription to multiple related topics, and establishes a "node graph" relating parent nodes to its descendents. An example "node graph" could be rooted in a "vulnerability definitions" topic, and contain descendent topics for OS family-level vulnerability definitions (i.e. Windows), and further for OS family version-level definitions (i.e. Windows 10 or Windows Server 2016).

PUBSUB Since (XEP-0312): This extension enables a subscriber to automatically receive PUBSUB and Personal Eventing Protocol (PEP) notifications since its last logout time. This extension may be useful in intermittent connection scenarios, or when entities disconnect and reconnect to the ecosystem.

PUBSUB Chaining (XEP-0253): This extension describes the federation of publishing nodes, enabling a publish node of one server to be a subscriber to a publishing node of another server.

C.1. Example Architecture using XMPP-Grid and Endpoint Posture Collection Protocol

Figure 5 depicts a further detailed view of the architecture including the Endpoint Posture Collection Protocol as the collection subsystem, illustrating the idea of a pluggable ecosystem of cooperative tools.
Figure 5: XMPP-based Architecture including EPCP
Authors’ Addresses

Adam W. Montville
Center for Internet Security
31 Tech Valley Drive
East Greenbush, NY 12061
USA
Email: adam.montville.sdo@gmail.com

Bill Munyan
Center for Internet Security
31 Tech Valley Drive
East Greenbush, NY 12061
USA
Email: bill.munyan.ietf@gmail.com