SACM Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 22, 2018
H. Birkholz
Fraunhofer SIT
N. Cam-Winget
Cisco Systems
January 18, 2018

YANG subscribed notifications via SACM Statements
draft-birkholz-sacm-yang-content-01

Abstract

This document summarizes a subset of the emerging generic SACM Data Model for inter-component distribution of SACM Content in and between SACM Domains. The subset defined in this document is covering every information element that can be acquired using YANG based protocols, i.e. NETCONF, RESTCONF, COMI or derived mechanisms that transfer YANG modeled data, such as MUD. As subscriptions to data origins in a SACM domain are one of the architectural corner-stones of the SACM architecture, this document recommends the use of YANG Push, YANG subscribed Notifications and corresponding Notification Headers and Bundles. Analogously, a mapping of Notification Header content to SACM Metadata is provided in this document.

Table of Contents

1. Introduction

This document defines an XML encoding of SACM Statements that contain SACM Content composed of YANG modeled data (i.e. NETCONF messages). Correspondingly, this documents provides a standardized mapping to derive SACM Metadata from YANG Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] using Notification Message Headers and Bundles [I-D.ietf-netconf-notification-messages] content.

Every message defined in the generic SACM Data Model is a SACM Statement. The SACM Statement structure is provided by the SACM Information Model. In consequence, a SACM Statement is an Information Element not acquired by, but created by SACM Components for inter-component distribution of SACM Content (Information Elements on the Data Plane that represent information about Target Endpoints (TE) or Guidance. Examples include: software identifiers, assessment guidance/results, ECA Policy rules, or VDD).

YANG modules are a powerful established tool to provide Information Elements about Target Endpoints with well-defined semantics. YANG Push [I-D.ietf-netconf-yang-push] and the corresponding YANG Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] drafts make use of these modules to create streams of notifications (YANG telemetry). Subscriptions to YANG data stores or YANG streams are Data Sources that provide Information Elements that can be acquired by SACM Collectors to provide SACM Content on the Data Plane.

Analogously, filter expressions used in the context of YANG subscriptions constitute SACM Content that is Imperative Guidance consumed by SACM Components on the Management Plane in order to create YANG telemetry.

In this document (not including the abstract, of course), terms that are Capitalized or prefixed with SACM are defined in the SACM Terminology document.

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].

3. Brokering of YANG Push Telemetry via SACM Statements

Every SACM Content is published into a SACM Domain using a statement envelope/encapsulation. The general structure of a SACM Statement is based on the Information Element definition found in [I-D.ietf-sacm-information-model] and can be summarized as follows:

- a sacm-statement encapsulates statement-metadata and content-elements

- a content-element encapsulates content-metadata and SACM Content

In the scope of this document, only one type of SACM Content is covered: YANG modeled data. Correspondingly, the minimal required structure of statements, statement-metadata, content-elements, and content-metadata are defined. A complete XML schema definition of this subset of the generic SACM Data Model can be found in Appendix A.

4. Encapsulation of YANG notifications in SACM content-elements

A YANG notification is associated with a set of YANG specific metadata as defined in [I-D.ietf-netconf-notification-messages]. Hence, SACM Content that is derived from a YANG notification published to a SACM Domain MUST be encapsulated with its corresponding Metadata in a content-element as defined below.

YANG output that is SACM Content MUST be represented according to the XSD definition included in the content choice of the content-element.

```
<CODE BEGINS>
<xs:complexType name="content-element">
  <xs:sequence>
    <xs:element name="content-metadata" type="content-metadata" maxOccurs="unbou
nded"/>
    <xs:choice>
      <xs:element name="yang-output" type="yang-output" />
        <!-- There is only one element here now, but virtually every other conte
nt choice
             will go here, i.e. data models, such as OVAL, SCAP, SWID, etc. -->
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<CODE ENDS>
```

4.1. Enumeration definition for content-type

An occurrence of the yang-output element MUST be instantiated in the content-metadata element, if YANG Push output is to be transferred. Also, the content-type MUST be set to the enumeration value "yang-output", respectively.

In general, the list of content-type enumerations is including every subject as defined in the SACM Information Model. Regarding the definition of the subset of the generic SACM Data Model provided by this document, the list of potential content-types is reduced to "yang-output". Please note, that the complete generic SACM Data Model includes additional content-type enumerations next to the definition provided by this document.

```
<CODE BEGINS>
<xs:simpleType name="content-type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="yang-output" />
       <!-- There is only one type here now, but virtually every other content-t
ype
            will go here, i.e. data models, such as OVAL, SCAP, SWID, etc. -->
  </xs:restriction>
</xs:simpleType>
<CODE ENDS>
```

4.2. Element definition for content-metadata

The list of optional elements included in content-metadata will incorporate any every potential metadata type. For the scope of this document, the list of elements is also limited to the minimal required set of metadata elements and the yang-output metadata element to support the encapsulation of NETCONF encoded subscribed notifications or YANG query result. As defined above, one occurrence of the yang-output element has to be included in the content-metadata element.

A more complete content-metadata element definition is illustrated in the Appendix A.

```
<CODE BEGINS>
<xs:complexType name="content-metadata">
  <xs:sequence>
    <xs:element name="content-element-guid" type="content-element-guid"/>
    <xs:element name="content-creation-timestamp" type="content-creation-timesta
mp"/>
    <xs:element name="content-topic" type="content-topic"/>
    <xs:element name="content-type" type="content-type"/>
    <xs:element name="data-source" type="data-source" minOccurs="0"/>
    <xs:element name="data-origin" type="data-origin" minOccurs="0"/>
    <xs:element name="relationship" type="relationship" minOccurs="0" maxOccurs=
"unbounded"/>
    <xs:element name="yang-output-metadata" type="yang-output-metadata" minOccur
s="0"/>
  </xs:sequence>
</xs:complexType>
<CODE ENDS>
```

4.3. Definition of the yang-output-metadata element included in content-metadata

The composition of metadata that can be associated with a XML NETCONF result depends on multiple factors:

- o acquisition method: query / subscription
- o encoding: XML # more content encodings will be supported as indicated by the definition
- o subscription interval: periodic / on-change
- o filter-type: xpath / subtree

Additionally, the actual filter expression (or in future iterations of this work, a referencing Label, such as a URI, UUID or other composed identifier) has to be included in the content-metadata.

```
<CODE BEGINS>
<xs:complexType name="yang-output-metadata">
  <xs:sequence>
    <xs:choice maxOccurs="1">
      <xs:element name="yang-query" type="yang-query-value" />
      <xs:element name="yang-subscribtion" type="yang-subscribtion-type" />
    </xs:choice>
    <xs:element name="encoding" type="yang-encoding" />
    <xs:element name="module-names" type="module-name" minOccurs="0" maxOccurs="
unbounded" />
    <xs:element name="filter-expression" type="filter-expression-value" minOccur
s="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="yang-subscribtion-type">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="periodic" />
    <xs:enumeration value="on-change" />
  </xs:restriction>
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="xpath" />
    <xs:enumeration value="subtree" />
  </xs:restriction>
<xs:complexType>

<xs:simpleType name="filter-expression-value">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="yang-query-value">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="yang-encoding">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="netconf" />
    <xs:enumeration value="restconf" />
    <xs:enumeration value="comi" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="module-name">
  <xs:restriction base="xs:string" />
</xs:simpleType>
<CODE ENDS>
```

5. Mapping of YANG Bundled Notifications to SACM Metadata

[I-D.ietf-netconf-notification-messages] includes the following definition:

```
       yang-data bundled-message
          +-- bundled-message-header
          |  +-- message-time                 yang:date-and-time
          |  +-- message-id?                  uint32
          |  +-- previous-message-id?         uint32
          |  +-- message-generator-id?        string
          |  +-- signature?                   string
          |  +-- notification-count?          uint16
          +-- notifications*
             +-- notification-header
             |  +-- notification-time         yang:date-and-time
             |  +-- subscription-id*          uint32
             |  +-- notification-id?          uint32
             |  +-- module?                   yang-identifier
             |  +-- notification-type?        notification
             |  +-- observation-domain-id?    string
             +-- receiver-record-contents?
```

The corresponding mapping MUST be used when deriving SACM Content Metadata for content-metadata items from YANG modeled data corresponding to YANG Notification Message Headers and Bundles:

```
     notification-time -> content-creation-timestamp
     subscription-id + (observation-domain-id OR "SACM Component Label") -> c
ontent-element-guid
     module -> module-names
     notification-type -> yang-subscribtion-type
     receiver-record-contents -> content-elements
```

If there are more than one receiver-record-contents instanced included in the received Notification Message Bundle, multiple content-elements MUST be instanciated, accordingly.

The following mapping MUST be used when deriving SACM Statement Metadata (see Appendix A) statement-metadata items representing NETCONF instances adhering to the definition of YANG Notification Message Headers and Bundles:

```
     message-id -> statement-guid
     "SACM Component Label" -> data-origin
     message-time -> statement-creation-timestamp
     "SACM Component Publictation Time" -> statement-publish-timestamp
     statement-type -> "Observation"
```

"SACM Component Publicatation Time" can only be inferred by the SACM Component using its "most trustworthy source of time".

If there is not receiver-record-contents included in the YANG notification, a SACM Component MUST NOT publish a corresponding SACM Statement to the SACM Domain.

6. SACM Component Composition

A SACM Component able to process YANG subscribed notifications requires at least two functions:

- o a SACM Function supporting YANG Push and YANG Notification Headers and
- o Bundles function [I-D.ietf-netconf-yang-push], [I-D.ietf-netconf-subscribed-notifications], and
- o an xmpp-grid provider function [I-D.ietf-mile-xmpp-grid]

Orchestration of functions inside a component, their discovery as capabilities and the internal distribution of SACM Content inside a SACM Component is out of scope of this document. # for now

7. IANA considerations

This document includes requests to IANA.

8. Security Considerations

TBD

9. Acknowledgements

Christoph Vigano, Guangying Zheng, Eric Voit, Alexander Clemm

10. Change Log

First version -00

Second version -01 * generalized the content of the document, detaching it from the implementation created at the Hackaton of IETF 99 * included a mapping of the -03 version of the YANG Notification Headers and Bundles draft to this draft

11. Contributors

Eric Voit

12. Normative References

[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-04 (work in progress), October 2017.

[I-D.ietf-netconf-notification-messages]
Voit, E., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-02 (work in progress), October 2017.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Streams", draft-ietf-netconf-subscribed-notifications-08 (work in progress), December 2017.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-12 (work in progress), December 2017.

[I-D.ietf-sacm-information-model]
Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus, M., Haynes, D., and H. Birkholz, "SACM Information Model", draft-ietf-sacm-information-model-10 (work in progress), April 2017.

[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>.

Appendix A. Minimal SACM Statement Definition for YANG Output

The definitions of statements, statement-metadata, content-element, and content-metadata are provided by the SACM Information Model [I-D.ietf-sacm-information-model].

Due to the stripping down of content-elements to YANG output, the enumerations still included in the relationship-type are not able to point to other types of content in the scope of this document, but are able to reference other content-types in the scope of the generic SACM Data Model.

```
<CODE BEGINS>
<?xml version="1.0"?>
<xs:schema version="1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:complexType name="StatementMetadata">
    <xs:sequence>
      <xs:element name="statement-guid" type="statement-guid" />
      <xs:element name="data-origin" type="data-origin" />
      <xs:element name="statement-creation-timestamp" type="statement-creation-t
imestamp" minOccurs="0" />
      <xs:element name="statement-publish-timestamp" type="statement-creation-ti
mestamp" />
      <xs:element name="statement-type" type="statement-type" />
      <xs:element name="content-elements" type="content-elements" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="sacm-statement">
    <xs:sequence>
      <xs:element name="statement-metadata" type="StatementMetadata" />
      <xs:element name="content-element" type="content-element" minOccurs="1" ma
xOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:element name="sacm-statement" type="sacm-statement">
  </xs:element>

  <xs:simpleType name="statement-guid">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="decimal-fraction-denominator">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="decimal-fraction-numerator">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="content-elements">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:complexType name="statement-creation-timestamp">
```

```
    <xs:sequence>
      <xs:element name="decimal-fraction-denominator" type="decimal-fraction-den
ominator"/>
      <xs:element name="decimal-fraction-numerator" type="decimal-fraction-numer
ator"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="content-creation-timestamp">
    <xs:sequence>
      <xs:element name="decimal-fraction-denominator" type="decimal-fraction-den
ominator"/>
      <xs:element name="decimal-fraction-numerator" type="decimal-fraction-numer
ator"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="statement-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Observation" />
      <xs:enumeration value="DirectoryContent" />
      <xs:enumeration value="Correlation" />
      <xs:enumeration value="Assessment" />
      <xs:enumeration value="Guidance" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="content-topic">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Session" />
      <xs:enumeration value="User" />
      <xs:enumeration value="Interface" />
      <xs:enumeration value="PostureProfile" />
      <xs:enumeration value="Flow" />
      <xs:enumeration value="PostureAssessment" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="content-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="EndpointConfiguration" />
      <xs:enumeration value="EndpointState" />
      <xs:enumeration value="DirectoryEntry" />
      <xs:enumeration value="Event" />
      <xs:enumeration value="Incident" />
      <xs:enumeration value="yang-output" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="content-element-guid">
    <xs:restriction base="xs:string" />
  </xs:simpleType>
```

```
  <xs:complexType name="yang-output-metadata">
    <xs:sequence>
      <xs:choice maxOccurs="1">
        <xs:element name="yang-query" type="yang-query" />
        <xs:element name="yang-subscribe" type="yang-subscribe" />
      </xs:choice>
      <xs:element name="encoding" type="yang-encoding" />
      <xs:element name="module-names" type="module-name" minOccurs="0" maxOccurs
="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="yang-subscribe">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="periodic" />
      <xs:enumeration value="on-change" />
    </xs:restriction>
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="xpath" />
      <xs:enumeration value="subtree" />
    </xs:restriction>
  </xs:complexType>

  <xs:simpleType name="filter-expression">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="yang-query">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="yang-encoding">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="netconf" />
      <xs:enumeration value="restconf" />
      <xs:enumeration value="comi" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="module-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="relationship-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="associated_with_user" />
      <xs:enumeration value="applies_to_session" />
      <xs:enumeration value="seen_on_interface" />
      <xs:enumeration value="associated_with_flow" />
```

```
      <xs:enumeration value="contains_virtual_device" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="relationship-content-element-guid">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="relationship-statement-guid">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="relationship-object-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="data-source-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="data-origin">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="host-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="administrative-domain-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="sub-administrative-domain">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="super-administrative-domain">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="relationship">
    <xs:sequence>
      <xs:element name="relationship-type" type="relationship-type" />
      <xs:element name="relationship-content-element-guid" type="relationship-co
ntent-element-guid" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="relationship-statement-guid" type="relationship-statemen
t-guid" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="relationship-object-label" type="relationship-object-lab
el" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
```

```
  <xs:complexType name="flow-element">
    <xs:sequence>
      <xs:element name="network-address" type="network-address"/>
      <xs:element name="layer4-port-address" type="layer4-port-address" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="flow-record">
    <xs:sequence>
      <xs:element name="src-flow-element" type="flow-element" />
      <xs:element name="dst-flow-element" type="flow-element" />
      <xs:element name="protocol" type="protocol" />
      <xs:element name="layer4-protocol" type="layer4-protocol" />
      <xs:element name="flow-statistics" type="flow-statistics" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="content-metadata">
    <xs:sequence>
      <xs:element name="content-element-guid" type="content-element-guid" />
      <xs:element name="content-creation-timestamp" type="content-creation-times
tamp" />
      <xs:element name="content-topic" type="content-topic" />
      <xs:element name="content-type" type="content-type" />
      <xs:element name="data-source" type="data-source" minOccurs="0" />
      <xs:element name="data-origin" type="data-origin" minOccurs="0" />
      <xs:element name="relationship" type="relationship" minOccurs="0" maxOccur
s="unbounded"/>
      <xs:element name="yang-output-metadata" type="yang-output-metadata" minOcc
urs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="content-element">
    <xs:sequence>
      <xs:element name="content-metadata" type="content-metadata" maxOccurs="unb
ounded"/>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="yang-output" type="yang-output" />
        <xs:element name="flow" type="flow-record" />
        <xs:element name="posture" type="xs:string" />
        <xs:element name="user" type="user" />
        <xs:element name="session" type="session" />
        <xs:element name="ethernet-interface" type="ethernet-interface" />
        <xs:element name="target-endpoint" type="target-endpoint" />
        <xs:element name="port" type="port" />
        <xs:element name="posture-assessment" type="posture-assessment" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="posture-assessment"></xs:complexType>
```

```
  <xs:complexType name="target-endpoint">
    <xs:sequence>
      <xs:element name="host-name" type="host-name" />
      <xs:element name="te-label" type="te-label" />
      <xs:element name="administrative-domain" type="administrative-domain" minO
ccurs="0" />
      <xs:element name="application-instance" type="application-instance" minOcc
urs="0" maxOccurs="unbounded" />
      <xs:element name="ethernet-interface" type="ethernet-interface" minOccurs=
"0" maxOccurs="unbounded" />
      <xs:element name="address-association" type="address-association" minOccur
s="0" maxOccurs="unbounded" />
      <xs:element name="data-source" type="data-source" minOccurs="0" />
      <xs:element name="operating-system" type="operating-system" minOccurs="0"
/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="te-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="application-instance">
    <xs:sequence>
      <xs:element name="application-label" type="application-label" />
      <xs:element name="target-endpoint" type="target-endpoint" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="attribute-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="attribute-value">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="attribute-value-pair">
    <xs:sequence>
      <xs:element name="attribute-name" type="attribute-name" />
      <xs:element name="attribute-value" type="attribute-value" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="application-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="application-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="application-version">
```

```
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="application-manufacturer">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <!--     TODO: is it possible to declare this as an enumeration or is that unr
ealistic? -->
  <xs:simpleType name="application-type">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="application-component">
    <xs:restriction base="xs:string" />
  </xs:simpleType>



  <xs:complexType name="application">
    <xs:sequence>
      <xs:element name="application-label" type="application-label" minOccurs="0
" />
      <xs:element name="application-name" type="application-name" />
      <xs:element name="application-type" type="application-type" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="application-component" type="application-component" minO
ccurs="0" maxOccurs="unbounded" />
      <xs:element name="application-manufacturer" type="application-manufacturer
" minOccurs="0" />
      <xs:element name="application-version" type="application-version" minOccur
s="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="address-association">
    <xs:sequence>
      <xs:element name="address" type="address" />
      <xs:element name="address-association-type" type="address-association-type
" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="address">
    <xs:sequence>
      <xs:element name="address-mask-value" type="address-mask-value" />
      <xs:element name="address-type" type="address-type" />
      <xs:element name="address-value" type="address-value" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="address-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Ethernet" />
      <xs:enumeration value="ZigBee" />
```

```
      <xs:enumeration value="ModBus" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="session-state-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Authenticating"></xs:enumeration>
      <xs:enumeration value="Authenticated"></xs:enumeration>
      <xs:enumeration value="Postured"></xs:enumeration>
      <xs:enumeration value="Started"></xs:enumeration>
      <xs:enumeration value="Disconnected"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="session">
    <xs:sequence>
      <xs:element name="session-state-type" type="session-state-type" />
      <!-- TODO: add additional elements for Session Type -->
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="user-id">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="username">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="user-directory">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="user">
    <xs:sequence>
      <xs:element name="user-id" type="user-id" />
      <xs:element name="username" type="username" minOccurs="0" />
      <xs:element name="data-source" type="data-source" minOccurs="0" />
      <xs:element name="user-directory" type="user-directory" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ethernet-interface">
    <xs:sequence>
      <xs:element name="interface-label" type="interface-label" />
      <xs:element name="network-interface-name" type="network-interface-name" mi
nOccurs="0"/>
      <xs:element name="mac-address" type="mac-address" />
      <xs:element name="network-name" type="network-name" minOccurs="0"/>
```

```
      <xs:element name="network-id" type="network-id" minOccurs="0"/>
      <xs:element name="layer2-interface-type" type="layer2-interface-type" minO
ccurs="0"/>
      <xs:element name="sub-interface-label" type="sub-interface-label" minOccur
s="0" maxOccurs="unbounded"/>
      <xs:element name="super-interface-label" type="super-interface-label" minO
ccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="event-type">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="event-threshold">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="event-threshold-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="event-trigger">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="timestamp-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="discovered" />
      <xs:enumeration value="classified" />
      <xs:enumeration value="collected" />
      <xs:enumeration value="published" />
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="typed-timestamp">
    <xs:sequence>
      <xs:element name="decimal-fraction-denominator" type="decimal-fraction-den
ominator"/>
      <xs:element name="decimal-fraction-numerator" type="decimal-fraction-numer
ator"/>
      <xs:element name="timestamp-type" type="timestamp-type" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="event">
    <xs:sequence>
      <xs:element name="event-type" type="event-type" minOccurs="0" />
      <xs:element name="event-threshold" type="event-threshold" minOccurs="0" />
      <xs:element name="event-threshold-name" type="event-threshold-name" minOcc
urs="0" />
      <xs:element name="event-trigger" type="event-trigger" minOccurs="0" />
      <xs:element name="typed-timestamp" type="typed-timestamp" />
      <xs:element name="content" type="xs:anySimpleType" />
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="os-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="os-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="os-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="real-time" />
      <xs:enumeration value="consumer" />
      <xs:enumeration value="server" />
      <xs:enumeration value="security-enhanced" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="os-component">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="os-manufacturer">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="os-version">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="operating-system">
    <xs:sequence>
      <xs:element name="os-label" type="os-label" minOccurs="0" />
      <xs:element name="os-name" type="os-name" />
      <xs:element name="os-type" type="os-type" minOccurs="0" maxOccurs="unbound
ed" />
      <xs:element name="os-component" type="os-component" minOccurs="0" maxOccur
s="unbounded"/>
      <xs:element name="os-manufacturer" type="os-manufacturer" minOccurs="0" />
      <xs:element name="os-version" type="os-version" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="sub-interface-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="super-interface-label">
```

```
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="address-association-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="broadcast-domain-member-list" />
      <xs:enumeration value="ip-subnet-member-list" />
      <xs:enumeration value="ip-mac" />
      <xs:enumeration value="shared-backhaul-interface" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="address-mask-value">
    <xs:restriction base="xs:string" />
  </xs:simpleType>
  <xs:simpleType name="address-value">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="interface-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="network-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="network-id">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="network-interface-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="layer2-interface-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="fastEther" />
      <xs:enumeration value="fastEtherFX" />
      <xs:enumeration value="gigabitEthernet" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ipv6-address-subnet-mask-cidrnot">
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipv6-address-value">
```

```
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ipv4-address-subnet-mask-cidrnot">
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipv4-address-subnet-mask">
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipv4-address-value">
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="network-address">
    <xs:choice>
      <xs:element name="ipv4-address" type="ipv4-address" />
      <xs:element name="ipv6-address" type="ipv6-address" />
      <xs:element name="mac-address" type="mac-address" />
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="endpoint-identifier">
    <xs:choice>
      <xs:element name="certificate" type="certificate" />
      <xs:element name="firmware-id" type="firmware-id" />
      <xs:element name="hardware-serial-number" type="hardware-serial-number" />
      <xs:element name="host-name" type="host-name" />
      <xs:element name="ipv4-address-value" type="ipv4-address-value" />
      <xs:element name="ipv6-address-value" type="ipv6-address-value" />
      <xs:element name="mac-address" type="mac-address" />
      <xs:element name="public-key" type="public-key" />
      <xs:element name="username" type="username" />
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="ipv4-address">
    <xs:sequence>
      <xs:element name="ipv4-address-value" type="ipv4-address-value" />
      <xs:element name="ipv4-address-subnet-mask-cidrnot" type="ipv4-address-sub
net-mask-cidrnot"/>
      <xs:element name="ipv4-address-subnet-mask" type="ipv4-address-subnet-mask
"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ipv6-address">
```

```
    <xs:sequence>
      <xs:element name="ipv6-address-value" type="ipv6-address-value" />
      <xs:element name="ipv6-address-subnet-mask-cidrnot" type="ipv6-address-sub
net-mask-cidrnot"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="mac-address">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="layer4-port-address">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="protocol">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="layer4-protocol">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="bytes-received">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="bytes-sent">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="units-received">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="units-sent">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:complexType name="flow-statistics">
    <xs:sequence>
      <xs:element name="bytes-received" type="bytes-received" />
      <xs:element name="bytes-sent" type="bytes-sent" />
      <xs:element name="units-received" type="units-received" />
      <xs:element name="units-sent" type="units-sent" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="person">
```

```
    <xs:sequence>
      <xs:element name="person-first-name" type="person-first-name" />
      <xs:element name="person-last-name" type="person-last-name" />
      <xs:element name="person-middle-name" type="person-middle-name" minOccurs=
"0" maxOccurs="unbounded" />
      <xs:element name="phone-contact" type="phone-contact" minOccurs="0" maxOcc
urs="unbounded" />
      <xs:element name="email-address" type="email-address" minOccurs="0" maxOcc
urs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="person-first-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="person-last-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="person-middle-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="email-address">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="phone-contact">
    <xs:sequence>
      <xs:element name="phone-number" type="phone-number" />
      <xs:element name="phone-number-type" type="phone-number-type" minOccurs="0
" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="phone-number">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="phone-number-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="DSN" />
      <xs:enumeration value="Fax" />
      <xs:enumeration value="Home" />
      <xs:enumeration value="Mobile" />
      <xs:enumeration value="Pager" />
      <xs:enumeration value="Secure" />
      <xs:enumeration value="Unsecure" />
      <xs:enumeration value="Work" />
      <xs:enumeration value="Other" />
    </xs:restriction>
```

```
  </xs:simpleType>

  <xs:complexType name="privilege">
    <xs:sequence>
      <xs:element name="privilege-name" type="privilege-name" />
      <xs:element name="privilege-value" type="privilege-value" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="privilege-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="privilege-value">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="location">
    <xs:sequence>
      <xs:element name="WGS84-longitude" type="WGS84-longitude" />
      <xs:element name="WGS84-latitude" type="WGS84-latitude" />
      <xs:element name="WGS84-altitude" type="WGS84-altitude" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="WGS84-longitude">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="WGS84-latitude">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="WGS84-altitude">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="organization-id">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="organization-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:complexType name="organization">
    <xs:sequence>
      <xs:element name="organization-id" type="organization-id" />
```

```
      <xs:element name="organization-name" type="organization-name" />
      <xs:element name="location" type="location" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="data-source">
    <xs:sequence>
      <xs:element name="data-source-label" type="data-source-label" minOccurs="0
" />
      <xs:element name="endpoint-identifier" type="endpoint-identifier" minOccur
s="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="administrative-domain">
    <xs:sequence>
      <xs:element name="administrative-domain-label" type="administrative-domain
-label" />
      <xs:element name="sub-administrative-domain" type="sub-administrative-doma
in" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="super-administrative-domain" type="super-administrative-
domain" minOccurs="0" />
      <xs:element name="location" type="location" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="access-privilege-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="read" />
      <xs:enumeration value="write" />
      <xs:enumeration value="none" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="account-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="authenticator">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="authentication-type">
    <xs:restriction base="xs:string">
      <!-- To be done -->
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="birthdate">
    <xs:restriction base="xs:date" />
  </xs:simpleType>

  <xs:simpleType name="certificate">
```

```
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="collection-task-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="network-observation" />
      <xs:enumeration value="remote-acquisition" />
      <xs:enumeration value="self-reported" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="confidence">
    <xs:restriction base="xs:float">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="1" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="content-action">
    <xs:restriction base="xs:string">
      <xs:enumeration value="add" />
      <xs:enumeration value="delete" />
      <xs:enumeration value="update" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="country-code">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="default-depth">
    <xs:restriction base="xs:integer" />
  </xs:simpleType>

  <xs:simpleType name="discoverer">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="firmware-id">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="hardware-serial-number">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="location-name">
    <xs:restriction base="xs:string" />
```

```
  </xs:simpleType>

  <xs:simpleType name="method-label">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="method-repository">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="network-access-level-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="block" />
      <xs:enumeration value="quarantine" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="patch-id">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="patch-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <!-- FIXME: is this type appropriate? -->
  <xs:simpleType name="public-key">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="role-name">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="true" />
      <xs:enumeration value="false" />
      <xs:enumeration value="error" />
      <xs:enumeration value="unknown" />
      <xs:enumeration value="not applicable" />
      <xs:enumeration value="not evaluated" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="te-assessment-state">
    <xs:restriction base="xs:string">
      <xs:enumeration value="in-discovery" />
```

```
      <xs:enumeration value="discovered" />
      <xs:enumeration value="in-classification" />
      <xs:enumeration value="classified" />
      <xs:enumeration value="in-assessment" />
      <xs:enumeration value="assessed" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="timestamp">
    <xs:restriction base="xs:dateTime" />
  </xs:simpleType>

  <xs:simpleType name="web-site">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="port-id">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="atm-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="lowSpeed" />
      <xs:enumeration value="highSpeed" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="enet-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="enet" />
      <xs:enumeration value="1genet" />
      <xs:enumeration value="10genet" />
      <xs:enumeration value="100genet" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="wifi-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="11n" />
      <xs:enumeration value="11a" />
      <xs:enumeration value="11gb" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="virtual-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="virtual-1g" />
    </xs:restriction>
```

```
  </xs:simpleType>

  <xs:complexType name="port">
    <xs:sequence>
      <xs:element name="port-id" type="port-id"/>
      <xs:element name="atm-type" type="atm-type" maxOccurs="1" minOccurs="0" />
      <xs:element name="enet-type" type="enet-type" maxOccurs="1" minOccurs="0"
/>
      <xs:element name="wifi-type" type="wifi-type" maxOccurs="1" minOccurs="0"
/>
      <xs:element name="virtual-type" type="virtual-type" maxOccurs="1" minOccur
s="0" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="user-account">
    <xs:sequence>
      <xs:element name="user" type="user"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
<CODE ENDS>
```

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

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

Email: ncamwing@cisco.com

Network Working Group   Y. Dong
Internet-Draft   L. Xia
Intended status: Standards Track   Huawei
Expires: March 11, 2018   September 07, 2017

The Data Model of Network Infrastructure Device Control Plane Security Baseline
draft-dong-sacm-nid-cp-security-baseline-00

Abstract

This document is one of the companion documents which describes the control plane security baseline YANG output for network infrastructure devices. The other parts of the whole document series [I-D.ietf- xia-sacm-nid-dp-security-baseline], [I-D.ietf-lin-sacm-nid-mp-security-baseline], [I-D.ietf-xia-sacm-nid-app-infr-layers-security-baseline] cover other parts of the security baseline for network infrastructure device in data plane, management plane, application layer and infrastructure layer respectively.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Objective

Nowdays network infrastructure devices such as switches, routers, and firewalls are always under the attack of the well-known network security threats which are sammrized in [I-D.ietf-xia-sacm-dp-security-profile]. Hence it is significant to ensure that the devices in a specific network meet the minimal security requirements according to their intended functions. In this case, the concept of security baseline for the network infrastructure device has been proposed in the above mentioned draft [I-D.ietf-xia-sacm-dp-security-profile] as well. The security baseline refers to the basic and compulsory capabilities of identifying the possible threats and vulnerabilities in the device itself, and enfocing the security hardening measurement. And it could be set to benchmark the security posture of an individual network device.

Basically, the overall security baseline of a particular network infrastructure device can be designed and deployed into three different layers, namely the application layer, the network layer, and the infrastructure layer. Moreover, the network layer security baseline is further classified into data plane, control plane, and management plane. In this document, we focus on the designation of data model for control plane security baseline while the security baseline of other layers and planes are proposed in the companion documents.

The control plane security basedline focus on the control signaling security of the network infrastructure device. The aim is to protect the normal information exchange between devices against various attcks (i.e. eavesdropping, tampering, spoofing and flooding attack) and restrict the malicious control signaling, for ensuring the correct network topology and forwading behavior.

1.2. Security Baseline Data Model Design

The security baseline of a certain device is dependent on many factors including but not limited to the different device types (i.e., router, switch, firewall) and their corresponding security features supported, and the specific security requirements of network operators. Owning to such a number of variations, it is impossible to design a comprehensive set of baseline for all devices. This document and the companion ones are going to propose the most important and universal points of them. More points can be added in future following the data model scheme specified in this document.

[I-D.ietf-birkholz-sacm-yang-content] defines a method of constructing the YANG data model scheme for the security posture assessment of the network infrastructure device by brokering of YANG push telemetry via SACM statements. The basic steps are:

- use YANG push mechanism[I-D.ietf-netconf-yang-push]to collect the created streams of notifications (telemetry) [I-D.ietf-netconf-subscribed-notifications]providing SACM content on SACM data plane, and the filter expressions used in the context of YANG subscriptions constitute SACM content that is imperative guidance consumed by SACM components on SACM management plane;

- then encapsulate the above YANG push output into a SACM Content Element envelope, which is again encapsulated in a SACM statement envelope;

- lastly, publish the SACM statement into a SACM domain via xmpp-grid publisher.

In this document, we follow the same way as [I-D.ietf-birkholz-sacm-yang-content] to define the YANG output for network infrastructure device security baseline posture based on the SACM information model definition [I-D.ietf-sacm-information-model].

1.3. Summary

The following contents propose part of the security baseline YANG output for network infrastructure device: control plane security baseline. The companion documents [I-D.ietf- xia-sacm-nid-dp-security-baseline], [I-D.ietf-lin-sacm-nid-mp-security-baseline], [I-D.ietf-xia-sacm-nid-app-infr-layers-security-baseline] cover other parts of the security baseline YANG output for network infrastructure device respectively: control plane security baseline, management plane security baseline, application layer and infrastructure layer security baseline.

2. Terminology

2.1. Key Words

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

2.2. Definition of Terms

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

3. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. Data Model Structure

A large amount of control protocols such as the typical TCP/IP stack and BGP in the control plane of network infrastructure device provide many operational services (i.e. farwording behavior control). These control protocols could be either the target under attack or the medium to attack the devices. The security baseline of several widely used protocols are specified in this section.

4.1. BGP

In a BGP network, TCP is always selected as the transport layer protocol. Thus it always subject to most of the attacks that targeting TCP-based protocols. In order to secure the BGP network, three types of functions, namely the GTSM, the RPKI, and the BGP peer connection authentication, could be configured in network device. This section specifies the authentication and RPKI configurations. The GTSM is summarized in another individual section together with some other protocols that all supports GTSM.

Various kinds of authentication techniques are able to be used for securing the TCP connections between BGP neighbors. They only allows the authorized peers to establish neighbor relationship with local device so that the information exchanged between the BGP neighbors via the TCP connection cannot be altered.

The Resource Public Key Infrastructure (RPKI) is usually applied in a network equipt with a RPKI server to secure the inter-domain BGP routing. The device is required to establish a connection to the RPKI server and then downloads or updates the Route Origin Authorizations (ROAs), which links certain IP prefixes or prefix range with an autonomous system (AS), from the RPKI server. After that, the received BGP route information is validated against the downloaded/updated ROAs to verify whether the BGP prefixe originates from the expected AS.

```
module: bgp-sec-config
    +--rw bgp-rpki
    |  +--rw bgp-rpki-session-config* [session-ipv4-addr]
    |  |  +--rw session-ipv4-addr        ipv4-address
    |  |  +--rw port-number              unit16
    |  |  +--rw cipher-password?         string
    |  |  +--rw aging-time?              unit32
    |  |  +--rw refresh-time?            unit16
    |  |  +--rw rpki-limit?
```

```
   |  |     +--rw limit                 unit32
   |  |     +--rw (action-type)
   |  |        +--:(alert)
   |  |        |  +--rw enable          boolean
   |  |        +--:(idle-forever)
   |  |        |  +--rw enable          boolean
   |  |        +--:(idle-timeout)
   |  |           +--rw timeout         unit16
   |  +--rw origin-as-validate-utilization
   |     +--rw origin-validation-enable boolean
   |     +--rw origin-as-validate       boolean
   |     +--rw allow-invalide           boolean
   |     +--rw (peer-identification-method)
   |        +--:(group)
   |        |  +--rw peer-group* [group-name]
   |        |     +--rw group-name      string
   |        |     +--rw advertise-enable boolean
   |        +--:(ip)
   |           +--rw peer-ip* [ipv4-addr]
   |              +--rw ipv4-addr     ipv4-address
   |              +--rw advertise-enable boolean
   +--rw bgp-authentication* [bgp-as-number]
      +--rw bgp-as-number             unit16
      +--rw (peer-identification-method)
         +--:(group)
         |  +--rw peer-group* [group-name]
         |     +--rw group-name       string
         |     +--rw (authentication-method)
         |        +--:(md5)
         |        |  +--rw password-type:{plain|cipher} enumeration
         |        |  +--rw password-text  string
         |        +--:(keychain)
         |           +--rw keychain-name
         +--:(ip)
            +--rw peer-ip* [ipv4-addr]
               +--rw ipv4-addr        inet-type:ipv4-address
               +--rw (authentication-method)
                  +--:(md5)
                  |  +--rw password-type:{plain|cipher} enumeration
                  |  +--rw password-text  string
                  +--:(keychain)
                     +--rw keychain-name  string
```

4.2. OSPF

There are a number of ways for spoofing procotol packet to attack OSPF protocol. One possible scenario is that the rogue device inject manipulated routing information to cause a Denial-of-Service attack.

Authentication has been demonstrated as a powerful tool to identify and drop these spoofing packets to protect OSPF protocol and secure the connection between the OSPF neighbors. A widely range of authentication methods can be deployed in a network device such as MD5, HAMC-MD5, and keychain. As shown in the following tree diagram, the authentication can be deployed in either area or interface basis.

```
 module:ospf-sec-config
     +--rw ospf-authentication
        +--rw area-authentication* [area-id]
        |  +--rw area-id              unit16
        |  +--rw (authentication-method)
        |     +--:(simple-authen)
        |     |  +--rw password-type:{plain|cipher} enumeration
        |     |  +--rw password-text  string
        |     +--:(md5-hmac-authen)
        |     |  +--rw sub-mode:
        |     |  |       {md5|hmac-md5|hmac-sha256} enumeration
        |     |  +--rw password-type   enumeration
        |     |  +--rw password-text   string
        |     +--:(keychain-authen)
        |        +--rw keychain-name   string
        +--rw interface-authentication* [interface-number]
           +--rw interface-type        enumeration
           +--rw interface-number      unit8
           +--rw (authentication-method)
              +--:(simple-authen)
              |  +--rw password-type   enumeration
              |  +--rw password-text   string
              +--:(md5-hmac-authen)
              |  +--rw sub-mode        enumeration
              |  +--rw password-type   enumeration
              |  +--rw password-text   string
              +--:(keychain-authen)
                 +--rw keychain-name   string
```

4.3. IS-IS

IS-IS optional checksum function adds the a checksum TLV in SNP and hello packet. The device firstly check the correctness of checksum TVL when it receive the packet. It secure the data in data link layer.

IS-IS authentication encapsulate the authentication information in hello packet, LSP packet, and SNP packet. Only the packets passed the verification will be further processed. The IS-IS authentication is mainly used to secure packet in network layer.

```
module:isis-sec-config
    +--rw isis-optional-checksum
    |  +--rw enable                     boolean
    +--rw isis-authentication
       +--rw area-authentication* [process-id]
       |  +--rw process-id          unit32
       |  +--rw (authentication-method)
       |     +--:(simple)
       |     |  +--rw authen-password-mode:{op|osi} enumeration
       |     |  +--rw password-type:{plain|cipher}  enumeration
       |     |  +--rw password-text                 string
       |     +--:(md5)
       |     |  +--rw authen-password-mode:{op|osi} enumeration
       |     |  +--rw password-type:{plain|cipher}  enumeration
       |     |  +--rw password-text                 string
       |     +--:(keychain)
       |     |  +--rw keychain-name                 string
       |     +--:(hmac-sha256)
       |     |  +--rw key-id                        unit16
       |     |  +--rw password-type:{plain|cipher}  enumeration
       |     |  +--rw password-text                 string
       |     +--rw snp-packet:
       |     |      {authentication-avoid|send-only} enumeration
       |     +--rw all-send-only?                   boolean
       +--rw domain-authentication* [process-id]
       |  +--rw process-id                       unit32
       |  +--rw (authentication-method)
       |     +--:(simple)
       |     |  +--rw authen-password-mode:{op|osi} enumeration
       |     |  +--rw password-type:{plain|cipher}  enumeration
       |     |  +--rw password-text                 string
       |     +--:(md5)
       |     |  +--rw authen-password-mode:{op|osi} enumeration
       |     |  +--rw password-type:{plain|cipher}  enumeration
       |     |  +--rw password-text                 string
       |     +--:(keychain)
       |     |  +--rw keychain-name                 string
       |     +--:(hmac-sha256)
       |     |  +--rw key-id                        unit16
       |     |  +--rw password-type:{plain|cipher}  enumeration
       |     |  +--rw password-text                 string
       |     +--rw snp-packet                       enumeration
       |     +--rw all-send-only?                   boolean
       +--rw interface-authentication* [interface-number]
          +--rw interface-type                   enumeration
          +--rw interface-number                 pub-type:ifNum
          +--rw (authentication-method)
             +--:(simple)
```

```
                 |  +--rw authen-password-mode:{op|osi} enumeration
                 |  +--rw password-type:{plain|cipher}  enumeration
                 |  +--rw password-text                 string
                 +--:(md5)
                 |  +--rw authen-password-mode:{op|osi} enumeration
                 |  +--rw password-type:{plain|cipher}  enumeration
                 |  +--rw password-text                 string
                 +--:(keychain)
                 |  +--rw keychain-name                 string
                 +--:(hmac-sha256)
                 |  +--rw key-id                        unit16
                 |  +--rw password-type:{plain|cipher}  enumeration
                 |  +--rw password-text                 string
                 +--rw authen-level?:{level1|level2} enumeration
                 +--rw send-only?                    boolean
```

4.4. MPLS

RSVP authentication is suggested to configure in the device in order to improve the network security and protect the local device against the malicious attack. It prevent the establishment of illegal RSVP peer connection in the following situation

The peer was unauthorized to establish connection with local device;

The attacker establish connection with locol device via spoofing RSVP packet.

Furthermore, it introduce a few enhancement to verify the lifetime, handshake and message window size for protection of RSVP against the playback attack and the termination of authentication relationships caused by packet out of order problem.

As shown in the tree diagram, the LDP also support MD5 and keychain authentication.

```
  module:mpls-sec-config
      +--rw rsvp-sec-config
      |  +--rw rsvp-authentication
      |     +--rw interface-authentication
      |     |  +--rw interface-authen* [interface-number]
      |     |     +--rw interface-type                  enumeration
      |     |     +--rw interface-number                pub-type:ifNum
      |     |     +--rw (authentication-method)
      |     |        +--:(md5)
      |     |        |  +--rw password-type:{plain|cipher} enumeration
      |     |        |  +--rw password-text                string
```

```
|     |        +--:(keychain)
|     |        |  +--rw keychain-name                string
|     |        +--rw life-time?              yang-type:timestamp
|     |        +--rw handshake-enable?               boolean
|     |        +--rw window-size?                    unit8
|     +--rw peer-authentication
|        +--rw peer-authen* [peer-addr]
|           +--rw peer-addr                   inet-type:ip-address
|           +--rw (authentication-method)
|              +--:(md5)
|              |  +--rw password-type:{plain|cipher} enumeration
|              |  +--rw password-text                string
|              +--:(keychain)
|              |  +--rw keychain-name                string
|              +--rw challenge-maximum-miss-times?   unit8
|              +--rw challenge-retrans-interval?     unit16
|              +--rw life-time?              yang-type:timestamp
|              +--rw handshake-enable?               boolean
|              +--rw window-size?                    unit8
+--rw ldp-sec-config
   +--rw ldp-authentication
   +--rw (authentication-method)
      +--:(keychain)
      |  +--rw (authen-object)
      |     +--:(peer-single)
      |     |  +--rw single-peer-authen* [peer-id]
      |     |     +--rw peer-id                 dotted decimal
      |     |     +--rw keychain-name               string
      |     +--:(peer-group)
      |     |  +--rw group-peer-authen* [ip-prefix-name]
      |     |     +--rw ip-prefix-name              string
      |     |     +--rw keychain-name               string
      |     +--:(peer-all)
      |        +--rw keychain-name                  string
      |        +--rw exclude-peer-id?           dotted decimal
      +--:(md5)
         +--rw (authen-object)
            +--:(peer-single)
            |  +--rw single-peer-authen* [peer-lsr-id]
            |     +--rw peer-lsr-id              dotted decimal
            |     +--rw password-type:{plain|cipher} enumeration
            |     +--rw password-text              string
            +--:(peer-group)
            |  +--rw group-peer-authen* [ip-prefix-name]
            |     +--rw ip-prefix-name             string
            |     +--rw password-type:{plain|cipher} enumeration
            |     +--rw password-text              string
            +--:(peer-all)
```

```
                     +--rw password-type:{plain|cipher}  enumeration
                     +--rw password-text                 string
```

4.5. Keychain

Authentication is a widely used technique to ensure the packet information are not been changed/altered by attackers. It requires the information sender and receiver to share the authentication information including the key and algorithm. In addition, the key pairs cannot be delivered in the network (symmetric). However, in order to improve the its reliability, the encryption algorithm and the keys have to be renewed dynamically. It is a complicated and time consuming process to change the keys and algorithm for all the used protocols manually. The keychain provide an solution to renew the authentication keys and algorithm periodically in a dynamic fashion.

```
module:keychain-config
    +--rw keychain-config* [keychain-name]
       +--rw keychain-name               string
       +--rw keychain-mode:
       |     {absolute|periodic|daily|
       |      weekly|monthly|yearly}    enumeration
       +--rw receive-tolerance?
       |  +--:(finite)
       |  |  +--rw tolerance-value               unit16
       |  +--:(infinite)
       |     +--rw infinite-enable               boolean
       +--rw time-mode:{utc|lmt}                 enumeration
       +--rw digest-length?                      boolean
       +--rw keychain-id* [key-id]
          +--rw key-id                           unit8
          +--rw keychain-string-type:{plain|cipher} enumeration
          +--rw keychain-string-text             string
          +--rw keychain-algorithm:
          |    {hmac-md5|hmac-sha-256|hmac-sha1_12|
          |     hmac-sha1_20|md5|sha-1|sha-256} enumeration
          +--rw default-key-id?                   unit8
          +--rw (send-time-mode)
          |  +--:(absolute)
          |  |  +--rw start-time            yang-type:timestamp
          |  |  +--rw start-date         yang-type:date-and-time
          |  |     +--rw (count-type)
          |  |        +--:(duration)
          |  |        |  +--rw (finite-or-infinite)
          |  |        |     +--:(finite)
          |  |        |     |  +--rw duration-value  unit32
          |  |        |     +--:(infinite)
```

```
|  |        |        +--rw infinite-enable boolean
|  |        +--:(end)
|  |           +--rw end-time    yang-type:timestamp
|  |           +--rw end-date    yang-type:date-and-time
|  +--:(periodic-daily)
|  |  +--rw start-time           yang-type:timestamp
|  |  +--rw end-time             yang-type:timestamp
|  +--:(periodic-weekly)
|  |  +--rw (count-type)
|  |     +--:(continues)
|  |     |  +--rw start-day-name  enumeration
|  |     |  +--rw end-day-name    enumeration
|  |     +--:(discrete)
|  |        +--rw day-name*       enumeration
|  +--:(periodic-montly)
|  |  +--rw (count-type)
|  |     +--:(continues)
|  |     |  +--rw start-date    yang-type:date-and-time
|  |     |  +--rw end-date      yang-type:date-and-time
|  |     +--:(discrete)
|  |        +--rw date*         yang-type:date-and-time
|  +--:(periodic-yearly)
|     +--rw (count-type)
|        +--:(continues)
|        |  +--rw start-month     enumeration
|        |  +--rw end-month       enumeration
|        +--:(discrete)
|           +--rw month*          enumeration
+--rw (receive-time-mode)
   +--:(absolute)
   |  +--rw start-time            yang-type:timestamp
   |  +--rw start-date         yang-type:date-and-time
   |     +--rw (count-type)
   |        +--:(duration)
   |        |  +--rw (finite-or-infinite)
   |        |     +--:(finite)
   |        |     |  +--rw duration-value  unit32
   |        |     +--:(infinite)
   |        |        +--rw infinite-enable boolean
   |        +--:(end)
   |           +--rw end-time    yang-type:timestamp
   |           +--rw end-date  yang-type:date-and-time
   +--:(periodic-daily)
   |  +--rw start-time         yang-type:timestamp
   |  +--rw end-time           yang-type:timestamp
   +--:(periodic-weekly)
   |  +--rw (count-type)
   |     +--:(continues)
```

```
|     |  +--rw start-day-name   enumeration
|     |  +--rw end-day-name     enumeration
|     +--:(discrete)
|        +--rw day-name*        enumeration
+--:(periodic-montly)
|  +--rw (count-type)
|     +--:(continues)
|     |  +--rw start-date    yang-type:date-and-time
|     |  +--rw end-date      yang-type:date-and-time
|     +--:(discrete)
|        +--rw date*         yang-type:date-and-type
+--:(periodic-yearly)
   +--rw (count-type)
      +--:(continues)
      |  +--rw start-month    enumeration
      |  +--rw end-month      enumeration
      +--:(discrete)
         +--rw month*         enumeration
```

4.6. GTSM

Attackers send a large amount of forging packets to a target network device. Then the forging packets are delivered to the cpu straigtforward when the destinations are correctly checked. The CPU will be overloaded owning to processing such a number of protocol packets. In order to protect the CPU against the CPU utilization attack, a GTSM (generized TTL security mechanism) function is configured to check the TTL (time to live) in the IP head. The packets will send to cpu for further processing only if the TTL number is whithin a pre-defined range.

As shown in the three diagram in the following figure, the GTSM function is configured separately for individual procotols. Each of the protocols, even each list instances in a protocol, has its own pre-defined TTL range.

```
module:gtsm
    +--rw gtsm-config
       +--rw default-gtsm-action:{drop|pass}
       +--rw bgp-gtsm* [bgp-as-number]
       |  +--rw bgp-as-number             unit32
       |  +--rw (peer-identification-method)
       |     +--:(group)
       |     |  +--rw peer-group* [group-name]
       |     |     +--rw group-name       string
       |     |     +--rw valid-ttl-hops   unit16
       |     +--:(ip)
       |        +--rw peer-ip*  [ipv4-addr]
       |           +--rw ipv4-addr   inet-type:ipv4-address
       |           +--rw valid-ttl-hops   unit8
       +--rw ospf-gtsm* [vpn-instance-name]
       |  +--rw vpn-instance-name       string
       |  +--rw valid-ttl-hops           unit16
       +--rw mpls-ldp-gtsm* [peer-ip-addr]
       |  +--rw peer-ip-addr          inet-type:ip-address
       |  +--rw valid-ttl-hops            unit16
       +--rw rip-gtsm* [vpn-instance-name]
          +--rw vpn-instance-name?      string
          +--rw valid-ttl-hops          unit16
```

5. Network Infrastructure Device Security Baseline Yang Module

TBD

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

TBD.

8. Acknowledgements

TBD

9. References

9.1. 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>.

9.2. Informative References

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Notifications", draft-ietf-netconf-subscribed-notifications-03 (work in progress), July 2017.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-08 (work in progress), August 2017.

[I-D.ietf-sacm-information-model]
Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus, M., Haynes, D., and H. Birkholz, "SACM Information Model", draft-ietf-sacm-information-model-10 (work in progress), April 2017.

Authors' Addresses

Yue Dong
Huawei

Email: dongyue6@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com

Network Working Group   Y. Dong
Internet-Draft   L. Xia
Intended status: Standards Track   Huawei
Expires: November 26, 2018   May 25, 2018

The Data Model of Network Infrastructure Device Infrastructure Layer Security Baseline

draft-dong-sacm-nid-infra-security-baseline-01

Abstract

This document is one of the companion documents which describes the infrastructure layer security baseline YANG output for network infrastructure devices. The infrastructure layer security baseline covers the security functions to secure the device itself, and the fundamental security capabilities provided by the device to the upper layer applications. In this specific document, the integrity measurement, cryptography algorithms, key management, and certificate management are sorted out to generate the data model.

Table of Contents

1. Introduction

Network devices such as switches, routers, and firewalls are the fundamental elements that a network is composed of. The vulnerabilities of a network device are always exploited by attackers to start up eavesdropping, spoofing, and man-in-middle attacks etc. Hence it is significant to assess the security postures for identifying the possible threats and vulnerabilities of a network device in anytime. The SACM working group is aim to provide such a mechanism to acquire the posture information, which including the security related configuration and status attributes, on the target devices and evaluate their security postures by comparing with the pre-defined benchmarking criteria. Furthermore, the evaluation results are able to be the guidance to enforce the corresponding security hardening measurement on the devices under assessment. But this hardening process is out of scope of this draft.

This draft and each of the companion document define a subset of posture information that have to be collected for the assessment purpose mentioned above. This entire set of posture information is so called security baseline of a network device that is proposed in the companion draft [I-D.draft-xia-sacm-dp-security-profile]. The proposed security baseline is presented in the fashion of yang data model. And the security baseline yang data model can be requested or subscribed by a collector agent such as a yang push client [draft-birkholz-sacm-yang-content]. The output of such a collector agent is then encapsulated into the SACM content and statement elements [draft-ietf-sacm-information-mdoel] and published to other SACM components (e.g. repository and evaluator) [draft-mandm-sacm-architecture-01]. Please note that document is only focus on the yang data model of security baseline, the messaging mechanisms is out of scope of this document. They are specified in other documents.

1.1. Infrastructure layer security baseline

In general, the entire security baseline of a network device is divided into three layers, namely the application layer, the network layer, and the infrastructure layer. This document focus on the data model on infrastructure layer. The infrastructure layer security baseline herein refers to the configuration and status attributes of security functions that secure the device itself, and the fundamental security capabilities provided by the device to the upper layer applications. More specifically, the essential configurable and key status attributes of the following function/capability modules are sorted out to generate the infrastructure layer security baseline data model.

o Integrity measurement: the integrity measurement herein refers to the functions such as trust computing to protect the device against the replacement and/or tampering attacks. For example, the trust boot and/or secure boot provide the integrity validation service for the kernel and early stage executable code (bios and bootloader) in bootstrapping phases, and the digital signature protect the upper layer software applications against the tampering attacks in software updating phases.

- o Cryptography algorithms: the cryptographic algorithms are the most important capabilities that the device provides to the upper layer security applications. For example, the symmetric (e.g. DES, AES) and asymmetric (eg. RSA, ECC) cryptographic algorithms can be used for sensitive data encryption, and peers authentication. And the key derivation function (KDF) can be used for secret key generation and passcode storage.

- o Key management: the cryptographic key (pair) and its associated algorithm provide various security features for network devices. How we manage the key (pair) provisioned in a network device is a critical issue. The key management covers the attributes to show how the key (pair) is managed in the key’s lifecycle (e.g. from generation to destroy).

- o Certificate management: the certificates are normally provided by the device for authentication purpose. The certificate management refers to how the certificates and the certificates revocation list (CRL) is requested, updated, and validated in the device.

The practical security baseline of a network device depends on the device type, the supported features, the requirements of operators and enterprises, and the role it plays exactly in the network. Owing to such a number of variance, it is impossible to design a comprehensive and unified data model for all devices. Thus the proposed data model in this document is only used to benchmark the most widely deployed security related functions and capabilities. And we would like it to be an extensible model so that more attributes are able to be added as per the practical use case scenario.

2. Terminology

2.1. Key Words

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

2.2. Definition of Terms

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

3. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

o Brackets "[" and "]" enclose list keys.

o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".

o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. Data Model Structure

As mentioned above, the top-level structure of the data model is shown in the following figure. There are four subtrees in the tree diagram. Each of the following sub-sections specifies the detail of an individual subtree.

```
  module: infrastructure-layer-baseline
      +--rw infrastructure-layer-baseline
         +--rw integrity-measurement
         |  . . . . . .
         +--rw cryptography-algorithms
         |  . . . . . .
         +--rw key-management
         |  . . . . . .
         +--rw certificate-management
            . . . . . .
```

4.1. Integrity measurement

The purpose of integrity measurement is to prevent the upper layer software applications, kernel, and early stage executable code (e.g. BIOS and bootloader) from replacement and/or tampering in bootstrapping and updating phases. Trusted boot and secure boot are the two widely used techniques for protecting the device bootstrapping. The read-only root of trust (RoT) should be always stored in a SoC or TPM chip. For software updating, digital signature has been demonstrated as a powerful tool to provide the integrity protection service. In using digital signature, the employed hash function and signature algorithm must be strong enough so that attackers cannot force crack them in a short period of time. Moreover, the public key used for verifying the signature should be stored properly. For example, it can be wrapped in a certificate of the software vendor or stored in the read-only SoC or TPM.

```
     module: integrity-measurement
         +--rw integrity-measurement
            +--rw bootstrapping
            |  +--rw trust-boot
            |  |  +--ro tmp-version              string
            |  |  +--rw tpm-enable               boolean
            |  |  +---u hash-function
            |  |  +--rw pcr-record*  [pcr-number]
            |  |     +--ro pcr-number            unit8
            |  |     +--ro measurement-item      enumeration
            |  |     +--ro pcr-value             string
            |  |     +--ro pcr-benchmark-value   string
            |  |     +--ro verify-result         boolean
            |  +--rw secure-boot
            |     +--ro soc-model                string
            |     +--ro measurement-item*        enumeration
            |     +---u hash-function
            |     +---u signature-algorithm
            |     +--ro verification-public-key
            |        +--ro key-name              string
            |        +--ro key-length            unit16
            |        +--ro key-store-medium      enumeration
            +--rw software-update
               +---u hash-function
               +---u signature-algorithm
               +--ro verification-public-key
                  +--ro key-name                 string
                  +--ro key-length               unit16
                  +--ro key-store-medium         enumeration
```

4.2. Cryptography security

Almost all the security features of communication network are built on the basis of modern cryptography. For example, the cryptographic algorithms are usually used to perform transmission data encryption and peers authentication. However, as the computing capability of the present computing system is getting faster and faster, more and more cryptographic algorithms can be brute force cracked in a short period of time. Therefore the algorithm has to be selected appropriately for different use case scenarios. And the configuration parameters must be set within an appropriate range so that the used algorithm is strong enough.

As a fundamental capabilities provided by the device, the practical configurations of each supported cryptographic algorithm varies as per the upper layer application that employs the algorithm. This section organizes the algorithms and their configuration parameters into groupings so that the upper layer applications can reference/ reuse them appropriately.

In general, this section covers the following cryptographic algorithm groupings:

- o Symmetric algorithms and their configurable parameters.

- o Asymmetric algorithms and their configurable parameters.

- o Hash functions.

- o Message authentication code (MAC) methods and their configurable parameters.

- o Key derivation functions (KDF) and their configurable parameters.

All the groupings enable the collection of the specific algorithms and their parameters on a case-by-case basis.

4.2.1. Symmetrical cryptography

The symmetric algorithms are typically used for providing data confidential service. The encryption and decryption process of symmetrical algorithms make use of two identical keys. And, most of the symmetrical algorithms are typically belong to either block ciphers or stream ciphers.

Block cipher: block cipher divides the plaintext in to a number of blocks with a constant bit length. And the last plaintext block should be filled to fit the bit length requirement. Then each of the plaintext blocks is encrypted individually. However, if a plaintext piece repeats several times in a long data stream, it is easier for an attacker to guess the original plaintext from the repeated ciphertext. Hence, some other operation modes of block cipher, including cipher block chaining (CBC) mode, cipher feedback (CFB) mode, counter mode (CRT), and Galois counter mode (GCM), are proposed to introduce a random bit stream, which is named initialization vector (IV), to augment the randomness of the original plaintext. The used random number generator must meet the randomness requirement so that the IV value is unpredicted. In addition, the bit length of IV should be the same as the bit length of a plaintext block for most block cipher working mode. But for CRT and GCM, the length of IV is optional.

Stream cipher: unlike block cipher, which encrypt a single plaintext block at one time, stream-cipher encrypt every bit of a plaintext separately. The stream cipher algorithms also use IV to increase the randomness of the original plaintext.

```
    grouping: symmetric-cryptosystem
          +--rw (algorithm-type)
             +--:(stream-cipher)
             |  +--rw algorithm         identityref
             |  +--rw iv-length         unit16
             |  +--rw iv-randomness     decimal64
             +--:(block-cipher)
                +--rw algorithm         identityref
                +--rw operation-mode    identityref
                +--rw padding-method    identityref
                +--rw iv-length         unit16
                +--rw iv-randomness     decimal64
```

4.2.2. Asymmetrical cryptography

The asymmetric cryptography is also called public key cryptography. In contrast to the symmetric one, asymmetric cryptography always employs a key pair that contains two different keys to deal with the encryption and decryption work. The private key in the key pairs is held and used only by the owner. The other key in the key pairs is theoretically public to everyone. The asymmetric cryptography algorithms are not only able to provide data encryption, but also provide authentication and/or integrity protection services (e.g. digital signature).

Asymmetric encryption: RSA is the most commonly used asymmetrical encryption algorithm. In the use of RSA, the smaller the public exponent is, the higher efficiency the algorithm has. In the other side, it will be much easier to crack the algorithm and recover the original plaintext if the public exponent is too small. Hence it has to trade off the value of public exponent. In addition, the RSA is recommend to use optimal asymmetrical encryption padding (OAEP) to fill up the original plaintext.

```
  grouping: encryption-algorithm
        +--rw encryption-algorithm
           +--rw rsa-attributes
              +--rw algorithm              identityref
              +--rw padding-method         identityref
              +--rw public-key
                 +--rw public-exponent     unit32
                 +--rw modulo-value        unit32
```

Digital signature: digital signature is a powerful tool to provide integrity protection. DSA, RSA, and ECDSA are three of the most popular signature algorithms. By using RSA in digital signature, it is better to use PSS for padding. If the data is required to be encrypted and signed at the same time, it is suggest to sign the data before encrypting.

```
  grouping: signature-algorithms
        +--rw (asymmetric-algorithms)
           +--:(rsa)
           |  +--rw algorithm                identityref
           |  +--rw padding-method           identityref
           |  +--rw public-key
           |     +--rw public-exponent       unit32
           |     +--rw modulo-value          unit32
           +--:(dsa)
           |  +--rw temporary-key
           |  |  +--rw key-length            unit16
           |  |  +--rw randomness            decimal64
           |  +--rw prime-number
           |     +--rw prime-modulo          unit32
           |     +--rw prime-order           unit32
           +--:(ecdsa)
              +--rw temporary-key
              |  +--rw key-length            unit16
              |  +--rw randomness            decimal64
              +---u hash-function
              +--rw prime-modulo             unit32
              +--rw prime-order              unit32
              +--rw ec-parameters
                 +--rw coefficient-a         unit16
                 +--rw coefficient-b         unit16
```

Key exchange: key exchange is meant to establish key pairs between communication peers. The peers send key material rather than key itself to each other.

```
      grouping: key-exchange
            +--rw (key-exchange)
               +--:(dh)
               |  +--rw dh-handshake
               |     +--rw prime-number-length    unit32
               |     +--rw public-integer-length  unit32
               +--:(ecdh)
                  +--rw ecdh-handshake
                     +--rw prime-modulo           unit32
                     +--rw ec-parameters
                     |  +--rw coefficient-a       unit16
                     |  +--rw coefficient-b       unit16
                     +--rw primitive-elements
                        +--rw coordinate-x        unit16
                        +--rw coordinate-y        unit16
```

4.2.3. Hash function

Hash functions are normally used to perform integrity measurement. The output of a Hash function is a digest with a constant bit length for a segment of messages or code. The digest is unique and unable to be reconstructed if the original message/code is tampered. The Hash function is widely used in digital signature, message authentication code, password hash storage, and etc.

```
     grouping: hash-function
            +--rw hash-function
               +--rw algorithm             identityref
               +--rw padding-method        identityref
               +--ro digest-length         unit16
```

4.2.4. Message authentication code

Similar to digital signature, message authentication code (MAC) is another method to provide integrity protection service. MAC applies hash function or block cipher algorithms on the message plaintext coupled with a pre-shared session key. It must be noted that, it is unsafe if simply extend the message with the session key.

```
     grouping: message-authentication-code
            +--rw (message-authentication-code)
               +--: (hmac)
               |  +--rw message-structure    enumeration
               |  |    {prefix|postfix|hmac structure}
               |  +---u hash-function
               |  +--rw session-key
               |     +--rw key-length        unit16
               |     +--rw randomness        decimal64
               +--: (cmac)
                  +--rw block-cipher-algorithm      identityref
                  +--rw block-length                unit16
                  +--rw iv-length                   unit16
                  +--rw randomness                  decimal64
```

4.2.5. Key derivation function

Key derivation function derives one or more keys from a master key or entered password. A salt value is generated by a random number generator to introduce the randomness of the derived keys.

```
     grouping: key-derivation-function
            +--rw (algorithm)
               +--:(pbkdf2)
               |  +---u hash-function
               |  +--rw iteration                unit16
               |  +--rw derived-key-length       unit16
               |  +--rw code-length              unit16
               |  +--rw salt-attributes
               |     +--rw salt-length           unit16
               |     +--rw randomness            decimal64
               +--:(scrypt)
                  +--rw code-length              unit16
                  +--rw cpu-memory-usage         unit16
                  +--rw block-size               unit8
                  +--rw parallelization          unit8
                  +--rw derived-key-length       unit16
                  +--rw salt-attributes
                     +--rw salt-length           unit16
                     +--rw randomness            decimal64
```

4.3. Key management

Cryptographic key plays the most important role in a cryptographic system. . If the key is disclosed or tampered, the corresponding service is not reliable any more. Hence the network device must provide the confidentiality and integrity protection for a key in its entire lifecycle. This section contains a list of key (pair) and their configuration/status parameters corresponding to different lifecycle phases. Each of the key (pair) is used in a specific use case.

```
  module: key-management
      +--rw key-management* [key-name]
         +--rw key-name             string
         +--rw key-length*          unit16
         +--rw lifetime             unit32
         +--rw key-type             enumeration
         +--rw num-of-keys          unit8
         +--rw key-generation
         | . . . . . .
         +--rw key-distribution
         | . . . . . .
         +--rw key-store
         | . . . . . .
         +--rw key-backup
         | . . . . . .
         +--rw key-update
         | . . . . . .
         +--rw key-destroy
           . . . . . .
```

4.3.1. Key generation

There are three types of commonly used key generation methods. The first method is on the basis of random number generator. In this method, the referenced random number generator has to ensure the generated key is unpredicted. The second key generation method is based on the manual entered password. However, the entered password is not meet the randomness requirement. In this case, a key derivation function (e.g. PBKDF2) is applied to derive the key. The last key generation method is key exchange such as Diffie-Hellman (DH) protocol. This kind of method requires the peers to authenticate each other before exchange the key material.

```
  submodule: key-generation
         +--rw key-generation
            +--: (random-number-generator)
            |  +--rw key-randomness           decimal64
            +--: (key-derivation-function)
            |  +---u key-derivation-function
            +--: (key-exchange)
               +--rw cert-name                 string
               +---u key-exchange
```

4.3.2. Key distribution

Key distribution aims to send the generated keys to authorized entities in a secure fashion. The confidentiality and integrity issues of the key in distribution are usually addressed by using either a secure transport protocol or digital envelop. [I-D.ietf-netconf-tls-client-server], IPsec [I-D.draft-tran-ipsecme-yang], or SSH [I-D.ietf-netconf-ssh-client-server], or digital envelop.

```
submodule: key-distribution
       +--rw key-distribution?
          +--rw symmetrical-key
             +--: (secure-transport-protocol)
             |  +--rw tls-config
             |  |   [I-D.ietf-netconf-tls-client-server]
             |  +--rw ipsec-config
             |  |   [I-D.draft-tran-ipsecme-yang]
             |  +--rw ssh-config
             |      [I-D.ietf-netconf-ssh-client-server]
             +--: (digital-envolop)
                +---u symmetric-algorithm
                +--rw encryption-key-name     string
                +--rw encryption-key-length   unit16
```

4.3.3. Key store

A typical key management system has three layers. The master keys that consumed by upper layer applications are in the top layer. The key in the middle layer, which is called key encryption key (KEK), is used to encrypt the master keys. And the KEK itself is encrypted by the root key which stays in the bottom layer of the three layer key management system.

```
submodule: key-store
       +--rw key-store
          +--ro store-medium {TPM|HSM|HDD}      enumeration
          +--rw key-component* [component-name]
             +--rw component-name               string
             +--ro store-medium                 enumeration
```

4.3.4. Key update

Network device must update the key in a reasonable period of time. Otherwise the long term used key will attract attackers to crack it. The practical update period of a certain key depends on the application the key serves and the strength (i.e. bit length) of the key itself.

```
submodule: key-update
       +--rw key-update
          +--rw next-update-time       yang-type:date-and-time
          +--rw hold-expired-key       boolean
          +--rw update-mode
             +--: (manual)
             |  +--rw manual-enable    boolean
             +--: (auto)
                +--rw auto-enble       boolean
                +--rw update-period    unit8
```

4.3.5. Key backup

The loss of keys will lead to data loss. Therefore, according to the different use case scenarios, a key (pair) may need to backup. It is better to divide the key into several parts and store them into different storage devices.

```
submodule: key-backup
       +--rw key-backup?
          +--rw backup-enable         boolean
          +--rw backup-expire-time    yang-type:date-and-time
          +--rw backup-component* [component-name]
             +--rw component-name     string
             +--ro backup-medium      enumeration
```

4.3.6. Key destroy

The key and its associated key material must be destroyed when it is expired. Otherwise the expired key will be used by attackers to decrypt the data encrypted by this key. Also, the expired key can be used to analysis the cryptosystem.

```
submodule: key-destory
       +--rw key-destory
          +--rw method    {one|zerod|random number} enumeration
          +--rw number-of-times         unit8
```

4.4. Cert management

The TLS/DTLS and IPsec have been demonstrated as powerful security tools to provide data confidentiality and integrity services between network elements. In order to protect the TLS/DTLS or the IPsec connection against man-in-middle attack, peers have to authenticate from each other before connection establishing. The pre-shared key and the certificate are two of the most widely used methods to authenticate peers' identities. However, it requires to re-configure the pre-shared keys on all other endpoints/network elements if an additional network device is added in network. This complicated re-configuration process is easy to make errors. In the other hand, certificate is an idea way to extend authentications to a much larger scale of network. Peers request certificates that contain their entity information and public keys from certification authority (CA) in advance. The connection will be established only if the certificates are verified.

For a specific network device, such as switch and router, the certification service normally includes certificates request and updating, certificates validity check.

```
module: cert-management
    +--rw cert-management
       +--rw cert-management
       | . . . . . .
       +--rw crl-management
         . . . . . .
```

4.4.1. Cert management

A cert request file that contains the device public key and entity information is sent to the CA to apply a certificate. A CMP session is configured to request and update the certificates. A build-in default certificate in the device is used for identity authentication for CMP session. And the certificate must be updated in a reasonable period of time via CMP session.

```
module: cert-management
    +--rw cert-management*  [cert-name]
       +--rw cert-name                          string
       +--ro version                            string
       +--ro serial-number                      string
       +--ro signature-algorithm                identityref
       +--ro issuer-name                        string
       +--rw cert-request
       |  +--rw cmp-session-name                string
       +--ro validity
       |  +--ro start-time                 yang-type:date-and-time
       |  +--ro end-time                   yang-type:data-and-time
       +--ro subject-public-key
       |  +--ro public-key-algorithm            identityref
       |  +--ro public-key-length               unit16
       |  +--ro exponent                        unit32
       +--rw cert-auto-update
          +--rw cert-name                       string
          +--rw pki-domain-name                 string
          +--rw cmp-session-name                string
          +--rw auto-update-enable              boolean
          +--rw trigger-condition
             +--rw validity-percentage-number  unit8

grouping: cmp-session-config
      +--rw cmp-session-config*  [session-name]
         +--rw domain-name                      string
         +--rw session-name                     string
         +--rw entity-name                      string
         +--rw key-name                         string
         +--rw ca-server-name                   string
         +--rw default-cert-name                string
         +--rw cmp-server-url                   string
```

4.4.2. CRL management

The certificate revocation list (CRL) contains the invalid/expired certificates. It is equivalent to a blacklist of certificates issued by CA. The validity of a received cert is able to be checked by comparing with the CRL. The CRL need to update from CA by either an automatic or manual way.

```
     submodule: crl-management
            +--rw crl-management
               +--rw cert-validity-check-enable      boolean
               +--rw crl-update
                  +--rw previous-update-time   yang-type:date-and-time
                  +--rw auto-update
                  |  +--rw auto-update-enable         boolean
                  |  +--rw update-period              unit32
                  |  +--rw next-update-time    yang-type:date-and-time
                  |  +--rw update-method  {http|ldap} enumeration
                  +--rw manual-update
                     +--rw manual-update-enable      boolean
                     +--rw update-method  {http|ldap} enumeration
```

5. Infrastructure Layer YANG Module

This section shows a fraction of the infrastructure layer security baseline YANG modules.

```
module ietf-integrity-measurement{
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-integrity-measurement";
  prefix "im";

  import ietf-yang-types{
    prefix yang;
    reference
      "RFC6991: Common Yang Data Types";
  }

  organization
    "Huawei Technologies";

  contact
    "Yue Dong: dongyue6@huawei.com"
    "Liang Xia: Frank.xialiang@huawei.com"

  description
    "This module defines the configuration and status parameters of the
    functions that provide the integrity services in the bootstrapping
    and software updating phases.";

  identity hash-algorithms {
    description
      "base identities of hash algorithms options";
  }
```

```
  identity md5 {
    base hash-algorithms;
    description
      "The MD5 algorithm";
  }

  identity sha1 {
    base hash-algorithms;
    description
      "The SHA-1 algorithm";
    reference
      "RFC3174: US Secure Hash Algorithm 1 (SHA1).";
  }

  identity sha224 {
    base hash-algorithms;
    description
      "The SHA-224 algorithm.";
    reference
      "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and
      HKDF).";
  }

  identity sha256 {
    base hash-algorithms;
    description
      "The SHA-256 algorithm.";
    reference
      "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and
      HKDF).";
  }

  identity sha384 {
    base hash-algorithms;
    description
      "The SHA-384 algorithm.";
    reference
      "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and
      HKDF).";
  }

  identity sha512 {
    base hash-algorithm;
    description
      "The SHA-512 algorithm.";
    reference
      "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and
      HKDF).";
```

```
  }

  identity rsa-algorithms {
    description
      "rsa algorithms with different key length";
  }

  identity rsa1024 {
    base rsa-algorithms;
    description
      "The RSA algorithm using a 1024 bit key";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
      Cryptography Specifications 2.1"
  }

  identity rsa2048 {
    base rsa-algorithms;
    description
      "The RSA algorithm using a 2048 bit key";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
      Cryptography Specifications 2.1"
  }

  identity rsa3072 {
    base rsa-algorithms;
    description
      "The RSA algorithm using a 3072 bit key";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
      Cryptography Specifications 2.1"
  }

  identity rsa4096 {
    base rsa-algorithms;
    description
      "The RSA algorithm using a 4096 bit key";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
      Cryptography Specifications 2.1"
  }

  identity rsa7680 {
    base rsa-algorithms;
    description
      "The RSA algorithm using a 7680 bit key";
    reference
```

```
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
      Cryptography Specifications 2.1"
  }

  identity rsa15360 {
    base rsa-algorithms;
    description
      "The RSA algorithm using a 15360 bit key";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
      Cryptography Specifications 2.1"
  }

  identity rsa-padding {
    description
      "The identities of padding methods for rsa.";
  }

  identity oaep {
    base rsa-padding;
    description
      "The OAEP padding method for RSA.";
  }

  identity pss {
    base rsa-padding;
    description
      "The PSS padding method for RSA.";
  }

  container integrity-measurement {
    container bootstrapping {
      container trust-boot {
        leaf tpm-version {
          type string;
          description
            "version of the tpm chip";
        }

        leaf tpm-enable {
          type boolean;
          description
            "switch of the trust boot function";
        }

        uses hash-function

        list pcr-record {
```

```
          key "pcr-number";

          leaf pcr-number {
            type unit8;
            description
              "Number of pcr register";
          }

          leaf measurement-item{
            type enumeration {
              enum bios;
              enum bootloader;
              enum kernel;
              enum patch;
            }
            description
              "This property shows which item is measured and recored by
              the pcr";
          }

          leaf pcr-value {
            type string;
            description
              "The practical measurement value";
          }

          leaf pcr-benchmark-value {
            type string;
            description
              "The pre-defined benchmark criterion";
          }

          leaf verify-result {
            type boolean;
            description
              "The benchmark result for each pcr recorded value";
          }
        }
      }

      container secure-boot {
        leaf soc-model {
          type string;
          description
            "Model of the used SoC";
        }

        leaf-list measurement-items {
```

```
          type enumeration {
            enum bios;
            enum bootloader;
            enum kernel;
            enum patch;
          }
          description
            "List of the items to be measured in the secure boot
            process";
        }

        uses hash-function

        uses signature-algorithm

        container verification-pub-key {
          leaf key-name {
            type string;
            description
              "Name of the public key for verfication";
          }

          leaf key-length {
            type unit16;
            description
              "Length of the public key"
          }

          leaf store-medium {
            type enumeration {
              enum tmp;
              enum soc;
              enum hdd;
              enum hsm;
            }
            description
              "This property describes where the public key stores"
          }
        }
      }
    }

    container software-update {
      uses hash-function;

      uses signature-algorithm;

      container verification-pub-key {
```

```
        leaf key-name {
          type string;
          description
            "Name of the public key for verification";
        }

        leaf key-length {
          type unit16;
          description
            "Length of the public key";
        }

        leaf store-medium {
          type enumeration {
            enum tpm;
            enum soc;
            enum hdd;
            enum hsm;
          }
          description
            "This property decribes where the pub key stores"
        }
      }
    }
  }

  grouping hash-function {
    description
      "A group of Hash functions and their parameters";

    leaf algorithm {
      type identityref {
        base "hash-algorithm";
      }
      description
        "Identities of the used Hash algorithm";
    }

    leaf padding-method {
      type identityref;
      description
        ""
    }

    leaf digest-length {
      type unit16;
      description
        "The length of the Hash output";
```

```
    }
  }

  grouping signature-algorithms {
    "A group of algorithms and their configurable parameters for digital
    signature";

    choice algorithm-type {
      case rsa {
        leaf algorithm {
          type identityref {
            base "rsa-algorithm";
          }
          description
            "identities of the rsa algorithms for digital signature";
        }

        leaf padding-method {
          type identityref;
          description
            "identities of padding method for the used algorithm"
        }

        container pub-key {
          leaf public-exponent {
            type unit32;
            description
              "value of public exponent";
          }

          leaf modulo-value {
            type unit32;
            description
              "value of modulo";
          }
        }
      }

      case dsa {
        container tempory-key {
          leaf key-length {
            type unit16;
            description
              "The length of the tempory key.";
          }

          leaf randomness {
            type decimal64;
```

```
            description
              "This value represents the randomness of this key.";
          }
        }

        container prime-number {
          leaf prime-modulo {
            type unit32;
            description
              "value of modulo";
          }

          leaf prime-order {
            type unit32;
            description
              "value of prime number";
          }
        }
      }

      case ecdsa {
        containter tempory-key {
          leaf key-length {
            type unit16;
            description
              "The length of the tempory key that is generated by a
              random number generator.";
          }

          leaf randomness {
            type decimal64
            description
              "This value represents the randomness of the key. It is
              generated by a tool like sts 2.1.";
          }
        }

        leaf prime-modulo {
          type unit32;
          description
            "value of modulo";
        }

        leaf prime-order {
          type unit32;
          description
            "value of order";
        }
```

```
        uses hash-function

        container ec-parameter {
          leaf coefficient-a {
            type unit8;
            description
              "constant coefficient of the selected elliptic curve.";
          }

          leaf coefficient-b {
            type unit8;
            description
              "constant coefficient of the selected elliptic curve.";
          }
        }
      }
    }
  }
}
```

6. IANA Considerations

TBD

7. Security Considerations

TBD.

8. Acknowledgements

TBD

9. References

9.1. 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>.

9.2. Informative References

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", draft-ietf-netconf-ssh-client-server-05 (work in progress), October 2017.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-05 (work in progress), October 2017.

[I-D.ietf-sacm-information-model]
Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus, M., Haynes, D., and H. Birkholz, "SACM Information Model", draft-ietf-sacm-information-model-10 (work in progress), April 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.

[I-D.mandm-sacm-architecture]
Montville, A. and B. Munyan, "Security Automation and Continuous Monitoring (SACM) Architecture", draft-mandm-sacm-architecture-01 (work in progress), March 2018.

[I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-00 (work in progress), October 2015.

[I-D.xia-sacm-nid-dp-security-baseline]
Xia, L. and G. Zheng, "The Data Model of Network Infrastructure Device Data Plane Security Baseline", draft-xia-sacm-nid-dp-security-baseline-01 (work in progress), January 2018.

Authors’ Addresses

Yue Dong
Huawei

Email: dongyue6@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com

SACM N. Cam-Winget, Ed.
Internet-Draft Cisco Systems
Intended status: Informational L. Lorenzin
Expires: April 18, 2016 Pulse Secure
I. McDonald
High North Inc
A. Woland
Cisco Systems
October 16, 2015

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

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Several data models and protocols (including - but not limited to - NEA, TCG TNC, SCAP, SWIDs, XMPP, etc.) are in use today that allow different applications to perform the collection, acquisition, and assessment of posture. These applications can vary from being focused on general system and security management to specialized configuration, compliance, and control systems. With an existing varied set of applications, there is a strong desire to standardize data models, protocols, and interfaces to better allow for the automation of such data processes.

This document addresses general and architectural requirements defined in [I-D.ietf-sacm-requirements]. The architecture described enables standardized collection, acquisition, and verification of Posture and Posture Assessments. This architecture includes the components and interfaces that can be used to better identify the Information Model and type(s) of transport protocols needed for communication.

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

1.1. Requirements Language

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

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

2. Problem Statement

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

Security automation and continuous monitoring require a large and broad set of mission and business processes; to make the most effective use of technology, the same data must support multiple processes. The need for complex characterization and assessment necessitates components and functions that interoperate and can build off each other to enable far-ranging and/or deep-diving analysis. SACM is standardizing an information model, data models, operations, and transports that will allow for administrators to share with others and to use data from others interoperably.

3. Architectural Overview

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

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

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

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

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

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

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

Figure 1 illustrates the relationships between component roles and interfaces:

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

Figure 1: Simple Architectural Model

3.1. Component Roles

An endpoint, as defined in [I-D.ietf-sacm-terminology], can operate in two primary ways: as the target of an assessment, and/or as a functional component of the SACM architecture that can instantiate one or more functions (see Section 5). In the SACM architecture, individual endpoints may be a target endpoint, a component, or both simultaneously. An endpoint acting as a component may perform one or more roles. Components can take on the role(s) of Provider, Consumer, and/or Controller.

3.1.1. Provider

The Provider (Pr) is the component that contributes Posture Assessment Information and/or Guidance either spontaneously or in response to a request. A Provider can be a Posture Evaluator, Posture Collector, Data Store (see Section 3.1.3), or an application that has aggregated Posture Assessment Information that can be shared.

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

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

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

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

The Provider may only be able to share the Posture Assessment Information using a specific data model and protocol. It may use a standard data model and/or protocol, a non-standard data model and/or protocol, or any combination of standard and non-standard data models and protocols. However, it must support either one or more standard data models, or one or more standard protocols. It may also choose to advertise its capabilities through a metadata abstraction within the data model itself, or through the use of the registration function of the Controller (see Section 3.1.4).

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

3.1.2. Consumer

The Consumer (Cs) is the component that requests or accepts Posture Assessment Information and/or Guidance. A Consumer can be a Posture Evaluator, Report Generator, Data Store (see Section 5.2), or an application that consumes Posture Assessment Information in order to perform another function.

As described in Section 2.2 of the SACM Use Cases [I-D.ietf-sacm-use-cases], several usage scenarios are posed with different application types requesting posture assessment information. Whether it is a configuration verification system; a checklist verification system; or a system for detecting posture deviations, compliance or vulnerabilities, they all need to acquire information about Posture Assessment. The architectural component performing such requests is a Consumer.

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

There may be instances where a Consumer may be requesting information from various Providers and, due to its policy or application requirements, may need to be better informed of the Providers and their capabilities. In those use cases, a Consumer may also request to discover the respective capabilities of those Providers using the discovery function of the Controller (see Section 3.1.4) or may request metadata reflecting the capabilities of the Providers.

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

3.1.3. Types of Providers and Consumers

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

3.1.3.1. Collector

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

3.1.3.1.1. Internal Collector

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

3.1.3.1.2. External Collector

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

Examples:

- A RADIUS server, which collects information about which endpoints have logged onto the network

- A network profiling system, which collects information by discovering and classifying network nodes

- A Network Intrusion Detection System (NIDS) sensor, which collects information about endpoint behavior by observing network traffic

- A vulnerability scanner, which collects information about endpoint configuration by scanning endpoints

- A hypervisor, which collects information about endpoints running as virtual guests in its host environment

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

3.1.3.1.3. Collector Interactions With Target Endpoints

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

3.1.3.2. Evaluator

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

TODO: update the terminology doc to reflect this definition

Example: a NEA posture validator [RFC5209]

3.1.3.3. Report Generator

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

- o Endpoint Attribute Assertions, including Evaluation Results

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

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

3.1.3.4. Data Store

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

3.1.4. Controller

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

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

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

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

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

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

Proxy: Intermediary negotiating on behalf of a Consumer or Provider. Implements both control and data plane functions. A Controller acting as a Proxy:

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

* Receives a publication request from a Provider, accepts the published information, and distributes it to appropriate consumers.

* The information itself is distributed by, but not stored by, the Controller.

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

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

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

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

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

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

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

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

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

4. Interfaces between Consumers, Providers, and Controllers

A SACM interface is a transport carrying operations (e.g. publication via a RESTful API). As shown in Figure 1, communication can proceed with the following interfaces and expected functions and behaviors:

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

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

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

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

5. Component Functions

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

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

5.1. Control Plane Functions

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

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

* Consumers may request information independent of knowing the identities of the Providers.

* Providers may want to share the information without prior solicitation.

To address the above use cases, the architecture must account for an abstraction where a Controller may be defined to effect the authentication of the Consumers and Providers independent of the actual information-sharing communication channel. Consumers and Providers that consume or publish information without requiring knowledge of the Providers and Consumers respectively would function in a SACM system where the Controller is a distinct entity. As a distinct SACM component, the Controller would authenticate Providers and Consumers.

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

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

The Controller needs to be able to identify the endpoints participating as SACM components and the roles that they play. Similar to how access control may be effected via Authentication, Authorization, and Accounting Systems (e.g. AAA services), the same principle is defined; as AAA services depend on Identity Management services, the Controller will need a similar function and interface to Identity Management services. Note that implementations of this function is abstractly centralized, but to address scalability and the need to manage different resources (e.g. users, processes and devices) a distributed system that is centrally coordinated may be used.

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

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

5.2. Data Plane Functions

There are three basic functions to facilitate data flow:

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

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

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

6. Component Capabilities

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

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

7. Example Illustration of Functions and Workflow

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

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

Figure 2: Communications Model

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

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

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

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

Figure 3 illustrates an example flow for how Posture Assessment Information may flow.

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

Figure 3: Example Posture Information Flow

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

8. Acknowledgements

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

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

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

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

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

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

11. References

11.1. Normative References

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

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

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

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

11.2. Informative References

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

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

Authors’ Addresses

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

Email: ncamwing@cisco.com

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

Email: llorenzin@pulsesecure.net

Ira E McDonald
High North Inc
PO Box 221
Grand Marais, MI 49839
US

Email: blueroofmusic@gmail.com

Aaron Woland
Cisco Systems
1900 South Blvd. Suite 200
Charlotte, NC 28203
US

Email: loxx@cisco.com

SACM Working Group H. Birkholz
Internet-Draft Fraunhofer SIT
Intended status: Standards Track J. Fitzgerald-McKay
Expires: May 6, 2021 Department of Defense
C. Schmidt
The MITRE Corporation
D. Waltermire
NIST
November 02, 2020

Concise Software Identification Tags
draft-ietf-sacm-coswid-16

Abstract

ISO/IEC 19770-2:2015 Software Identification (SWID) tags provide an extensible XML-based structure to identify and describe individual software components, patches, and installation bundles. SWID tag representations can be too large for devices with network and storage constraints. This document defines a concise representation of SWID tags: Concise SWID (CoSWID) tags. CoSWID supports a similar set of semantics and features as SWID tags, as well as new semantics that allow CoSWIDs to describe additional types of information, all in a more memory efficient format.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

SWID tags, as defined in ISO-19770-2:2015 [SWID], provide a standardized XML-based record format that identifies and describes a specific release of software, a patch, or an installation bundle, which are referred to as software components in this document. Different software components, and even different releases of a particular software component, each have a different SWID tag record associated with them. SWID tags are meant to be flexible and able to express a broad set of metadata about a software component.

SWID tags are used to support a number of processes including but not limited to:

- Software Inventory Management, a part of a Software Asset Management [SAM] process, which requires an accurate list of discernible deployed software components.

- Vulnerability Assessment, which requires a semantic link between standardized vulnerability descriptions and software components installed on IT-assets [X.1520].

- Remote Attestation, which requires a link between reference integrity measurements (RIM) and Attester-produced event logs that complement attestation Evidence [I-D.ietf-rats-architecture].

While there are very few required fields in SWID tags, there are many optional fields that support different uses. A SWID tag consisting of only required fields might be a few hundred bytes in size; however, a tag containing many of the optional fields can be many orders of magnitude larger. Thus, real-world instances of SWID tags can be fairly large, and the communication of SWID tags in usage scenarios, such as those described earlier, can cause a large amount of data to be transported. This can be larger than acceptable for constrained devices and networks. Concise SWID (CoSWID) tags significantly reduce the amount of data transported as compared to a typical SWID tag through the use of the Concise Binary Object Representation (CBOR) [RFC7049]. [TODO: Add CoSWID size comparison.]

In a CoSWID, the human-readable labels of SWID data items are replaced with more concise integer labels (indices). This approach allows SWID and CoSWID to share a common implicit information model, with CoSWID providing an alternate data model [RFC3444]. While SWID and CoSWID are intended to share the same implicit information model, this specification does not define this information model, or a mapping between the the two data formats. While an attempt to align SWID and CoSWID tags has been made here, future revisions of ISO/IEC 19770-2:2015 or this specification might cause this implicit information model to diverge, since these specifications are maintained by different standards groups.

The use of CBOR to express SWID information in CoSWID tags allows both CoSWID and SWID tags to be part of an enterprise security solution for a wider range of endpoints and environments.

1.1. The SWID and CoSWID Tag Lifecycle

In addition to defining the format of a SWID tag record, ISO/IEC 19770-2:2015 defines requirements concerning the SWID tag lifecycle. Specifically, when a software component is installed on an endpoint, that software component's SWID tag is also installed. Likewise, when the software component is uninstalled or replaced, the SWID tag is deleted or replaced, as appropriate. As a result, ISO/IEC 19770-2:2015 describes a system wherein there is a correspondence between the set of installed software components on an endpoint, and the presence of the corresponding SWID tags for these components on that endpoint. CoSWIDs share the same lifecycle requirements as a SWID tag.

The SWID specification and supporting guidance provided in NIST Internal Report (NISTIR) 8060: Guidelines for the Creation of Interoperable SWID Tags [SWID-GUIDANCE] defines four types of SWID tags: primary, patch, corpus, and supplemental. The following text is paraphrased from these sources.

1. Primary Tag - A SWID or CoSWID tag that identifies and describes an installed software component on an endpoint. A primary tag is intended to be installed on an endpoint along with the corresponding software component.

2. Patch Tag - A SWID or CoSWID tag that identifies and describes an installed patch that has made incremental changes to a software component installed on an endpoint. A patch tag is intended to be installed on an endpoint along with the corresponding software component patch.

3. Corpus Tag - A SWID or CoSWID tag that identifies and describes an installable software component in its pre-installation state. A corpus tag can be used to represent metadata about an installation package or installer for a software component, a software update, or a patch.

4. Supplemental Tag - A SWID or CoSWID tag that allows additional information to be associated with a referenced SWID tag. This allows tools and users to record their own metadata about a software component without modifying SWID primary or patch tags created by a software provider.

The type of a tag is determined by specific data elements, which are discussed in Section 3, which also provides normative language for CoSWID semantics that implement this lifecycle. The following information helps to explain how these semantics apply to use of a CoSWID tag.

> Corpus, primary, and patch tags have similar functions in that they describe the existence and/or presence of different types of software components (e.g., software installers, software installations, software patches), and, potentially, different states of these software components. Supplemental tags have the same structure as other tags, but are used to provide information not contained in the referenced corpus, primary, and patch tags. All four tag types come into play at various points in the software lifecycle and support software management processes that depend on the ability to accurately determine where each software component is in its lifecycle.

```
                                  +------------+
                                  v            |
  Software      Software        Software     Software      Software
  Deployment -> Installation -> Patching  -> Upgrading  -> Removal

  Corpus        Primary         Primary      xPrimary      xPrimary
  Supplemental  Supplemental    Supplemental xSupplemental xSupplemental
                                Patch        xPatch
                                             Primary
                                             Supplemental
```

Figure 1: Use of Tag Types in the Software Lifecycle

Figure 1 illustrates the steps in the software lifecycle and the relationships among those lifecycle events supported by the four types of SWID and CoSWID tags. A detailed description of the four tags types is provided in Section 2.3. The figure identifies the types of tags that are used in each lifecycle event.

There are many ways in which software tags might be managed for the host the software is installed on. For example, software tags could be made available on the host or to an external software manager when storage is limited on the host.

In these cases the host or external software manager is responsible for management of the tags, including deployment and removal of the tags as indicated by the above lifecycle. Tags are deployed and previously deployed tags that are typically removed (indicated by an "x" prefix) at each lifecycle stage, as follows:

* Software Deployment. Before the software component is installed (i.e., pre-installation), and while the product is being deployed, a corpus tag provides information about the installation files and distribution media (e.g., CD/DVD, distribution package).

Corpus tags are not actually deployed on the target system but are intended to support deployment procedures and their dependencies at install-time, such as to verify the installation media.

* Software Installation. A primary tag will be installed with the software component (or subsequently created) to uniquely identify and describe the software component. Supplemental tags are created to augment primary tags with additional site-

specific or extended information. While not illustrated in the figure, patch tags can also be installed during software installation to provide information about software fixes deployed along with the base software installation.

* Software Patching. A new patch tag is provided, when a patch is applied to the software component, supplying details about the patch and its dependencies. While not illustrated in the figure, a corpus tag can also provide information about the patch installer and patching dependencies that need to be installed before the patch.

* Software Upgrading. As a software component is upgraded to a new version, new primary and supplemental tags replace existing tags, enabling timely and accurate tracking of updates to software inventory. While not illustrated in the figure, a corpus tag can also provide information about the upgrade installer and dependencies that need to be installed before the upgrade.

Note: In the context of software tagging software patching and updating differ in an important way. When installing a patch, a set of file modifications are made to pre-installed software which do not alter the version number or the descriptive metadata of an installed software component. An update can also make a set of file modifications, but the version number or the descriptive metadata of an installed software component are changed.

* Software Removal. Upon removal of the software component, relevant SWID tags are removed. This removal event can trigger timely updates to software inventory reflecting the removal of the product and any associated patch or supplemental tags.

As illustrated in the figure, supplemental tags can be associated with any corpus, primary, or patch tag to provide additional metadata about an installer, installed software, or installed patch respectively.

Understanding the use of CoSWIDs in the software lifecycle provides a basis for understanding the information provided in a CoSWID and the associated semantics of this information. Each of the different SWID and CoSWID tag types provide different sets of information. For example, a "corpus tag" is used to describe a software component's installation image on an installation media, while a "patch tag" is meant to describe a patch that modifies some other software component.

1.2. Concise SWID Format

This document defines the CoSWID tag format, which is based on CBOR. CBOR-based CoSWID tags offer a more concise representation of SWID information as compared to the XML-based SWID tag representation in ISO-19770-2:2015. The structure of a CoSWID is described via the Concise Data Definition Language (CDDL) [RFC8610]. The resulting CoSWID data definition is aligned to the information able to be expressed with the XML schema definition of ISO-19770-2:2015 [SWID]. This alignment allows both SWID and CoSWID tags to represent a common set of software component information and allows CoSWID tags to support the same uses as a SWID tag.

The vocabulary, i.e., the CDDL names of the types and members used in the CoSWID CDDL specification, are mapped to more concise labels represented as small integer values (indices). The names used in the CDDL specification and the mapping to the CBOR representation using integer indices is based on the vocabulary of the XML attribute and element names defined in ISO/IEC 19770-2:2015.

1.3. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Concise SWID Data Definition

The following describes the general rules and processes for encoding data using CDDL representation. Prior familiarity with CBOR and CDDL concepts will be helpful in understanding this CoSWID specification.

This section describes the conventions by which a CoSWID is represented in the CDDL structure. The CamelCase [CamelCase] notation used in the XML schema definition is changed to a hyphen-separated notation [KebabCase] (e.g. ResourceCollection is named resource-collection) in the CoSWID CDDL specification. This deviation from the original notation used in the XML representation reduces ambiguity when referencing certain attributes in corresponding textual descriptions. An attribute referred to by its name in CamelCase notation explicitly relates to XML SWID tags; an attribute referred to by its name in KebabCase notation explicitly relates to CBOR CoSWID tags. This approach simplifies the composition of further work that reference both XML SWID and CBOR CoSWID documents.

In most cases, mapping attribute names between SWID and CoSWID can be done automatically by converting between CamelCase and KebabCase attribute names. However, some CoSWID CDDL attribute names show greater variation relative to their corresponding SWID XML Schema attributes. This is done when the change improves clarity in the CoSWID specification. For example the "name" and "version" SWID fields corresponds to the "software-name" and "software-version" CoSWID fields, respectively. As such, it is not always possible to mechanically translate between corresponding attribute names in the two formats. In such cases, a manual mapping will need to be used.

The 57 human-readable text labels of the CDDL-based CoSWID vocabulary are mapped to integer indices via a block of rules at the bottom of the definition. This allows a more concise integer-based form to be stored or transported, as compared to the less efficient text-based form of the original vocabulary.

In CBOR, an array is encoded using bytes that identify the array, and the array's length or stop point (see [RFC7049]). To make items that support 1 or more values, the following CDDL notion is used.

```
_name_ = (_label_ => _data_ / [ 2* _data_ ])
```

The CDDL rule above allows either a single data item or an array of 2 or more data values to be provided. When a singleton data value is provided, the CBOR markers for the array, array length, and stop point are not needed, saving bytes. When two or more data values are provided, these values are encoded as an array. This modeling pattern is used frequently in the CoSWID CDDL specification to allow for more efficient encoding of singleton values.

[TODO: Are there any considerations that would need to be made for versioning CoSWID beyond the native support provided with CBOR?]

The following subsections describe the different parts of the CoSWID model.

2.1. Character Encoding

The CDDL "text" type is represented in CBOR as a major type 3, which represents "a string of Unicode characters that [are] encoded as UTF-8 [RFC3629]" (see [RFC7049] Section 2.1). Thus both SWID and CoSWID use UTF-8 for the encoding of characters in text strings.

To ensure that UTF-8 character strings are able to be encoded/decoded and exchanged interoperably, text strings in CoSWID MUST be encoded consistent with the Net-Unicode definition defined in [RFC5198].

All names registered with IANA according to requirements in Section Section 5.2 also MUST be valid according to the XML Schema NMToken data type (see [W3C.REC-xmlschema-2-20041028] Section 3.3.4) to ensure compatibility with the SWID specification where these names are used.

2.2. Concise SWID Extensions

The CoSWID specification contains two features that are not included in the SWID specification on which it is based. These features are:

- o The explicit definition of types for some attributes in the ISO-19770-2:2015 XML representation that are typically represented by the "any attribute" in the SWID model. These are covered in Section 2.5.

- o The inclusion of extension points in the CoSWID specification using CDDL sockets (see [RFC8610] Section 3.9). The use of CDDL sockets allow for well-formed extensions to be defined in supplementary CDDL descriptions that support additional uses of CoSWID tags that go beyond the original scope of ISO-19770-2:2015 tags. This extension mechanism can also be used to update the CoSWID format as revisions to ISO-19770-2 are published.

The following CDDL sockets (extension points) are defined in this document, which allow the addition of new information structures to their respective CDDL groups.

Map Name	CDDL Socket	Defined in
concise-swid-tag	$$coswid-extension	Section 2.3
entity-entry	$$entity-extension	Section 2.6
link-entry	$$link-extension	Section 2.7
software-meta-entry	$$software-meta-extension	Section 2.8
file-entry	$$file-extension	Section 2.9.2
directory-entry	$$directory-extension	Section 2.9.2
process-entry	$$process-extension	Section 2.9.2
resource-entry	$$resource-extension	Section 2.9.2
payload-entry	$$payload-extension	Section 2.9.3
evidence-entry	$$evidence-extension	Section 2.9.4

Table 1: CoSWID CDDL Group Extension Points

The CoSWID Items Registry defined in Section 5.1 provides a registration mechanism allowing new items, and their associated index values, to be added to the CoSWID model through the use of the CDDL sockets described in the table above. This registration mechanism provides for well-known index values for data items in CoSWID extensions, allowing these index values to be recognized by implementations supporting a given extension.

The following additional CDDL sockets are defined in this document to allow for adding new values to corresponding type-choices (i.e. to represent enumerations) via custom CDDL specifications.

Enumeration Name	CDDL Socket	Defined in
version-scheme	$version-scheme	Section 4.1
role	$role	Section 4.2
ownership	$ownership	Section 4.3
rel	$rel	Section 4.4
use	$use	Section 4.5

Table 2: CoSWID CDDL Enumeration Extension Points

A number of CoSWID value registries are also defined in Section 5.2 that allow new values to be registered with IANA for the enumerations above. This registration mechanism supports the definition of new well-known index values and names for new enumeration values used by CoSWID, which can also be used by other software tagging specifications. This registration mechanism allows new standardized enumerated values to be shared between multiple tagging specifications (and associated implementations) over time.

2.3. The concise-swid-tag Map

The CDDL specification for the root concise-swid-tag map is as follows and this rule and its constraints MUST be followed when creating or validating a CoSWID tag:

```
concise-swid-tag = {
  tag-id => text / bstr .size 16,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  software-name => text,
  ? software-version => text,
  ? version-scheme => $version-scheme,
  ? media => text,
  ? software-meta => software-meta-entry / [ 2* software-meta-entry ],
  entity => entity-entry / [ 2* entity-entry ],
  ? link => link-entry / [ 2* link-entry ],
  ? payload-or-evidence,
  global-attributes,
  * $$coswid-extension,
}
```

```
payload-or-evidence //= ( payload => payload-entry ] )
payload-or-evidence //= ( payload => [ 2* payload-entry )
payload-or-evidence //= ( evidence => evidence-entry )
payload-or-evidence //= ( evidence => [ 2* evidence-entry ] )

tag-id = 0
software-name = 1
entity = 2
evidence = 3
link = 4
software-meta = 5
payload = 6
corpus = 8
patch = 9
media = 10
supplemental = 11
tag-version = 12
software-version = 13
version-scheme = 14

$version-scheme /= multipartnumeric
$version-scheme /= multipartnumeric-suffix
$version-scheme /= alphanumeric
$version-scheme /= decimal
$version-scheme /= semver
$version-scheme /= uint / text
multipartnumeric = 1
multipartnumeric-suffix = 2
alphanumeric = 3
decimal = 4
semver = 16384
```

The following describes each member of the concise-swid-tag root map.

- global-attributes: A list of items including an optional language definition to support the processing of text-string values and an unbounded set of any-attribute items. Described in Section 2.5.

- tag-id (index 0): A 16 byte binary string or textual identifier uniquely referencing a software component. The tag identifier MUST be globally unique. If represented as a 16 byte binary string, the identifier MUST be a valid universally unique identifier as defined by [RFC4122]. There are no strict guidelines on how this identifier is structured, but examples include a 16 byte GUID (e.g. class 4 UUID) [RFC4122], or a text string appended to a DNS domain name to ensure uniqueness across organizations.

- o tag-version (index 12): An integer value that indicate the specific release revision of the tag. Typically, the initial value of this field is set to 0 and the value is monotonically increased for subsequent tags produced for the same software component release. This value allows a CoSWID tag producer to correct an incorrect tag previously released without indicating a change to the underlying software component the tag represents. For example, the tag version could be changed to add new metadata, to correct a broken link, to add a missing payload entry, etc. When producing a revised tag, the new tag-version value MUST be greater than the old tag-version value.

- o corpus (index 8): A boolean value that indicates if the tag identifies and describes an installable software component in its pre-installation state. Installable software includes a installation package or installer for a software component, a software update, or a patch. If the CoSWID tag represents installable software, the corpus item MUST be set to "true". If not provided, the default value MUST be considered "false".

- o patch (index 9): A boolean value that indicates if the tag identifies and describes an installed patch that has made incremental changes to a software component installed on an endpoint. If a CoSWID tag is for a patch, the patch item MUST be set to "true". If not provided, the default value MUST be considered "false". A patch item's value MUST NOT be set to "true" if the installation of the associated software package changes the version of a software component.

- o supplemental (index 11): A boolean value that indicates if the tag is providing additional information to be associated with another referenced SWID or CoSWID tag. This allows tools and users to record their own metadata about a software component without modifying SWID primary or patch tags created by a software provider. If a CoSWID tag is a supplemental tag, the supplemental item MUST be set to "true". If not provided, the default value MUST be considered "false".

- o software-name (index 1): This textual item provides the software component's name. This name is likely the same name that would appear in a package management tool.

- o software-version (index 13): A textual value representing the specific release or development version of the software component.

- o version-scheme (index 14): An integer or textual value representing the versioning scheme used for the software-version item. If an integer value is used it MUST be an index value in the range -256 to 65535. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 65535 correspond to registered entries in the IANA "Software Tag Version Scheme Values" registry (see Section Section 5.2.4. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on a Version Scheme Name from the IANA "Software Tag Version Scheme Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- media (index 10): This text value is a hint to the tag consumer to understand what target platform this tag applies to. This item item MUST be formatted as a query as defined by the W3C Media Queries Recommendation (see [W3C.REC-css3-mediaqueries-20120619]). Support for media queries are included here for interoperability with [SWID], which does not provide any further requirements for media query use. Thus, this specification does not clarify how a media query is to be used for a CoSWID.

- software-meta (index 5): An open-ended map of key/value data pairs. A number of predefined keys can be used within this item providing for common usage and semantics across the industry. Use of this map allows any additional attribute to be included in the tag. It is expected that industry groups will use a common set of attribute names to allow for interoperability within their communities. Described in Section 2.8.

- entity (index 2): Provides information about one or more organizations responsible for producing the CoSWID tag, and producing or releasing the software component referenced by this CoSWID tag. Described in Section 2.6.

- link (index 4): Provides a means to establish relationship arcs between the tag and another items. A given link can be used to establish the relationship between tags or to reference another resource that is related to the CoSWID tag, e.g. vulnerability database association, ROLIE feed [RFC8322], MUD resource [RFC8520], software download location, etc). This is modeled after the HTML "link" element. Described in Section 2.7.

- payload (index 6): This item represents a collection of software artifacts (described by child items) that compose the target software. For example, these artifacts could be the files included with an installer for a corpus tag or installed on an endpoint when the software component is installed for a primary or patch tag. The artifacts listed in a payload may be a superset of the software artifacts that are actually installed. Based on user selections at install time, an installation might not include every artifact that could be created or executed on the endpoint when the software component is installed or run. Described in Section 2.9.3.

- evidence-entry (index 3): This item can be used to record the results of a software discovery process used to identify untagged software on an endpoint or to represent indicators for why software is believed to be installed on the endpoint. In either case, a CoSWID tag can be created by the tool performing an analysis of the software components installed on the endpoint. Described in Section 2.9.4.

- $$coswid-extension: This CDDL socket is used to add new information structures to the concise-swid-tag root map. See Section 2.2.

2.4. concise-swid-tag Co-constraints

The following co-constraints apply to the information provided in the concise-swid-tag group.

- The patch and supplemental items MUST NOT both be set to "true".

- If the patch item is set to "true", the tag SHOULD contain at least one link item (see Section Section 2.7) with both the rel item value of "patches" and an href item specifying an association with the software that was patched.

- If the supplemental item is set to "true", the tag SHOULD contain at least one link item with both the rel item value of "supplemental" and an href item specifying an association with the software that is supplemented.

- If all of the corpus, patch, and supplemental items are "false", or if the corpus item is set to "true", then a software-version item MUST be included with a value set to the version of the software component. This ensures that primary and corpus tags have an identifiable software version.

2.5. The global-attributes Group

The global-attributes group provides a list of items, including an optional language definition to support the processing of text-string values, and an unbounded set of any-attribute items allowing for additional items to be provided as a general point of extension in the model.

The CDDL for the global-attributes follows:

```
global-attributes = (
  ? lang,
  * any-attribute,
)

any-attribute = (
  label => text / int / [ 2* text ] / [ 2* int ]
)

label = text / int
```

The following describes each child item of this group.

- lang (index 15): A textual language tag that conforms with IANA "Language Subtag Registry" [RFC5646]. The context of the specified language applies to all sibling and descendant textual values, unless a descendant object has defined a different language tag. Thus, a new context is established when a descendant object redefines a new language tag. All textual values within a given context MUST be considered expressed in the specified language.

- any-attribute: This sub-group provides a means to include arbitrary information via label/index ("key") value pairs. Labels can be either a single integer or text string. Values can be a single integer, a text string, or an array of integers or text strings.

2.6. The entity-entry Map

The CDDL for the entity-entry map follows:

```
entity-entry = {
  entity-name => text,
  ? reg-id => any-uri,
  role => $role / [ 2* $role ],
  ? thumbprint => hash-entry,
  global-attributes,
  * $$entity-extension,
}

entity-name = 31
reg-id = 32
role = 33
thumbprint = 34

$role /= tag-creator
$role /= software-creator
$role /= aggregator
$role /= distributor
$role /= licensor
$role /= maintainer
$role /= uint / text
tag-creator=1
software-creator=2
aggregator=3
distributor=4
licensor=5
maintainer=6
```

The following describes each child item of this group.

- global-attributes: The global-attributes group described in Section 2.5.

- entity-name (index 31): The textual name of the organizational entity claiming the roles specified by the role item for the CoSWID tag.

- reg-id (index 32): The registration id value is intended to uniquely identify a naming authority in a given scope (e.g. global, organization, vendor, customer, administrative domain, etc.) for the referenced entity. The value of a registration ID MUST be a RFC 3986 URI. The scope SHOULD be the scope of an organization.

- role (index 33): An integer or textual value representing the relationship(s) between the entity, and this tag or the referenced software component. If an integer value is used it MUST be an index value in the range -256 to 255. Integer values in the range

-256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Software Tag Entity Role Values" registry (see Section Section 5.2.5. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on a Role Name from the IANA "Software Tag Entity Role Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

The following additional requirements exist for the use of the "role" item:

* An entity item MUST be provided with the role of "tag-creator" for every CoSWID tag. This indicates the organization that created the CoSWID tag.

* An entity item SHOULD be provided with the role of "software-creator" for every CoSWID tag, if this information is known to the tag creator. This indicates the organization that created the referenced software component.

- thumbprint (index 34): The value of the thumbprint item provides an integer-based hash algorithm identifier (hash-alg-id) and a byte string value (hash-value) that contains the corresponding hash value (i.e. the thumbprint) of the signing entity's public key certificate. This provides an indicator of which entity signed the CoSWID tag, which will typically be the tag creator. If the hash-alg-id is not known, then the integer value "0" MUST be used. This ensures parity between the SWID tag specification [SWID], which does not allow an algorithm to be identified for this field. See Section 2.9.1 for more details on the use of the hash-entry data structure.

- $$entity-extension: This CDDL socket can be used to extend the entity-entry group model. See Section 2.2.

2.7. The link-entry Map

The CDDL for the link-entry map follows:

```
link-entry = {
  ? artifact => text,
  href => any-uri,
  ? media => text,
  ? ownership => $ownership,
  rel => $rel,
  ? media-type => text,
```

```
     ? use => $use,
     global-attributes,
     * $$link-extension,
   }

   media = 10
   artifact = 37
   href = 38
   ownership = 39
   rel = 40
   media-type = 41
   use = 42

   $ownership /= shared
   $ownership /= private
   $ownership /= abandon
   $ownership /= uint / text
   shared=1
   private=2
   abandon=3

   $rel /= ancestor
   $rel /= component
   $rel /= feature
   $rel /= installationmedia
   $rel /= packageinstaller
   $rel /= parent
   $rel /= patches
   $rel /= requires
   $rel /= see-also
   $rel /= supersedes
   $rel /= supplemental
   $rel /= -356..65536 / text
   ancestor=1
   component=2
   feature=3
   installationmedia=4
   packageinstaller=5
   parent=6
   patches=7
   requires=8
   see-also=9
   supersedes=10
   supplemental=11

   $use /= optional
   $use /= required
   $use /= recommended
```

```
$use /= uint / text
optional=1
required=2
recommended=3
```

The following describes each member of this map.

- global-attributes: The global-attributes group described in Section 2.5.

- artifact (index: 37): To be used with rel="installation-media", this item's value provides the path to the installer executable or script that can be run to launch the referenced installation. Links with the same artifact name MUST be considered mirrors of each other, allowing the installation media to be acquired from any of the described sources.

- href (index 38): A URI-reference [RFC3986] for the referenced resource. The "href" item's value can be, but is not limited to, the following (which is a slightly modified excerpt from [SWID]):

  * If no URI scheme is provided, then the URI-reference is a a relative reference relative to the URI of the CoSWID tag. For example, "./folder/supplemental.coswid".

  * a physical resource location with any acceptable URI scheme (e.g., file:// http:// https:// ftp://)

  * a URI with "swid:" as the scheme refers to another SWID or CoSWID by the referenced tag's tag-id. This URI needs to be resolved in the context of the endpoint by software that can lookup other SWID or CoSWID tags. For example, "swid:2df9de35-0aff-4a86-ace6-f7dddd1ade4c" references the tag with the tag-id value "2df9de35-0aff-4a86-ace6-f7dddd1ade4c".

  * a URI with "swidpath:" as the scheme, which refers to another software tag via an XPATH query [W3C.REC-xpath20-20101214]. This scheme is provided for compatibility with [SWID]. This specification does not define how to resolve an XPATH query in the context of CBOR.

- media (index 10): A hint to the consumer of the link to what target platform the link is applicable to. This item represents a query as defined by the W3C Media Queries Recommendation (see [W3C.REC-css3-mediaqueries-20120619]). See also media defined in Section 2.3.

- ownership (index 39): An integer or textual value used when the "href" item references another software component to indicate the degree of ownership between the software component referenced by the COSWID tag and the software component referenced by the link. If an integer value is used it MUST be an index value in the range -256 to 255. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Software Tag Link Ownership Values" registry (see Section Section 5.2.6. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on a Ownership Type Name from the IANA "Software Tag Link Ownership Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- rel (index 40): An integer or textual value that identifies the relationship between this CoSWID and the target resource identified by the "href" item. If an integer value is used it MUST be an index value in the range -256 to 65535. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 65535 correspond to registered entries in the IANA "Software Tag Link Relationship Values" registry (see Section Section 5.2.7. If a string value is used it MUST be either a private use name as defined in Section Section 5.2.2 or a "Relation Name" from the IANA "Link Relation Types" registry: https://www.iana.org/assignments/link-relations/link-relations.xhtml as defined by [RFC8288]. When a string value defined in the IANA "Software Tag Link Relationship Values" registry matches a Relation Name defined in the IANA "Link Relation Types" registry, the index value in the IANA "Software Tag Link Relationship Values" registry MUST be used instead, as this relationship has a specialized meaning in the context of a CoSWID tag. String values based on a Relationship Type Name from the IANA "Software Tag Link Relationship Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- media-type (index 41): A link can point to arbitrary resources on the endpoint, local network, or Internet using the href item. Use of this item supplies the resource consumer with a hint of what type of resource to expect. Media types are identified by referencing a "Name" from the IANA "Media Types" registry: http://www.iana.org/assignments/media-types/media-types.xhtml.

- use (index 42): An integer or textual value used to determine if the referenced software component has to be installed before installing the software component identified by the COSWID tag. If an integer value is used it MUST be an index value in the range -256 to 255. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Link Use Values" registry (see Section Section 5.2.8. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on an Link Use Type Name from the IANA "Software Tag Link Use Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- $$link-extension: This CDDL socket can be used to extend the link-entry map model. See Section 2.2.

2.8. The software-meta-entry Map

The CDDL for the software-meta-entry map follows:

```
software-meta-entry = {
  ? activation-status => text,
  ? channel-type => text,
  ? colloquial-version => text,
  ? description => text,
  ? edition => text,
  ? entitlement-data-required => bool,
  ? entitlement-key => text,
  ? generator => text,
  ? persistent-id => text,
  ? product => text,
  ? product-family => text,
  ? revision => text,
  ? summary => text,
  ? unspsc-code => text,
  ? unspsc-version => text,
  global-attributes,
  * $$software-meta-extension,
}

activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
edition = 47
entitlement-data-required = 48
entitlement-key = 49
generator = 50
persistent-id = 51
product = 52
product-family = 53
revision = 54
summary = 55
unspsc-code = 56
unspsc-version = 57
```

The following describes each child item of this group.

- global-attributes: The global-attributes group described in Section 2.5.

- activation-status (index 43): A textual value that identifies how the software component has been activated, which might relate to specific terms and conditions for its use (e.g. Trial, Serialized, Licensed, Unlicensed, etc) and relate to an entitlement. This attribute is typically used in supplemental tags as it contains information that might be selected during a specific install.

- channel-type (index 44): A textual value that identifies which sales, licensing, or marketing channel the software component has been targeted for (e.g. Volume, Retail, OEM, Academic, etc). This attribute is typically used in supplemental tags as it contains information that might be selected during a specific install.

- colloquial-version (index 45): A textual value for the software component's informal or colloquial version. Examples may include a year value, a major version number, or similar value that are used to identify a group of specific software component releases that are part of the same release/support cycle. This version can be the same through multiple releases of a software component, while the software-version specified in the concise-swid-tag group is much more specific and will change for each software component release. This version is intended to be used for string comparison only and is not intended to be used to determine if a specific value is earlier or later in a sequence.

- description (index 46): A textual value that provides a detailed description of the software component. This value MAY be multiple paragraphs separated by CR LF characters as described by [RFC5198].

- edition (index 47): A textual value indicating that the software component represents a functional variation of the code base used to support multiple software components. For example, this item can be used to differentiate enterprise, standard, or professional variants of a software component.

- entitlement-data-required (index 48): A boolean value that can be used to determine if accompanying proof of entitlement is needed when a software license reconciliation process is performed.

- entitlement-key (index 49): A vendor-specific textual key that can be used to identify and establish a relationship to an entitlement. Examples of an entitlement-key might include a serial number, product key, or license key. For values that relate to a given software component install (i.e., license key), a supplemental tag will typically contain this information. In other cases, where a general-purpose key can be provided that applies to all possible installs of the software component on different endpoints, a primary tag will typically contain this information.

- generator (index 50): The name (or tag-id) of the software component that created the CoSWID tag. If the generating software component has a SWID or CoSWID tag, then the tag-id for the generating software component SHOULD be provided.

- persistent-id (index 51): A globally unique identifier used to identify a set of software components that are related. Software components sharing the same persistent-id can be different versions. This item can be used to relate software components, released at different points in time or through different release channels, that may not be able to be related through use of the link item.

- product (index 52): A basic name for the software component that can be common across multiple tagged software components (e.g., Apache HTTPD).

- product-family (index 53): A textual value indicating the software components overall product family. This should be used when multiple related software components form a larger capability that is installed on multiple different endpoints. For example, some software families may consist of server, client, and shared service components that are part of a larger capability. Email systems, enterprise applications, backup services, web conferencing, and similar capabilities are examples of families. Use of this item is not intended to represent groups of software that are bundled or installed together. The persistent-id or link items SHOULD be used to relate bundled software components.

- revision (index 54): A string value indicating an informal or colloquial release version of the software. This value can provide a different version value as compared to the software-version specified in the concise-swid-tag group. This is useful when one or more releases need to have an informal version label that differs from the specific exact version value specified by software-version. Examples can include SP1, RC1, Beta, etc.

- summary (index 55): A short description of the software component. This MUST be a single sentence suitable for display in a user interface.

- unspsc-code (index 56): An 8 digit UNSPSC classification code for the software component as defined by the United Nations Standard Products and Services Code (UNSPSC, [UNSPSC]).

- unspsc-version (index 57): The version of UNSPSC used to define the unspsc-code value.

- $$meta-extension: This CDDL socket can be used to extend the software-meta-entry group model. See Section 2.2.

2.9. The Resource Collection Definition

2.9.1. The hash-entry Array

CoSWID adds explicit support for the representation of hash entries using algorithms that are registered in the IANA "Named Information Hash Algorithm Registry" using the hash member (index 7) and the corresponding hash-entry type. This is the equivalent of the namespace qualified "hash" attribute in [SWID].

```
hash-entry = [
  hash-alg-id: int,
  hash-value: bytes,
]
```

The number used as a value for hash-alg-id MUST refer an ID in the "Named Information Hash Algorithm Registry" with a Status of "current" (see https://www.iana.org/assignments/named-information/named-information.xhtml); other hash algorithms MUST NOT be used. The hash-value MUST represent the raw hash value of the hashed resource generated using the hash algorithm indicated by the hash-alg-id.

2.9.2. The resource-collection Group

A list of items both used in evidence (created by a software discovery process) and payload (installed in an endpoint) content of a CoSWID tag document to structure and differentiate the content of specific CoSWID tag types. Potential content includes directories, files, processes, or resources.

The CDDL for the resource-collection group follows:

```
path-elements-group = ( ? directory => directory-entry / [ 2* directory-entry ],
                        ? file => file-entry / [ 2* file-entry ],
                      )

resource-collection = (
  path-elements-group,
  ? process => process-entry / [ 2* process-entry ],
  ? resource => resource-entry / [ 2* resource-entry ],
  * $$resource-collection-extension,
)

filesystem-item = (
  ? key => bool,
  ? location => text,
  fs-name => text,
```

```
  ? root => text,
  global-attributes,
)

file-entry = {
  filesystem-item,
  ? size => integer,
  ? file-version => text,
  ? hash => hash-entry,
  * $$file-extension,
}

directory-entry = {
  filesystem-item,
  path-elements => { path-elements-group },
  * $$directory-extension,
}

process-entry = {
  process-name => text,
  ? pid => integer,
  global-attributes,
  * $$process-extension,
}

resource-entry = {
  type => text,
  global-attributes,
  * $$resource-extension,
}

directory = 16
file = 17
process = 18
resource = 19
size = 20
file-version = 21
key = 22
location = 23
fs-name = 24
root = 25
path-elements = 26
process-name = 27
pid = 28
type = 29
```

The following describes each member of the groups and maps illustrated above.

- filesystem-item: A list of common items used for representing the filesystem root, relative location, name, and significance of a file or directory item.

- global-attributes: The global-attributes group described in Section 2.5.

- directory (index 16): A directory item allows child directory and file items to be defined within a directory hierarchy for the software component.

- file (index 17): A file item allows details about a file to be provided for the software component.

- process (index 18): A process item allows details to be provided about the runtime behavior of the software component, such as information that will appear in a process listing on an endpoint.

- resource (index 19): A resource item can be used to provide details about an artifact or capability expected to be found on an endpoint or evidence collected related to the software component. This can be used to represent concepts not addressed directly by the directory, file, or process items. Examples include: registry keys, bound ports, etc. The equivalent construct in [SWID] is currently under specified. As a result, this item might be further defined through extension in the future.

- size (index 20): The file's size in bytes.

- file-version (index 21): The file's version as reported by querying information on the file from the operating system.

- key (index 22): A boolean value indicating if a file or directory is significant or required for the software component to execute or function properly. These are files or directories that can be used to affirmatively determine if the software component is installed on an endpoint.

- location (index 23): The filesystem path where a file is expected to be located when installed or copied. The location MUST be either relative to the location of the parent directory item (preferred) or relative to the location of the CoSWID tag if no parent is defined. The location MUST NOT include a file's name, which is provided by the fs-name item.

- fs-name (index 24): The name of the directory or file without any path information. This aligns with a file "name" in [SWID].

- root (index 25): A filesystem-specific name for the root of the filesystem. The location item is considered relative to this location if specified. If not provided, the value provided by the location item is expected to be relative to its parent or the location of the CoSWID tag if no parent is provided.

- path-elements (index 26): This group allows a hierarchy of directory and file items to be defined in payload or evidence items.

- process-name (index 27): The software component's process name as it will appear in an endpoint's process list. This aligns with a process "name" in [SWID].

- pid (index 28): The process ID identified for a running instance of the software component in the endpoint's process list. This is used as part of the evidence item.

- type (index 29): A string indicating the type of resource.

- $$resource-collection-extension: This CDDL socket can be used to extend the resource-collection group model. This can be used to add new specialized types of resources. See Section 2.2.

- $$file-extension: This CDDL socket can be used to extend the file-entry group model. See Section 2.2.

- $$directory-extension: This CDDL socket can be used to extend the directory-entry group model. See Section 2.2.

- $$process-extension: This CDDL socket can be used to extend the process-entry group model. See Section 2.2.

- $$resource-extension: This CDDL socket can be used to extend the resource-entry group model. See Section 2.2.

2.9.3. The payload-entry Map

The CDDL for the payload-entry map follows:

```
payload-entry = {
  resource-collection,
  global-attributes,
  * $$payload-extension,
}
```

The following describes each child item of this group.

o global-attributes: The global-attributes group described in Section 2.5.

o resource-collection: The resource-collection group described in Section 2.9.2.

o $$payload-extension: This CDDL socket can be used to extend the payload-entry group model. See Section 2.2.

2.9.4. The evidence-entry Map

The CDDL for the evidence-entry map follows:

```
evidence-entry = {
  resource-collection,
  ? date => time,
  ? device-id => text,
  global-attributes,
  * $$evidence-extension,
}

date = 35
device-id = 36
```

[QUESTION: Is "time" a correct representation of XSD:date?]

The following describes each child item of this group.

o global-attributes: The global-attributes group described in Section 2.5.

o resource-collection: The resource-collection group described in Section 2.9.2.

o date (index 35): The date and time the information was collected pertaining to the evidence item.

o device-id (index 36): The endpoint's string identifier from which the evidence was collected.

o $$evidence-extension: This CDDL socket can be used to extend the evidence-entry group model. See Section 2.2.

2.10. Full CDDL Specification

In order to create a valid CoSWID document the structure of the corresponding CBOR message MUST adhere to the following CDDL specification.

```
<CODE BEGINS>
concise-swid-tag = {
  tag-id => text / bstr .size 16,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  software-name => text,
  ? software-version => text,
  ? version-scheme => $version-scheme,
  ? media => text,
  ? software-meta => software-meta-entry / [ 2* software-meta-entry ],
  entity => entity-entry / [ 2* entity-entry ],
  ? link => link-entry / [ 2* link-entry ],
  ? payload-or-evidence,
  global-attributes,
  * $$coswid-extension,
}

payload-or-evidence //= ( payload => payload-entry )
payload-or-evidence //= ( payload => [ 2* payload-entry ] )
payload-or-evidence //= ( evidence => evidence-entry )
payload-or-evidence //= ( evidence => [ 2* evidence-entry ] )

any-uri = text
label = text / int

$version-scheme /= multipartnumeric
$version-scheme /= multipartnumeric-suffix
$version-scheme /= alphanumeric
$version-scheme /= decimal
$version-scheme /= semver
$version-scheme /= uint / text

any-attribute = (
  label => text / int / [ 2* text ] / [ 2* int ]
)

global-attributes = (
  ? lang => text,
  * any-attribute,
)

hash-entry = [
  hash-alg-id: int,
  hash-value: bytes,
]
```

```
entity-entry = {
  entity-name => text,
  ? reg-id => any-uri,
  role => $role / [ 2* $role ],
  ? thumbprint => hash-entry,
  global-attributes,
  * $$entity-extension,
}

$role /= tag-creator
$role /= software-creator
$role /= aggregator
$role /= distributor
$role /= licensor
$role /= maintainer
$role /= uint / text

link-entry = {
  ? artifact => text,
  href => any-uri,
  ? media => text,
  ? ownership => $ownership,
  rel => $rel,
  ? media-type => text,
  ? use => $use,
  global-attributes,
  * $$link-extension
}

$ownership /= shared
$ownership /= private
$ownership /= abandon
$ownership /= uint / text

$rel /= ancestor
$rel /= component
$rel /= feature
$rel /= installationmedia
$rel /= packageinstaller
$rel /= parent
$rel /= patches
$rel /= requires
$rel /= see-also
$rel /= supersedes
$rel /= supplemental
$rel /= -256..64436 / text

$use /= optional
```

```
$use /= required
$use /= recommended
$use /= uint / text

software-meta-entry = {
  ? activation-status => text,
  ? channel-type => text,
  ? colloquial-version => text,
  ? description => text,
  ? edition => text,
  ? entitlement-data-required => bool,
  ? entitlement-key => text,
  ? generator => text,
  ? persistent-id => text,
  ? product => text,
  ? product-family => text,
  ? revision => text,
  ? summary => text,
  ? unspsc-code => text,
  ? unspsc-version => text,
  global-attributes,
  * $$software-meta-extension,
}

path-elements-group = ( ? directory => directory-entry / [ 2* directory-entry ],
                        ? file => file-entry / [ 2* file-entry ],
                      )

resource-collection = (
  path-elements-group,
  ? process => process-entry / [ 2* process-entry ],
  ? resource => resource-entry / [ 2* resource-entry ],
  * $$resource-collection-extension,
)

file-entry = {
  filesystem-item,
  ? size => uint,
  ? file-version => text,
  ? hash => hash-entry,
  * $$file-extension,
}

directory-entry = {
  filesystem-item,
  ? path-elements => { path-elements-group },
  * $$directory-extension,
}
```

```
process-entry = {
  process-name => text,
  ? pid => integer,
  global-attributes,
  * $$process-extension,
}

resource-entry = {
  type => text,
  global-attributes,
  * $$resource-extension,
}

filesystem-item = (
  ? key => bool,
  ? location => text,
  fs-name => text,
  ? root => text,
  global-attributes,
)

payload-entry = {
  resource-collection,
  global-attributes,
  * $$payload-extension,
}

evidence-entry = {
  resource-collection,
  ? date => time,
  ? device-id => text,
  global-attributes,
  * $$evidence-extension,
}

; "global map member" integer indexes
tag-id = 0
software-name = 1
entity = 2
evidence = 3
link = 4
software-meta = 5
payload = 6
hash = 7
corpus = 8
patch = 9
media = 10
supplemental = 11
```

```
tag-version = 12
software-version = 13
version-scheme = 14
lang = 15
directory = 16
file = 17
process = 18
resource = 19
size = 20
file-version = 21
key = 22
location = 23
fs-name = 24
root = 25
path-elements = 26
process-name = 27
pid = 28
type = 29
entity-name = 31
reg-id = 32
role = 33
thumbprint = 34
date = 35
device-id = 36
artifact = 37
href = 38
ownership = 39
rel = 40
media-type = 41
use = 42
activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
edition = 47
entitlement-data-required = 48
entitlement-key = 49
generator = 50
persistent-id = 51
product = 52
product-family = 53
revision = 54
summary = 55
unspsc-code = 56
unspsc-version = 57

; "version-scheme" integer indexes
multipartnumeric = 1
```

```
multipartnumeric-suffix = 2
alphanumeric = 3
decimal = 4
semver = 16384

; "role" integer indexes
tag-creator=1
software-creator=2
aggregator=3
distributor=4
licensor=5
maintainer=6

; "ownership" integer indexes
shared=1
private=2
abandon=3

; "rel" integer indexes
ancestor=1
component=2
feature=3
installationmedia=4
packageinstaller=5
parent=6
patches=7
requires=8
see-also=9
supersedes=10
; supplemental=11 ; this is already defined earlier

; "use" integer indexes
optional=1
required=2
recommended=3
<CODE ENDS>
```

3. Determining the Type of CoSWID

The operational model for SWID and CoSWID tags was introduced in Section 1.1, which described four different CoSWID tag types. The following additional rules apply to the use of CoSWID tags to ensure that created tags properly identify the tag type.

The first matching rule MUST determine the type of the CoSWID tag.

1. Primary Tag: A CoSWID tag MUST be considered a primary tag if the corpus, patch, and supplemental items are "false".

2. Supplemental Tag: A CoSWID tag MUST be considered a supplemental tag if the supplemental item is set to "true".

3. Corpus Tag: A CoSWID tag MUST be considered a corpus tag if the corpus item is "true".

4. Patch Tag: A CoSWID tag MUST be considered a patch tag if the patch item is "true".

Note: Multiple of the corpus, patch, and supplemental items can have values set as "true". The rules above provide a means to determine the tag's type in such a case. For example, a SWID or CoSWID tag for a patch installer might have both corpus and patch items set to "true". In such a case, the tag is a "Corpus Tag". The tag installed by this installer would have only the patch item set to "true", making the installed tag type a "Patch Tag".

4. CoSWID Indexed Label Values

4.1. Version Scheme

The following table contains a set of values for use in the concise-swid-tag group's version-scheme item. These values match the version schemes defined in the ISO/IEC 19770-2:2015 [SWID] specification. Index value indicates the value to use as the version-scheme item's value. The Version Scheme Name provides human-readable text for the value. The Definition describes the syntax of allowed values for each entry.

Index	Version Scheme Name	Definition
1	multipartnumeric	Numbers separated by dots, where the numbers are interpreted as integers (e.g., 1.2.3, 1.4.5, 1.2.3.4.5.6.7)
2	multipartnumeric+suffix	Numbers separated by dots, where the numbers are interpreted as integers with an additional textual suffix (e.g., 1.2.3a)
3	alphanumeric	Strictly a string, sorting is done alphanumerically
4	decimal	A floating point number (e.g., 1.25 is less than 1.3)
16384	semver	Follows the [SEMVER] specification

Table 3: Version Scheme Values

[TODO: What text do we need to include to get a waiver to use SEMVER as a normative requirement?]

The values above are registered in the IANA "Software Tag Version Scheme Values" registry defined in Section Section 5.2.4. Additional entries will likely be registered over time in this registry.

These version schemes have partially overlapping value spaces. The following guidelines help to ensure that the most specific version-scheme is used:

- o "decimal" and "multipartnumeric" partially overlap in their value space when a value matches a decimal number. When a corresponding software-version item's value falls within this overlapping value space, the "decimal" version scheme SHOULD be used.

- o "multipartnumeric" and "semver" partially overlap in their value space when a "multipartnumeric" value matches the semantic versioning syntax. When a corresponding software-version item's value falls within this overlapping value space, the "semver" version scheme SHOULD be used.

- o "alphanumeric" and other version schemes might overlap in their value space. When a corresponding software-version item's value falls within this overlapping value space, the other version scheme SHOULD be used instead of "alphanumeric".

4.2. Entity Role Values

The following table indicates the index value to use for the entity-entry group's role item (see Section 2.6). These values match the entity roles defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the role item's value. The "Role Name" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.

Index	Role Name	Definition
1	tagCreator	The person or organization that created the containing SWID or CoSWID tag
2	softwareCreator	The person or organization entity that created the software component.
3	aggregator	From [SWID], "An organization or system that encapsulates software from their own and/or other organizations into a different distribution process (as in the case of virtualization), or as a completed system to accomplish a specific task (as in the case of a value added reseller)."
4	distributor	From [SWID], "An entity that furthers the marketing, selling and/or distribution of software from the original place of manufacture to the ultimate user without modifying the software, its packaging or its labelling."
5	licensor	From [SAM] as "software licensor", a "person or organization who owns or holds the rights to issue a software license for a specific software [component]"
6	maintainer	The person or organization that is responsible for coordinating and making updates to the source code for the software component. This SHOULD be used when the "maintainer" is a different person or organization than the original "softwareCreator".

Table 4: Entity Role Values

The values above are registered in the IANA "Software Tag Entity Role Values" registry defined in Section Section 5.2.5. Additional values will likely be registered over time. Additionally, the index values 128 through 255 and the name prefix "x_" have been reserved for private use.

4.3. Link Ownership Values

The following table indicates the index value to use for the link-entry group's ownership item (see Section 2.7). These values match the link ownership values defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the link-entry group ownership item's value. The "Ownership Type" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.

Index	Ownership Type	Definition
1	abandon	If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD NOT be uninstalled
2	private	If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD be uninstalled as well.
3	shared	If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD be uninstalled if no other components sharing the software.

Table 5: Link Ownership Values

The values above are registered in the IANA "Software Tag Link Ownership Values" registry defined in Section Section 5.2.6. Additional values will likely be registered over time. Additionally, the index values 128 through 255 and the name prefix "x_" have been reserved for private use.

4.4. Link Rel Values

The following table indicates the index value to use for the link-entry group's rel item (see Section 2.7). These values match the link rel values defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the link-entry group ownership item's value. The "Relationship Type" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.

+-------+-------------------+-------------------------------------+

Index	Relationship Type	Definition
1	ancestor	The link references a software tag for a previous release of this software. This can be useful to define an upgrade path.
2	component	The link references a software tag for a separate component of this software.
3	feature	The link references a configurable feature of this software that can be enabled or disabled without changing the installed files.
4	installationmedia	The link references the installation package that can be used to install this software.
5	packageinstaller	The link references the installation software needed to install this software.
6	parent	The link references a software tag that is the parent of the referencing tag. This relationship can be used when multiple software components are part of a software bundle, where the "parent" is the software tag for the bundle, and each child is a "component". In such a case, each child component can provide a "parent" link relationship to the bundle's software tag, and the bundle can provide a "component" link relationship to each child software component.
7	patches	The link references a software tag that the referencing software patches. Typically only used for patch tags (see Section 1.1).
8	requires	The link references a prerequisite for installing this software. A patch tag (see Section 1.1) can use this to represent base software or another

		patch that needs to be installed first.
9	see-also	The link references other software that may be of interest that relates to this software.
10	supersedes	The link references another software that this software replaces. A patch tag (see Section 1.1) can use this to represent another patch that this patch incorporates or replaces.
11	supplemental	The link references a software tag that the referencing tag supplements. Used on supplemental tags (see Section 1.1).

Table 6: Link Relationship Values

The values above are registered in the IANA "Software Tag Link Relationship Values" registry defined in Section Section 5.2.7. Additional values will likely be registered over time. Additionally, the index values 32768 through 65535 and the name prefix "x_" have been reserved for private use.

4.5. Link Use Values

The following table indicates the index value to use for the link-entry group's use item (see Section 2.7). These values match the link use values defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the link-entry group use item's value. The "Use Type" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.

Index	Use Type	Definition
1	optional	From [SWID], "Not absolutely required; the [Link]'d software is installed only when specified."
2	required	From [SWID], "The [Link]'d software is absolutely required for an operation software installation."
3	recommended	From [SWID], "Not absolutely required; the [Link]'d software is installed unless specified otherwise."

Table 7: Link Use Values

The values above are registered in the IANA "Software Tag Link Use Values" registry defined in Section Section 5.2.8. Additional values will likely be registered over time. Additionally, the index values 128 through 255 and the name prefix "x_" have been reserved for private use.

5. IANA Considerations

This document has a number of IANA considerations, as described in the following subsections. In summary, 6 new registries are established with this request, with initial entries provided for each registry. New values for 5 other registries are also requested.

5.1. CoSWID Items Registry

This registry uses integer values as index values in CBOR maps.

This document defines a new registry titled "CoSWID Items". Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-32767	Standards Action
32768-4294967295	Specification Required

Table 8: CoSWID Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoSWID Items" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	tag-id	RFC-AAAA
1	software-name	RFC-AAAA
2	entity	RFC-AAAA
3	evidence	RFC-AAAA
4	link	RFC-AAAA
5	software-meta	RFC-AAAA
6	payload	RFC-AAAA
7	hash	RFC-AAAA
8	corpus	RFC-AAAA
9	patch	RFC-AAAA
10	media	RFC-AAAA
11	supplemental	RFC-AAAA
12	tag-version	RFC-AAAA
13	software-version	RFC-AAAA
14	version-scheme	RFC-AAAA
15	lang	RFC-AAAA
16	directory	RFC-AAAA
17	file	RFC-AAAA
18	process	RFC-AAAA
19	resource	RFC-AAAA

20	size	RFC-AAAA
21	file-version	RFC-AAAA
22	key	RFC-AAAA
23	location	RFC-AAAA
24	fs-name	RFC-AAAA
25	root	RFC-AAAA
26	path-elements	RFC-AAAA
27	process-name	RFC-AAAA
28	pid	RFC-AAAA
29	type	RFC-AAAA
31	entity-name	RFC-AAAA
32	reg-id	RFC-AAAA
33	role	RFC-AAAA
34	thumbprint	RFC-AAAA
35	date	RFC-AAAA
36	device-id	RFC-AAAA
37	artifact	RFC-AAAA
38	href	RFC-AAAA
39	ownership	RFC-AAAA
40	rel	RFC-AAAA
41	media-type	RFC-AAAA
42	use	RFC-AAAA
43	activation-status	RFC-AAAA
44	channel-type	RFC-AAAA

45	colloquial-version	RFC-AAAA
46	description	RFC-AAAA
47	edition	RFC-AAAA
48	entitlement-data-required	RFC-AAAA
49	entitlement-key	RFC-AAAA
50	generator	RFC-AAAA
51	persistent-id	RFC-AAAA
52	product	RFC-AAAA
53	product-family	RFC-AAAA
54	revision	RFC-AAAA
55	summary	RFC-AAAA
56	unspsc-code	RFC-AAAA
57	unspsc-version	RFC-AAAA
58-4294967295	Unassigned	

Table 9: CoSWID Items Inital Registrations

5.2. Software Tag Values Registries

The following IANA registries provide a mechanism for new values to be added over time to common enumerations used by SWID and CoSWID.

5.2.1. Registration Procedures

The following registries allow for the registration of index values and names. New registrations will be permitted through either the Standards Action policy or the Specification Required policy [BCP26]. New index values will be provided on a First Come First Served as defined by [BCP26].

The following registries also reserve the integer-based index values in the range of -1 to -256 for private use as defined by [BCP26] in Section 4.1. This allows values -1 to -24 to be expressed as a single uint_8t in CBOR, and values -25 to -256 to be expressed using an additional uint_8t in CBOR.

5.2.2. Private Use of Index and Name Values

The integer-based index values in the private use range (-1 to -256) are intended for testing purposes and closed environments; values in other ranges SHOULD NOT be assigned for testing.

For names that correspond to private use index values, an Internationalized Domain Name prefix MUST be used to prevent name conflicts using the form:

" domain.prefix-name "

Where "domain.prefix" MUST be a valid Internationalized Domain Name as defined by [RFC5892], and "name" MUST be a unique name within the namespace defined by the "domain.prefix". Use of a prefix in this way allows for a name to be used initially in the private use range, and to be registered at a future point in time. This is consistent with the guidance in [BCP178].

5.2.3. Expert Review Guidelines

Designated experts MUST ensure that new registration requests meet the following additional guidelines:

- The requesting specification MUST provide a clear semantic definition for the new entry. This definition MUST clearly differentiate the requested entry from other previously registered entries.

- The requesting specification MUST describe the intended use of the entry, including any co-constraints that exist between the use of the entry's index value or name, and other values defined within the SWID/CoSWID model.

- Index values and names outside the private use space MUST NOT be used without registration. This is considered squatting and SHOULD be avoided. Designated experts MUST ensure that reviewed specifications register all appropriate index values and names.

- Standards track documents MAY include entries registered in the range reserved for entries under the Specification Required policy. This can occur when a standards track document provides further guidance on the use of index values and names that are in common use, but were not registered with IANA. This situation SHOULD be avoided.

- o All registered names MUST be valid according to the XML Schema NMTOKEN data type (see [W3C.REC-xmlschema-2-20041028] Section 3.3.4). This ensures that registered names are compatible with the SWID format [SWID] where they are used.

- o Registration of vanity names SHOULD be discouraged. The requesting specification MUST provide a description of how a requested name will allow for use by multiple stakeholders.

5.2.4. Software Tag Version Scheme Values Registry

This document establishes a new registry titled "Software Tag Version Scheme Values". This registry provides index values for use as version-scheme item values in this document and version scheme names for use in [SWID].

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-16383	Standards Action
16384-65535	Specification Required

Table 10: CoSWID Version Scheme Registration Procedures

Assignments MUST consist of an integer Index value, the Version Scheme Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Version Scheme Values" registry are provided below, which are derived from the textual version scheme names defined in [SWID].

Index	Version Scheme Name	Specification
0	Reserved	
1	multipartnumeric	See Section 4.1
2	multipartnumeric+suffix	See Section 4.1
3	alphanumeric	See Section 4.1
4	decimal	See Section 4.1
5-16383	Unassigned	
16384	semver	[SEMVER]
16385-65535	Unassigned	

Table 11: CoSWID Version Scheme Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

Designated experts MUST also ensure that newly requested entries define a value space for the corresponding version item that is unique from other previously registered entries. Note: The initial registrations violate this requirement, but are included for backwards compatibility with [SWID]. Guidelines on how to deconflict these value spaces are defined in Section Section 4.1.

5.2.5. Software Tag Entity Role Values Registry

This document establishes a new registry titled "Software Tag Entity Role Values". This registry provides index values for use as entity-entry role item values in this document and entity role names for use in [SWID].

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 12: CoSWID Entity Role Registration Procedures

Assignments consist of an integer Index value, a Role Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Entity Role Values" registry are provided below, which are derived from the textual entity role names defined in [SWID].

Index	Role Name	Specification
0	Reserved	
1	tagCreator	See Section 4.2
2	softwareCreator	See Section 4.2
3	aggregator	See Section 4.2
4	distributor	See Section 4.2
5	licensor	See Section 4.2
6	maintainer	See Section 4.2
7-255	Unassigned	

Table 13: CoSWID Entity Role Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

5.2.6. Software Tag Link Ownership Values Registry

This document establishes a new registry titled "Software Tag Link Ownership Values". This registry provides index values for use as link-entry ownership item values in this document and link ownership names for use in [SWID].

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 14: CoSWID Link Ownership Registration Procedures

Assignments consist of an integer Index value, an Ownership Type Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Link Ownership Values" registry are provided below, which are derived from the textual entity role names defined in [SWID].

Index	Ownership Type Name	Definition
0	Reserved	
1	abandon	See Section 4.3
2	private	See Section 4.3
3	shared	See Section 4.3
4-255	Unassigned	

Table 15: CoSWID Link Ownership Inital Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

5.2.7. Software Tag Link Relationship Values Registry

This document establishes a new registry titled "Software Tag Link Relationship Values". This registry provides index values for use as link-entry rel item values in this document and link ownership names for use in [SWID].

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-32767	Standards Action
32768-65535	Specification Required

Table 16: CoSWID Link Relationship Registration Procedures

Assignments consist of an integer Index value, the Relationship Type Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Link Relationship Values" registry are provided below, which are derived from the link relationship values defined in [SWID].

Index	Relationship Type Name	Specification
0	Reserved	
1	ancestor	See Section 4.4
2	component	See Section 4.4
3	feature	See Section 4.4
4	installationmedia	See Section 4.4
5	packageinstaller	See Section 4.4
6	parent	See Section 4.4
7	patches	See Section 4.4
8	requires	See Section 4.4
9	see-also	See Section 4.4
10	supersedes	See Section 4.4
11	supplemental	See Section 4.4
12-65535	Unassigned	

Table 17: CoSWID Link Relationship Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

Designated experts MUST also ensure that a newly requested entry documents the URI schemes allowed to be used in an href associated with the link relationship and the expected resolution behavior of these URI schemes. This will help to ensure that applications processing software tags are able to interoperate when resolving resources referenced by a link of a given type.

5.2.8. Software Tag Link Use Values Registry

This document establishes a new registry titled "Software Tag Link Use Values". This registry provides index values for use as link-entry use item values in this document and link use names for use in [SWID].

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 18: CoSWID Link Use Registration Procedures

Assignments consist of an integer Index value, the Link Use Type Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Link Use Values" registry are provided below, which are derived from the link relationship values defined in [SWID].

Index	Link Use Type Name	Specification
0	Reserved	
1	optional	See Section 4.5
2	required	See Section 4.5
3	recommended	See Section 4.5
4-255	Unassigned	

Table 19: CoSWID Link Use Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

5.3. swid+cbor Media Type Registration

[TODO: Per Section 5.1 of RFC6838, was a message sent to media-types@iana.org for preliminary review? I didn't see it on that mailing list (did I miss it?). Please kick that off.]

IANA is requested to add the following to the IANA "Media Types" registry.

Type name: application

Subtype name: swid+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: Must be encoded as using [RFC7049]. See RFC-AAAA for details.

Security considerations: See Section 6 of RFC-AAAA.

Interoperability considerations: Applications MAY ignore any key value pairs that they do not understand. This allows backwards compatible extensions to this specification.

Published specification: RFC-AAAA

Applications that use this media type: The type is used by software asset management systems, vulnerability assessment systems, and in applications that use remote integrity verification.

Fragment identifier considerations: Fragment identification for application/swid+cbor is supported by using fragment identifiers as specified by RFC7049 Section 7.5.

Additional information:

Magic number(s): first five bytes in hex: da 53 57 49 44

File extension(s): coswid

Macintosh file type code(s): none

Macintosh Universal Type Identifier code: org.ietf.coswid conforms to public.data

Person & email address to contact for further information: Henk Birkholz <henk.birkholz@sit.fraunhofer.de>

Intended usage: COMMON

Restrictions on usage: None

Author: Henk Birkholz <henk.birkholz@sit.fraunhofer.de>

Change controller: IESG

5.4. CoAP Content-Format Registration

IANA is requested to assign a CoAP Content-Format ID for the CoSWID media type in the "CoAP Content-Formats" sub-registry, from the "IETF Review or IESG Approval" space (256..999), within the "CoRE Parameters" registry [RFC7252]:

Media type	Encoding	ID	Reference
application/swid+cbor	-	TBD1	RFC-AAAA

Table 20: CoAP Content-Format IDs

5.5. CBOR Tag Registration

IANA is requested to allocate a tag in the "CBOR Tags" registry, preferably with the specific value requested:

Tag	Data Item	Semantics
1398229316	map	Concise Software Identifier (CoSWID) [RFC-AAAA]

Table 21: CoSWID CBOR Tag

5.6. URI Scheme Registrations

The ISO 19770-2:2015 SWID specification describes use of the "swid" and "swidpath" URI schemes, which are currently in use in implementations. This document continues this use for CoSWID. The following subsections provide registrations for these schemes in to ensure that a permanent registration exists for these schemes that is suitable for use in the SWID and CoSWID specifications.

[TODO: Per Step 3.2 of Section 7.2 of RFC7595, has this been sent to uri-review@ietf.org? I didn't see it on that mailing list (did I miss it?). Please kick that off.]

5.6.1. "swid" URI Scheme Registration

There is a need for a scheme name that can be used in URIs that point to a specific software tag by that tag's tag-id, such as the use of the link entry as described in Section Section 2.7) of this document. Since this scheme is used in a standards track document and an ISO standard, this scheme needs to be used without fear of conflicts with current or future actual schemes. The scheme "swid" is hereby registered as a 'permanent' scheme for that purpose.

The "swid" scheme is specified as follows:

Scheme name: FIXME

Status: Permanent

Applications/protocols that use this scheme name: FIXME

Contact: FIXME

Change controller: FIXME

References: FIXME

5.6.2. "swid" URI Scheme Specification [TODO: FIXME: has to move out of registration]

Scheme syntax: The scheme specific part consists of a software tag's tag-id that is URI encoded according to [RFC3986] Section 2.1. The following expression is a valid example:

<swid:2df9de35-0aff-4a86-ace6-f7dddd1ade4c>

Scheme semantics: URIs in the "swid" scheme are to be used to reference a software tag by its tag-id. A tag-id referenced in this way can be used to identify the tag resource in the context of where it is referenced from. For example, when a tag is installed on a given device, that tag can reference related tags on the same device using this URI scheme.

Encoding considerations: See Section 2.5 of [RFC3986] for guidelines.

Interoperability considerations: None.

Security considerations: None.

5.6.3. "swidpath" URI Scheme Registration

There is a need for a scheme name that can be used in URIs to identify a collection of specific software tags with data elements that match an XPath expression, such as the use of the link entry as described in Section Section 2.7) of this document. Since this scheme is used in a standards track document and an ISO standard, this scheme needs to be used without fear of conflicts with current or future actual schemes. The scheme "swidpath" is hereby registered as a 'permanent' scheme for that purpose.

The "swidpath" scheme is specified as follows:

Scheme name: FIXME

Status: Permanent

Applications/protocols that use this scheme name: FIXE

Contact: FIXME

Change controller: FIXME

References: FIXME

5.6.4. "swidpath" URI Scheme Specification [TODO: FIXME: has to move out of registration]

Scheme syntax: The scheme specific part consists of an XPath expression as defined by [W3C.REC-xpath20-20101214]. The included XPath expression will be URI encoded according to [RFC3986] Section 2.1.

Scheme semantics: URIs in the "swidpath" scheme are to be used specify the data that must be found in a given software tag for that tag to be considered a matching tag to be included in the identified tag collection. Tags to be evaluated include all tags in the context of where the tag is referenced from. For example, when a tag is installed on a given device, that tag can reference related tags on the same device using this URI scheme. A tag is matching if the XPath evaluation result value has an effective boolean value of "true" according to [W3C.REC-xpath20-20101214] Section 2.4.3. rence related tags on the same device using this URI scheme.

Encoding considerations: See Section 2.5 of [RFC3986] for guidelines.

Interoperability considerations: None.

Security considerations: None.

5.7. CoSWID Model for use in SWIMA Registration

The Software Inventory Message and Attributes (SWIMA) for PA-TNC specification [RFC8412] defines a standardized method for collecting an endpoint device's software inventory. A CoSWID can provide evidence of software installation which can then be used and exchanged with SWIMA. This registration adds a new entry to the IANA "Software Data Model Types" registry defined by [RFC8412] to support CoSWID use in SWIMA as follows:

Pen: 0

Integer: TBD2

Name: Concise Software Identifier (CoSWID)

Defining Specification: RFC-AAAA

Deriving Software Identifiers:

A Software Identifier generated from a CoSWID tag is expressed as a concatenation of the form:

TAG_CREATOR_REGID "_" "_" UNIQUE_ID

Where TAG_CREATOR_REGID is the reg-id item value of the tag's entity item having the role value of 1 (corresponding to "tag creator"), and the UNIQUE_ID is the same tag's tag-id item. If the tag-id item's value is expressed as a 16 byte binary string, the UNIQUE_ID MUST be represented using the UUID string representation defined in [RFC4122] including the "urn:uuid:" prefix.

The TAG_CREATOR_REGID and the UNIQUE_ID are connected with a double underscore (_), without any other connecting character or whitespace.

6. Security Considerations

CoSWID tags contain public information about software components and, as such, do not need to be protected against disclosure on an endpoint. Similarly, CoSWID tags are intended to be easily discoverable by applications and users on an endpoint in order to make it easy to identify and collect all of an endpoint's SWID tags. As such, any security considerations regarding CoSWID tags focus on the application of CoSWID tags to address security challenges, and the possible disclosure of the results of those applications.

A tag is considered "authoritative" if the CoSWID tag was created by the software provider. An authoritative CoSWID tag contains information about a software component provided by the maintainer of the software component, who is expected to be an expert in their own software. Thus, authoritative CoSWID tags can be trusted to represent authoritative information about the software component.

A signed CoSWID tag (see Appendix A) whose signature has been validated can be relied upon to be unchanged since it was signed. By contrast, the data contained in unsigned tags cannot be trusted to be unmodified.

When an authoritative tag is signed, the software provider can be authenticated as the originator of the signature. A trustworthy association between the signature and the originator of the signature can be established via trust anchors. A certification path between a trust anchor and a certificate including a pub-key enabling the validation of a tag signature can realize the assessment of trustworthiness of an authoritative tag. Having a signed authoritative CoSWID tag can be useful when the information in the tag needs to be trusted, such as when the tag is being used to convey reference integrity measurements for software components.

CoSWID tags are designed to be easily added and removed from an endpoint along with the installation or removal of software components. On endpoints where addition or removal of software components is tightly controlled, the addition or removal of SWID tags can be similarly controlled. On more open systems, where many users can manage the software inventory, CoSWID tags can be easier to add or remove. On such systems, it can be possible to add or remove CoSWID tags in a way that does not reflect the actual presence or absence of corresponding software components. Similarly, not all software products automatically install CoSWID tags, so products can be present on an endpoint without providing a corresponding SWID tag. As such, any collection of CoSWID tags cannot automatically be assumed to represent either a complete or fully accurate representation of the software inventory of the endpoint. However, especially on endpoint devices that more strictly control the ability to add or remove applications, CoSWID tags are an easy way to provide an preliminary understanding of that endpoint's software inventory.

Any report of an endpoint's CoSWID tag collection provides information about the software inventory of that endpoint. If such a report is exposed to an attacker, this can tell them which software products and versions thereof are present on the endpoint. By examining this list, the attacker might learn of the presence of applications that are vulnerable to certain types of attacks. As noted earlier, CoSWID tags are designed to be easily discoverable by an endpoint, but this does not present a significant risk since an attacker would already need to have access to the endpoint to view that information. However, when the endpoint transmits its software inventory to another party, or that inventory is stored on a server for later analysis, this can potentially expose this information to attackers who do not yet have access to the endpoint. For this reason, it is important to protect the confidentiality of CoSWID tag information that has been collected from an endpoint in transit and at rest, not because those tags individually contain sensitive information, but because the collection of CoSWID tags and their association with an endpoint reveals information about that endpoint's attack surface.

Finally, both the ISO-19770-2:2015 XML schema SWID definition and the CoSWID CDDL specification allow for the construction of "infinite" tags with link item loops or tags that contain malicious content with the intent of creating non-deterministic states during validation or processing of those tags. While software providers are unlikely to do this, CoSWID tags can be created by any party and the CoSWID tags collected from an endpoint could contain a mixture of vendor and non-vendor created tags. For this reason, a CoSWID tag might contain potentially malicious content. Input sanitization and loop detection are two ways that implementations can address this concern.

7. Acknowledgments

This document draws heavily on the concepts defined in the ISO/IEC 19770-2:2015 specification. The authors of this document are grateful for the prior work of the 19770-2 contributors.

We are also grateful to the careful reviews provided by ...

8. Change Log

[THIS SECTION TO BE REMOVED BY THE RFC EDITOR.]

Changes from version 12 to version 14:

- Moved key identifier to protected COSE header
- Fixed index reference for hash
- Removed indirection of CDDL type definition for filesystem-item
- Fixed quantity of resource and process
- Updated resource-collection

- o Renamed socket name in software-meta to be consistent in naming

- o Aligned excerpt examples in I-D text with full CDDL

- o Fixed titels where title was referring to group instead of map

- o Added missig date in SEMVER

- o Fixed root cardinality for file and directory, etc.

- o Transformed path-elements-entry from map to group for re-usability

- o Scrubbed IANA Section

- o Removed redundant supplemental rule

- o Aligned discrepancy with ISO spec.

- o Addressed comments on typos.

- o Fixed kramdown nits and BCP reference.

- o Addressed comments from WGLC reviewers.

Changes in version 12:

- o Addressed a bunch of minor editorial issues based on WGLC feedback.

- o Added text about the use of UTF-8 in CoSWID.

- o Adjusted tag-id to allow for a UUID to be provided as a bstr.

- o Cleaned up descriptions of index ranges throughout the document, removing discussion of 8 bit, 16 bit, etc.

- o Adjusted discussion of private use ranges to use negative integer values and to be more clear throughout the document.

- o Added discussion around resolving overlapping value spaces for version schemes.

- o Added a set of expert review guidelines for new IANA registries created by this document.

- o Added new registrations for the "swid" and "swidpath" URI schemes, and for using CoSWID with SWIMA.

Changes from version 03 to version 11:

- o Reduced representation complexity of the media-entry type and removed the Section describing the older data structure.
- o Added more signature schemes from COSE
- o Included a minimal required set of normative language
- o Reordering of attribute name to integer label by priority according to semantics.
- o Added an IANA registry for CoSWID items supporting future extension.
- o Cleaned up IANA registrations, fixing some inconsistencies in the table labels.
- o Added additional CDDL sockets for resource collection entries providing for additional extension points to address future SWID/ CoSWID extensions.
- o Updated Section on extension points to address new CDDL sockets and to reference the new IANA registry for items.
- o Removed unused references and added new references to address placeholder comments.
- o Added table with semantics for the link ownership item.
- o Clarified language, made term use more consistent, fixed references, and replacing lowercase RFC2119 keywords.

Changes from version 02 to version 03:

- o Updated core CDDL including the CDDL design pattern according to RFC 8428.

Changes from version 01 to version 02:

- o Enforced a more strict separation between the core CoSWID definition and additional usage by moving content to corresponding appendices.
- o Removed artifacts inherited from the reference schema provided by ISO (e.g. NMTOKEN(S))

- o Simplified the core data definition by removing group and type choices where possible

- o Minor reordering of map members

- o Added a first extension point to address requested flexibility for extensions beyond the any-element

Changes from version 00 to version 01:

- o Ambiguity between evidence and payload eliminated by introducing explicit members (while still

- o allowing for "empty" SWID tags)

- o Added a relatively restrictive COSE envelope using cose_sign1 to define signed CoSWID (single signer only, at the moment)

- o Added a definition how to encode hashes that can be stored in the any-member using existing IANA tables to reference hash-algorithms

Changes since adopted as a WG I-D -00:

- o Removed redundant any-attributes originating from the ISO-19770-2:2015 XML schema definition

- o Fixed broken multi-map members

- o Introduced a more restrictive item (any-element-map) to represent custom maps, increased restriction on types for the any-attribute, accordingly

- o Fixed X.1520 reference

- o Minor type changes of some attributes (e.g. NMTOKENS)

- o Added semantic differentiation of various name types (e,g. fs-name)

Changes from version 06 to version 07:

- o Added type choices/enumerations based on textual definitions in 19770-2:2015

- o Added value registry request

- o Added media type registration request

- Added content format registration request
- Added CBOR tag registration request
- Removed RIM appendix to be addressed in complementary draft
- Removed CWT appendix
- Flagged firmware resource collection appendix for revision
- Made use of terminology more consistent
- Better defined use of extension points in the CDDL
- Added definitions for indexed values
- Added IANA registry for Link use indexed values

Changes from version 05 to version 06:

- Improved quantities
- Included proposals for implicit enumerations that were NMTOKENS
- Added extension points
- Improved exemplary firmware-resource extension

Changes from version 04 to version 05:

- Clarified language around SWID and CoSWID to make more consistent use of these terms.
- Added language describing CBOR optimizations for single vs. arrays in the model front matter.
- Fixed a number of grammatical, spelling, and wording issues.
- Documented extension points that use CDDL sockets.
- Converted IANA registration tables to markdown tables, reserving the 0 value for use when a value is not known.
- Updated a number of references to their current versions.

Changes from version 03 to version 04:

- Re-index label values in the CDDL.

- o Added a Section describing the CoSWID model in detail.
- o Created IANA registries for entity-role and version-scheme

Changes from version 02 to version 03:

- o Updated CDDL to allow for a choice between a payload or evidence
- o Re-index label values in the CDDL.
- o Added item definitions
- o Updated references for COSE, CBOR Web Token, and CDDL.

Changes from version 01 to version 02:

- o Added extensions for Firmware and CoSWID use as Reference Integrity Measurements (CoSWID RIM)
- o Changes meta handling in CDDL from use of an explicit use of items to a more flexible unconstrained collection of items.
- o Added Sections discussing use of COSE Signatures and CBOR Web Tokens

Changes from version 00 to version 01:

- o Added CWT usage for absolute SWID paths on a device
- o Fixed cardinality of type-choices including arrays
- o Included first iteration of firmware resource-collection

9. References

9.1. Normative References

[BCP178] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, DOI 10.17487/RFC6648, June 2012, <https://www.rfc-editor.org/info/rfc6648>.

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[SAM] "Information technology - Software asset management - Part 5: Overview and vocabulary", ISO/IEC 19770-5:2015, November 2013.

[SEMVER] Preston-Werner, T., "Semantic Versioning 2.0.0", <https://semver.org/spec/v2.0.0.html>.

[SWID] "Information technology - Software asset management - Part 2: Software identification tag", ISO/IEC 19770-2:2015, October 2015.

[UNSPSC] "United Nations Standard Products and Services Code", October 2020, <https://www.unspsc.org/>.

[W3C.REC-css3-mediaqueries-20120619]
Rivoal, F., "Media Queries", World Wide Web Consortium Recommendation REC-css3-mediaqueries-20120619, June 2012, <https://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619>.

[W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

[W3C.REC-xpath20-20101214]
Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., and J. Simeon, "XML Path Language (XPath) 2.0 (Second Edition)", World Wide Web Consortium Recommendation REC-xpath20-20101214, December 2010, <https://www.w3.org/TR/2010/REC-xpath20-20101214>.

[X.1520] "Recommendation ITU-T X.1520 (2014), Common vulnerabilities and exposures", April 2011.

9.2. Informative References

[CamelCase]
"UpperCamelCase", August 2014, <http://wiki.c2.com/?CamelCase>.

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-07 (work in progress), October 2020.

[KebabCase]
"KebabCase", December 2014, <http://wiki.c2.com/?KebabCase>.

[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>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[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>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[SWID-GUIDANCE]
Waltermire, D., Cheikes, B., Feldman, L., and G. Witte, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", NISTIR 8060, April 2016, <https://doi.org/10.6028/NIST.IR.8060>.

Appendix A. Signed Concise SWID Tags using COSE

SWID tags, as defined in the ISO-19770-2:2015 XML schema, can include cryptographic signatures to protect the integrity of the SWID tag. In general, tags are signed by the tag creator (typically, although not exclusively, the vendor of the software component that the SWID tag identifies). Cryptographic signatures can make any modification of the tag detectable, which is especially important if the integrity of the tag is important, such as when the tag is providing reference integrity measurements for files.

The ISO-19770-2:2015 XML schema uses XML DSIG to support cryptographic signatures. CoSWID tags require a different signature scheme than this. COSE (CBOR Object Signing and Encryption) provides the required mechanism [RFC8152]. Concise SWID can be wrapped in a COSE Single Signer Data Object (COSE_Sign1) that contains a single signature. The following CDDL defines a more restrictive subset of header attributes allowed by COSE tailored to suit the requirements of Concise SWID tags.

```
<CODE BEGINS>
signed-coswid = #6.18(COSE-Sign1-coswid)

cose-label = int / tstr
cose-values = any

protected-signed-coswid-header = {
    1 => int,                    ; algorithm identifier
    3 => "application/swid+cbor",
    4 => bstr,                   ; key identifier
    * cose-label => cose-values,
}

unprotected-signed-coswid-header = {
    * cose-label => cose-values,
}

COSE-Sign1-coswid = [
    protected: bstr .cbor protected-signed-coswid-header,
    unprotected: unprotected-signed-coswid-header,
    payload: bstr .cbor concise-swid-tag,
    signature: bstr,
]
<CODE ENDS>
```

Optionally, the COSE_Sign structure that allows for more than one signature to be applied to a CoSWID tag MAY be used. The corresponding usage scenarios are domain-specific and require well-defined application guidance. Representation of the corresponding guidance is out-of-scope of this document.

Additionally, the COSE Header counter signature MAY be used as an attribute in the unprotected header map of the COSE envelope of a CoSWID. The application of counter signing enables second parties to provide a signature on a signature allowing for a proof that a signature existed at a given time (i.e., a timestamp).

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Jessica Fitzgerald-McKay
Department of Defense
9800 Savage Road
Ft. Meade, Maryland
USA

Email: jmfitz2@nsa.gov

Charles Schmidt
The MITRE Corporation
202 Burlington Road
Bedford, Maryland 01730
USA

Email: cmschmidt@mitre.org

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

Email: david.waltermire@nist.gov

SACM D. Haynes
Internet-Draft The MITRE Corporation
Intended status: Best Current Practice J. Fitzgerald-McKay
Expires: December 23, 2019 Department of Defense
L. Lorenzin
Pulse Secure
June 21, 2019

Endpoint Posture Collection Profile
draft-ietf-sacm-ecp-05

Abstract

This document specifies the Endpoint Posture Collection Profile, which describes the best practices for the application of IETF, TNC, and ISO/IEC data models, protocols, and interfaces to support the on-going collection and communication of endpoint posture to a centralized server where it can be stored and made available to other tools. This document is an extension of the Trusted Computing Group's Endpoint Compliance Profile Version 1.0 specification [ECP].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Endpoint Posture Collection Profile (EPCP) builds on prior work from the IETF NEA WG, the IETF NETCONF WG, IETF NETMOD WG, the Trusted Computing Group (TCG) Trusted Network Communications [TNC] WG, and the International Organization for Standardization/International Electrotechnical Commission Joint Technical Committee (JTC) 1, Subcommittee (SC) 7, WG 21 (ISO/IEC JTC 1, SC7, WG21) to describe the best practices for the collection and communication of posture information from network-connected endpoints to a centralized server.

This document focuses on reducing the security exposure of a network by enabling event-driven posture collection, standardized querying of additional posture information as needed, and the communication of that data to a centralized server where it can made available to other components. Thus, eliminating the need for redundant collection and agents on endpoints. Future revisions of this document may include support for the collection of posture information from other endpoint types as well as a standardized interface for storing and querying data in repositories among other capabilities. Additional information about this future work can be found in Section 6 of this document.

To support the collection of posture information from new endpoint types, this document is organized such that it first provides a high-level overview of EPCP as well as its abstract architectural components and transactions that will be realized by implementations (Section 3). This is followed by individual sections that discuss the best practices for specific implementations of the EPCP for a given endpoint type (e.g., traditional, network device, etc.) along with any extensions for supported use cases (software asset management, vulnerability management, etc.).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

Furthermore, this document uses terms as defined in [I-D.ietf-sacm-terminology] unless otherwise specified.

3. Endpoint Posture Collection Profile

The EPCP describes how IETF, TCG, and ISO/IEC data models, protocols, and interfaces can be used to support the posture assessment of endpoints on a network. This profile does not generate new data models, protocols, or interfaces; rather, it offers best practices for a full end-to-end solution for posture assessment, as well as a fresh perspective on how existing standards can be leveraged against vulnerabilities. Rationale for the EPCP solution as well as the supported and non-supported use cases is available in Appendix A and Appendix B respectively.

The EPCP makes it possible to perform posture assessments against all network-connected endpoints by:

1. uniquely identifying the endpoint;

2. collecting and evaluating posture based on data from the endpoint (asset management, software asset management, vulnerability management, and configuration management);

3. creating a secure, authenticated, confidential channel between the endpoint and the posture manager;

4. enabling the endpoint to notify the posture manager about changes to its configuration;

5. enabling the posture manager to request information about the configuration of the endpoint; and

6. storing the posture information in a repository linked to the identifier for the endpoint.

Furthermore, the EPCP aims to support data storage and data sharing capabilities to make the collected posture information available to authorized parties and components in support of other processes (analytic, access control, remediation, reporting, etc.).

3.1. Components

To perform posture assessment, data storage, and data sharing, the EPCP defines several components. Some of these components reside on the target endpoint. Others reside on a posture manager that manages communications with the target endpoint and stores the target endpoint's posture information in a repository.

It should be noted that the primary focus of this document is on the communication between the posture manager and endpoints. While the orchestrator, evaluator, repository, and administrative interface and API will be discussed in the context of the broader EPCP architecture, these components are not strictly defined nor are best practices provided for them at this time. As a result, vendors are free to implement these components and interfaces in a way that makes the most sense for their products.

```
          ********FUTURE WORK**********  Posture Manager             Endpoint
          *       Orchestrator       *  +-----------------+            +---------
--------+
          *       +---------+        *  |                 |            |
        |
          *       |         |<------------->|             |            |
        |
          *       |         | publish/  *  |                 |            |
        |
          *       |         | subscribe *  |                 |            |
        |
          *       |         |           *  | +-------------+ |            | +------
------+ |
          *       +---------+          *  | |             | |            | |
      | |
          ********FUTURE WORK**********  | | Posture     | | report/    | | Postu
re    | |
                                         | | Collection  | | publish    | | Colle
ction | |
  Evaluator        Repository            | | Manager     | |<-----------| | Engin
e     | |
  +------+         +---------+           | |             | |            | |
      | |
  |      |         |         |           | |             | |            | |
      | |
  |      |         |         |           | +-------------+ |            | +------
------+ |
  |      |<-------->|        |<----------->|               | query/     |
        |
  |      | request/ |        | store       |               | subscribe  |
        |
  |      | respond  |        |             |               |----------->|
        |
  |      |          |        |             |               |            |
        |
  +------+          +---------+            +-----------------+          +---------
--------+
     |                                         ^  ^
     |              query                      |  |
     +-----------------------------------------+  |
                                                  |
***************************FUTURE WORK***********|*************
*                                                 |            *
*                          +---------------------------------+ *
*                          | Administrative Interface        | *
*                          | and API                         | *
*                          +---------------------------------+ *
*                                                              *
***************************FUTURE WORK*************************
```

Figure 1: EPCP Components

3.1.1. Endpoint

An endpoint is defined in [RFC6876]. In the EPCP, the endpoint is monitored by the enterprise and is the target of posture assessments. To support these posture assessments, posture information is collected via a posture collection engine.

3.1.1.1. Posture Collection Engine

The posture collection engine is located on the target endpoint and can either receive queries for data from the posture collection manager (see Section 3.2.4) or can push data to the posture collection manager (see Section 3.2.3). The posture collection engine sends collected posture information to the posture manager where it can be sanity checked and stored in the repository. The posture collection engine also contains a capability that sets up exchanges between the target endpoint and posture manager. This capability makes the posture collection engine responsible for performing the client-side portion of encryption handshakes, and for locating authorized posture managers with which to communicate.

3.1.2. Posture Manager

The posture manager is an endpoint that collects, validates, and enriches posture information received about a target endpoint. It also stores the posture information it receives in the repository where it can be evaluated. The posture manager does not evaluate the posture information.

3.1.2.1. Posture Collection Manager

A posture collection manager is a lightweight and extensible component that facilitates the coordination and execution of posture collection requests using collection mechanisms deployed across the enterprise. The posture collection manager may query and retrieve guidance from the repository to guide the collection of posture information from the target endpoint.

The posture collection manager also contains a capability that sets up exchanges between the target endpoint and the posture manager, and manages data sent to and from posture collection engine. It is also responsible for performing the server-side portion of encryption handshakes.

If the posture manager wants to register for continuous collection of endpoint posture changes with the endpoint, then it must do so in a scalable way. Specifically, it will need to create subscriptions with endpoints in a way which allows the posture data to be securely pushed. Effectively this means that the endpoint must be able to establish secure transport connectivity to the posture collection manager as needed, and the collection manager must be able to periodically collect the current state of the endpoint to verify the expected state of that endpoint.

3.1.3. Repository

The repository hosts guidance, endpoint identification information, and posture information reported by target endpoints where it is made available to authorized components and persisted over a period of time set by the administrator. Information stored in the repository will be accessible to authorized parties via a standard administrative interface as well as through a standardized API. The repository may be a standalone component or may be located on the posture manager. Furthermore, an implementation is not restricted to a single repository and may leverage several repositories to provide this functionality.

3.1.4. Evaluator

The evaluator assesses the posture status of a target endpoint by comparing collected posture information against the desired state of the target endpoint specified in guidance. The evaluator queries and retrieves the appropriate guidance from the repository as well as queries and retrieves the posture information required for the assessment from the repository. If the required posture information is not available in the repository, the evaluator may request the posture information from the posture collection manager, which will result in the collection of additional posture information from the target endpoint. This information is subsequently stored in the repository where it is made available to the evaluator and other components. The results of the assessment are stored in the repository where they are available to tools and administrators for follow-up actions, further evaluation, and historical purposes.

3.1.5. Orchestrator

The orchestrator provides a publish/subscribe interface for the repository so that infrastructure endpoints can subscribe to and receive published posture assessment results from the repository regarding endpoint posture changes.

3.1.6. Administrative Interface and API

The administrative interface allows administrators to query the repository and manage the endpoints and software used in the EPCP via the posture manager. Similarly, an API is necessary to allow infrastructure endpoints and software access to the information stored in the repository and to manage the endpoints and software used in the EPCP. The administrative interface and API provide authorized users, infrastructure endpoints, and software with the ability to query the repository for data, send commands to the posture collection managers requesting information from the associated posture collection engines residing on endpoints, and establish and update the policy that resides on the posture manager.

3.2. Transactions

The following sections describe the transactions associated with the components of the EPCP architecture and may be provided in an implementation.

3.2.1. Provisioning

An endpoint is provisioned with one or more attributes that will serve as its unique identifier on the network as well as the components and data models necessary to interact with the posture manager. Examples of such identifiers include MAC addresses, serial numbers, hardware certificates compliant with [IEEE-802-1ar], and the identities of hardware cryptographic modules among others. Once provisioning is complete, the endpoint is deployed on the network. Over time, components and data models may need to be added to the endpoint or updated to support the collection needs of an enterprise.

3.2.2. Discovery and Validation

If necessary, the target endpoint finds and validates the posture manager. The posture collection engine on the target endpoint and posture collection manager on the posture manager complete an encryption handshake, during which endpoint identity information is exchanged.

3.2.3. Event Driven Collection

The posture assessment is initiated when the posture collector engine on the target endpoint notices that relevant posture information on the endpoint has changed. Then, the posture collection engine initiates a posture assessment information exchange with the posture collection manager.

3.2.4. Querying the Endpoint

The posture assessment is initiated by the posture collection manager. This can occur because:

1. policy states that a previous assessment has aged out or become invalid, or

2. the posture collection manager is alerted by a sensor or an administrator (via the posture manager's administrative interface) that an assessment must be completed.

3.2.5. Data Storage

Once posture information is received by the posture manager, it is forwarded to the repository. The repository could be co-located with the posture manager, or there could be direct or brokered communication between the posture manager and the repository. The posture information is stored in the repository along with past posture information collected about the target endpoint.

3.2.6. Data Sharing

Because the target endpoint posture information was sent in standards-based data models over secure, standardized protocols, and then stored in a centralized repository linked to unique endpoint identifiers, authorized parties are able to access the posture information. Such authorized parties may include, but are not limited to, administrators or endpoint owners (via the posture manager's administrative interface), evaluators that access the repository directly, and orchestrators that rely on publish/subscribe communications with the repository.

4. IETF NEA EPCP Implementation for Traditional Endpoints

When EPCP is used, posture collectors running on the target endpoint gather posture information as changes occur on the endpoint. The data is aggregated by the posture broker client and forwarded to a posture manager, over a secure channel, via the posture transport client. Once received by the posture transport server on the posture manager, the posture information is directed by the posture broker server to the appropriate posture validators where it can be processed and stored in a repository. There the posture information can be used by other tools to carry out assessment tasks. Posture collectors can also be queried by posture validators to refresh posture information about the target endpoint or to ask a specific question about posture information. This is shown in Figure 2.

```
Posture                   Posture
Collection                Collection
Manager                   Engine
+----------------+        +----------------+
|                |        |                |
| +------------+ | PA-TNC | +------------+ |
| | Posture    | |--------| | Posture    | |
| | Validator  | |        | | Collector  | |
| +------------+ |        | +------------+ |
|       |        |        |       |        |
|       | IF-IMV |        |       | IF-IMC |
|       |        |        |       |        |
| +------------+ | PB-TNC | +------------+ |
| | PB Server  | |--------| | PB Client  | |
| +------------+ |        | +------------+ |
|       |        |        |       |        |
|       |        |        |       |        |
|       |        |        |       |        |
| +------------+ |        | +------------+ |
| | PT Server  | |<------>| | PT Client  | |
| +------------+ | PT-TLS | +------------+ |
|                |        |                |
+----------------+        +----------------+
```

Figure 2: NEA Components

These requirements are written with a view to performing a posture assessment on an endpoint; as the EPCP grows and evolves, these requirements will be expanded to address issues that arise. Note that these requirements refer to defined components of the NEA architecture [RFC5209]. As with the NEA architecture, vendors have discretion as to how these NEA components map to separate pieces of software or endpoints.

Furthermore, it should be noted that the posture broker client and posture transport client components of the posture collection engine and the posture broker server and posture transport server components of the posture collection manager would likely need to be implemented by a single vendor because there are no standardized interfaces between the respective components and would not be interoperable.

Examples of the EPCP as implemented using the components from the NEA architecture are provided in Appendix C.

4.1. Endpoint Provisioning

An endpoint is provisioned with a machine certificate that will serve as its unique identifier on the network as well as the components necessary to interact with the posture manager. This includes a posture collection engine to manage requests from the posture manager and the posture collectors necessary to collect the posture information of importance to the enterprise. The endpoint is deployed on the network.

The target endpoint SHOULD authenticate to the posture manager using a machine certificate during the establishment of the outer tunnel achieved with the posture transport protocol defined in [RFC6876]. [IF-IMV] specifies how to pull an endpoint identifier out of a machine certificate. An endpoint identifier SHOULD be created in conformance with [IF-IMV] from a machine certificate sent via [RFC6876].

In the future, the identity could be a hardware certificate compliant with [IEEE-802-1ar]; ideally, this identifier SHOULD be associated with the identity of a hardware cryptographic module, in accordance with [IEEE-802-1ar], if present on the endpoint. The enterprise SHOULD stand up a certificate root authority; install its root certificate on endpoints and on the posture manager; and provision the endpoints and the posture manager with machine certificates. The target endpoint MAY authenticate to the posture manager using a combination of the machine account and password; however, this is less secure and not recommended.

4.2. Endpoint

The endpoint MUST conform to [RFC5793], which levies several requirements against the endpoint. An endpoint that complies with these requirements will be able to:

1. attempt to initiate a session with the posture manager if the posture makes a request to send an update to posture manager;

2. notify the posture collector if no PT-TLS session with the posture manager can be created;

3. notify the posture collector when a PT-TLS session is established; and

4. receive information from the posture collectors, forward this information to the posture manager via the posture collection engine.

4.2.1. Posture Collector

Any posture collector used in an EPCP solution MUST be conformant with the TCG TNC Integrity Measurement Collector interface [IF-IMC].

4.2.2. Posture Broker Client

The posture broker client MUST conform to [IF-IMC] to enable communications between the posture broker client and the posture collectors on the endpoint.

4.2.3. Posture Transport Client

The posture transport client MUST implement PT-TLS.

The posture transport client MUST support the use of machine certificates for TLS at each endpoint consistent with the requirements stipulated in [RFC6876] and [Server-Discovery].

The posture transport client MUST be able to locate an authorized posture manager, and switch to a new posture manager when required by the network, in conformance with [Server-Discovery].

4.3. Posture Manager

The posture manager MUST conform to all requirements in the [RFC5793].

4.3.1. Posture Validator

Any posture validator used in an EPCP solution MUST be conformant with the TCG TNC Integrity Measurement Verifier interface [IF-IMV].

4.3.2. Posture Broker Server

The posture broker server MUST conform to [IF-IMV]. Conformance to [IF-IMV] enables the posture broker server to obtain endpoint identity information from the posture transport server, and pass this information to any posture validators on the posture manager.

4.3.3. Posture Transport Server

The posture transport server MUST implement PT-TLS.

The posture transport server MUST support the use of machine certificates for TLS at each endpoint consistent with the requirements stipulated in [RFC6876] and [Server-Discovery].

4.4. Repository

EPCP requires a simple administrative interface for the repository. Posture validators on the posture manager receive the target endpoint posture information via PA-TNC [RFC5792] messages sent from corresponding posture collectors on the target endpoint. The posture validators store this information in the repository linked to the identity of the target endpoint where the posture collectors are located.

4.5. IETF SACM SWAM Extension to the IETF NEA EPCP Implementation

This section defines the requirements associated with the software asset management extension [RFC8412] to the IETF NEA EPCP implementation.

4.5.1. Endpoint Pre-Provisioning

This section defines the requirements associated with implementing SWIMA.

The following requirements assume that the platform or OS vendor supports the use of SWID tags and has identified a standard directory location for the SWID tags to be located as specified by [SWID].

4.5.2. SWID Tags

The primary content for the EPCP is the information conveyed in the elements of a SWID tag.

The endpoint MUST have SWID tags stored in a directory specified in [SWID]. The tags SHOULD be provided by the software vendor; they MAY also be generated by:

- the software installer; or

- third-party software that creates tags based on the applications it sees installed on the endpoint.

The elements in the SWID tag MUST be populated as specified in [SWID]. These tags, and the directory in which they are stored, MUST be updated as software is added, removed, or updated.

4.5.3. SWID Posture Collectors and Posture Validators

4.5.3.1. The SWID Posture Collector

For the EPCP, the SWID posture collector MUST be conformant with [RFC8412], which includes requirements for:

1. Collecting SWID tags from the SWID directory;

2. Monitoring the SWID directory for changes;

3. Initiating a session with the posture manager to report changes to the directory;

4. Maintaining a list of changes to the SWID directory when updates take place and no PT-TLS connection can be created with the posture manager;

5. Responding to a request for SWID tags from the SWID Posture Validator on the posture manager; and

6. Responding to a query from the SWID posture validator as to whether all updates have been sent.

The SWID posture collector is not responsible for detecting that the SWID directory was not updated when an application was either installed or uninstalled.

4.5.3.2. The SWID Posture Validator

Conformance to [RFC8412] enables the SWID posture validator to:

1. Send messages to the SWID posture collector (at the behest of the administrator at the posture manager console) requesting updates for SWID tags located on endpoint;

2. Ask the SWID posture collector whether all updates to the SWID directory located at the posture manager have been sent; and

3. Perform any validation and processing on the collected SWID posture information prior to storage.

In addition to these requirements, a SWID posture validator used in conformance with this profile MUST be capable of passing this SWID posture information as well as the associated endpoint identity to the repository for storage.

4.5.4. Repository

The administrative interface SHOULD enable an administrator to:

1. Query which endpoints have reported SWID tags for a particular application

2. Query which SWID tags are installed on an endpoint; and

3. Query tags based on characteristics, such as vendor, publisher, etc.

5. IETF NETCONF EPCP Implementation for Network Device Endpoints

When EPCP is used, a NETCONF client that implements the posture collection manager sends a query to target network device endpoint requesting posture information over a secure channel. Once the NETCONF server on the endpoint receives the request, it queries one or more datastores for the posture information. The NETCONF server then reports the information back to the NETCONF client where it can be stored in a repository for use by other tools. This is shown in Figure 3.

```
Posture                   Posture
Collection                Collection
Manager                   Engine
+---------------+         +---------------+
|               |         |               |
|               |         | +-----------+ |
|               |         | | Data      | |
|               |         | | Store(s)  | |
|               |         | +-----------+ |
|               |         |       |       |
|               |         |       |       |
| +-----------+ |         | +-----------+ |
| | NETCONF   | |         | | NETCONF   | |
| | Client    | |<------->| | Server    | |
| +-----------+ | NETCONF | +-----------+ |
|               |         |               |
+---------------+         +---------------+
```

Figure 3: NETCONF Components

These requirements are written with a view to performing a posture assessment on network device endpoints (routers, switches, etc.); as the EPCP grows and evolves, these requirements will be expanded to address issues that arise.

Note that these requirements refer to defined components of the NETCONF architecture and map back to EPCP. As with the NETCONF architecture, vendors have discretion as to how these NETCONF components map to separate pieces of software or endpoints.

5.1. Endpoint Provisioning

For the posture manager to be able to query the datastores on the endpoint, the endpoint MUST be configured to grant the posture manager access to its datastores as described in [RFC6241]. The posture manager is identified by its NETCONF username. The endpoint is deployed on the network.

5.2. Posture Manager Provisioning

For the posture manager to be able to query the datastores on the endpoint, the posture manager MUST be provisioned with a NETCONF username that will be used to authenticate the posture manager to the endpoint as described in [RFC6241]. The username generated will be determined by the selected transport protocol. The posture manager is deployed on the network.

5.3. Endpoint

An endpoint MUST conform to the requirements outlined for servers in the NETCONF protocol as defined in [RFC6241]. This requires the implementation of NETCONF over SSH [RFC6242]. An endpoint MAY support the NETCONF protocol over other transports such as TLS [RFC7589] as well as the RESTCONF protocol as defined in [RFC8040].

5.3.1. Datastore

A NETCONF datastore on an endpoint MUST support the operations outlined in [RFC6241], but, the actual implementation of the datastore is left to the endpoint vendor.

Datastores MUST support the YANG data modeling language [RFC7950] for expressing endpoint posture information in a structured format. In addition, datastores MAY support other data models such as XML (via YIN) for representing posture information.

Datastores MUST support the compliance posture information specified in [RFC7317]. Datastores MAY support other models standardized or proprietary as deemed appropriate by the endpoint vendor.

5.4. Posture Manager

A posture manager MUST conform to the requirements specified for clients in the NETCONF protocol as defined in [RFC6241]. This requires the implementation of NETCONF over SSH [RFC6242]. A posture manager MAY also support the NETCONF protocol over other transports such as TLS [RFC7589]. In addition, a posture manager MAY support the RESTCONF protocol as defined in [RFC8040].

While ad-hoc fetch/polling via NETCONF and RESTCONF is useful for assessing endpoint compliance, such solutions by themselves are not able to detect changes as they occur on the endpoint. As a result, a future revision of this document will support [I-D.ietf-netconf-yang-push] to receive updates on YANG-modeled posture information. Similarly, because not all posture information is modeled in YANG, a future revision of this document will reference [I-D.ietf-netconf-subscribed-notifications] once it is a standard to support continuous streams of unstructured data from the endpoint to the posture manager.

5.5. Repository

EPCP requires a simple administrative interface for the repository. The posture collection manager on the posture manager receives the target endpoint posture information via NETCONF [RFC6241] messages sent from posture collection engine on the target endpoint. The posture collection manager stores this information in the repository linked to the identity of the target endpoint from which it was collected.

6. Future Work

This section captures ideas for future work related to EPCP that might be of interest to the IETF SACM WG. These ideas are listed in no particular order.

o The [I-D.ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] which have been submitted to IESG for publication could be leveraged for an HTTP-based subscription for EPCP. Specifically, it could be used for the posture collection manager to continuously receive posture changes as they happen from the posture collection engine. At this point, it seems like [I-D.ietf-netconf-restconf-notif] would be a good match to these requirements. However further investigation into the applicability of supporting a RESTCONF server capability on to handle subscription requests needs to be made. Specific questions which should be examined include:

  * Number of endpoints which can be continuously tracked by a single posture collection manager. Scalability questions to be considered include elements from the number of transport connects maintained to the volume of volume and churn of posture evidence which will be continuously pushed to the posture collection manager manager.

  * Ability of the posture collection manager to establish and maintain a continuous state of endpoint posture during failures. This includes failures/reboots on either side of the interface.

  * Ability to support for the full set of functions described for NETCONF within Section 5.

- Add support endpoint types beyond workstations, servers, and network infrastructure devices.

- Examine the integration of [I-D.ietf-mile-xmpp-grid].

- Define a standard interface and API for interacting with the repository. Requirements to consider include: creating a secure channel between a publisher and the repository, creating a secure channel between a subscriber and the repository, and the types of interactions that must be supported between publishers and subscribers to a repository.

- Define a standard interface for communications between the posture broker client and posture transport client(s) as well as the posture broker server and posture transport server(s).

- Retention of posture information on the target endpoint.

- Define an orchestrator component as well as publish/subscribe interface for it.

- Define an evaluator component as well as an interface for it.

- Reassess the use of MAC addresses, including market research to determine if MAC addresses continue to be a widely implemented device identifier among network tools.

7. Acknowledgements

The authors wish to thank all of those in the TCG TNC work group who contributed to development of the TNC ECP specification upon which this document is based.

Member	Organization
Padma Krishnaswamy	Battelle Memorial Institute
Eric Fleischman	Boeing
Richard Hill	Boeing
Steven Venema	Boeing
Nancy Cam-Winget	Cisco Systems
Scott Pope	Cisco Systems
Max Pritikin	Cisco Systems
Allan Thompson	Cisco Systems
Nicolai Kuntze	Fraunhofer Institute for Secure Information Technology (SIT)
Ira McDonald	High North
Dr. Andreas Steffen	HSR University of Applied Sciences Rapperswil
Josef von Helden	Hochschule Hannover
James Tan	Infoblox
Steve Hanna (TNC-WG Co-Chair)	Juniper Networks
Cliff Kahn	Juniper Networks
Lisa Lorenzin	Juniper Networks
Atul Shah (TNC-WG Co-Chair)	Microsoft
Jon Baker	MITRE
Charles Schmidt	MITRE
Rainer Enders	NCP Engineering
Dick Wilkins	Phoenix Technologies

David Waltermire	NIST
Mike Boyle	U.S. Government
Emily Doll	U.S. Government
Jessica Fitzgerald-McKay	U.S. Government
Mary Lessels	U.S. Government
Chris Salter	U.S. Government

Table 1: Members of the TNC Work Group that Contributed to the Document

Special thanks also to Dan Ehrlich, Kathleen Moriarty, David Oliva and Eric Voit for their thoughtful comments and edits.

8. IANA Considerations

This document does not define any new IANA registries. However, this document does reference other documents that do define IANA registries. As a result, the IANA Considerations section of the referenced documents should be consulted.

9. Security Considerations

This Security Considerations section includes an analysis of the attacks that may be mounted against systems that implement the EPCP (Section 9.1) and the countermeasures that may be used to prevent or mitigate these attacks (Section 9.2). Overall, a substantial reduction in cyber risk can be achieved.

9.1. Threat Model

This section lists the attacks that can be mounted on a NEA implementation of an EPCP environment. The following section (Section 9.2) describes countermeasures.

Because the EPCP describes a specific use case for NEA components, many security considerations for these components are addressed in more detail in the technical specifications: [RFC8412], [IF-IMC], [RFC5793], [Server-Discovery], [RFC6876], [IF-IMV].

9.1.1. Endpoint Attacks

While the EPCP provides substantial improvements in endpoint security, endpoints can still be compromised. For this reason, all parties must regard data coming from endpoints as potentially unreliable or even malicious. An analogy can be drawn with human testimony in an investigation or trial. Human testimony is essential but must be regarded with suspicion.

- Compromise of endpoint: A compromised endpoint may report false information to confuse or even provide maliciously crafted information with a goal of infecting others.

- Putting bad information in SWID directory: Even if an endpoint is not completely compromised, some of the software running on it may be unreliable or even malicious. This software, potentially including the SWID generation or discovery tool, or malicious software pretending to be a SWID generation or discovery tool, can place incorrect or maliciously crafted information into the SWID directory. Endpoint users may even place such information in the directory, whether motivated by curiosity or confusion or a desire to bypass restrictions on their use of the endpoint.

- Identity spoofing (impersonation): A compromised endpoint may attempt to impersonate another endpoint to gain its privileges or to besmirch the reputation of that other endpoint. This is of particular concern when using MAC addresses to identify endpoints, which, while widely used in endpoint behavior monitoring and threat assessment tools, are easy to spoof.

9.1.2. Network Attacks

Generally, the network cannot be trusted. A variety of attacks can be mounted using the network, including:

- Eavesdropping, modification, injection, replay, deletion;

- Traffic analysis; and

- Denial of service and blocking traffic.

9.1.3. Posture Manager Attacks

The posture manager is a critical security element and therefore merits considerable scrutiny. A variety of attacks can be leveraged against the Posture Manager.

- Compromised trusted manager: A compromised posture manager or a malicious party that is able to impersonate a posture manager can incorrectly grant or deny access to endpoints, place incorrect information into the repository, or send malicious messages to endpoints.

- Misconfiguration of posture manager: Accidental or purposeful misconfiguration of a trusted posture manager can cause effects that are similar to those listed for compromised trusted posture manager.

- Malicious untrusted posture manager: An untrusted posture manager cannot mount any significant attacks because all properly implemented endpoints will refuse to engage in any meaningful dialog with such a posture manager.

9.1.4. Repository Attacks

The repository is also an important security element and therefore merits careful scrutiny.

- Putting bad information into trusted repository: An authorized repository client such as a server may be able to put incorrect information into a trusted repository or delete or modify historical information, causing incorrect decisions about endpoint security. Placing maliciously crafted data in the repository could even lead to compromise of repository clients, if they fail to carefully check such data.

- Compromised trusted repository: A compromised trusted repository or a malicious untrusted repository that is able to impersonate a trusted repository can lead to effects similar to those listed for "Putting bad information into trusted repository". Further, a compromised trusted repository can report different results to different repository clients or deny access to the repository for selected repository clients.

- Misconfiguration of trusted repository: Accidental or purposeful misconfiguration of a trusted repository can deny access to the repository or result in loss of historical data.

- Malicious untrusted repository: An untrusted repository cannot mount any significant attacks because all properly implemented repository clients will refuse to engage in any meaningful dialog with such a repository.

9.2. Countermeasures

This section lists the countermeasures that can be used in a NEA implementation of an EPCP environment.

9.2.1. Countermeasures for Endpoint Attacks

This profile is in and of itself a countermeasure for a compromised endpoint. A primary defense for an endpoint is to run up to date software configured to be run as safely as possible.

Ensuring that anti-virus signatures are up to date and that a firewall is configured are also protections for an endpoint that are supported by the current NEA specifications.

For secure device identification and to correlate device identifiers if the MAC address is randomized, MAC addresses should be collected along with other, more secure endpoint identifiers. Endpoints that have hardware cryptographic modules that are provisioned by the enterprise, in accordance with [IEEE-802-1ar], can protect the private keys used for authentication and help prevent adversaries from stealing credentials that can be used for impersonation. Future versions of the EPCP may want to discuss in greater detail how to use a hardware cryptographic module, in accordance with [IEEE-802-1ar], to protect credentials and to protect the integrity of the code that executes during the bootstrap process by hashing or recording indicators of compromise.

9.2.2. Countermeasures for Network Attacks

To address network attacks, [RFC6876] includes required encryption, authentication, integrity protection, and replay protection. [Server-Discovery] also includes authorization checks to ensure that only authorized servers are trusted by endpoints. Any unspecified or not yet specified network protocols employed in the EPCP (e.g. the protocol used to interface with the repository) should include similar protections.

These protections reduce the scope of the network threat to traffic analysis and denial of service. Countermeasures for traffic analysis (e.g. masking) are usually impractical but may be employed. Countermeasures for denial of service (e.g. detecting and blocking particular sources) SHOULD be used when appropriate to detect and block denial of service attacks. These are routine practices in network security.

9.2.3. Countermeasures for Posture Manager Attacks

Because of the serious consequences of posture manager compromise, posture managers SHOULD be especially well hardened against attack and minimized to reduce their attack surface. They SHOULD be monitored using the NEA protocols to ensure the integrity of the behavior and analysis data stored on the posture manager and SHOULD utilize an [IEEE-802-1ar]-compliant hardware cryptographic module for identity and/or integrity measurements of the posture manager. They should be well managed to minimize vulnerabilities in the underlying platform and in systems upon which the posture manager depends. Network security measures such as firewalls or intrusion detection systems may be used to monitor and limit traffic to and from the posture manager. Personnel with administrative access to the posture manager should be carefully screened and monitored to detect problems as soon as possible. Posture manager administrators should not use password-based authentication but should instead use non-reusable credentials and multi-factor authentication (where available). Physical security measures should be employed to prevent physical attacks on posture managers.

To ease detection of posture manager compromise, should it occur, posture manager behavior should be monitored to detect unusual behavior (such as a server reboot, unusual traffic patterns, or other odd behavior). Endpoints should log and/or notify users and/or administrators when peculiar posture manager behavior is detected. To aid forensic investigation, permanent read-only audit logs of security-relevant information pertaining to posture manager (especially administrative actions) should be maintained. If posture manager compromise is detected, the posture manager's certificate should be revoked and careful analysis should be performed of the source and impact of this compromise. Any reusable credentials that may have been compromised should be reissued.

Endpoints can reduce the threat of server compromise by minimizing the number of trusted posture managers, using the mechanisms described in [Server-Discovery].

9.2.4. Countermeasures for Repository Attacks

If the host for the repository is located on its own endpoint, it should be protected with the same measures taken to protect the posture manager. In this circumstance, all messages between the posture manager and repository should be protected with a mature security protocol such as TLS or IPsec.

The repository can aid in the detection of compromised endpoints if an adversary cannot tamper with its contents. For instance, if an endpoint reports that it does not have an application with a known vulnerability installed, an administrator can check whether the endpoint might be lying by querying the repository for the history of what applications were installed on the endpoint.

To help prevent tampering with the information in the repository:

1. Only authorized parties should have privilege to run code on the endpoint and to change the repository.

2. If a separate endpoint hosts the repository, then the functionality of that endpoint should be limited to hosting the repository. The firewall on the repository should only allow access to the posture manager and to any endpoint authorized for administration.

3. The repository should ideally use "write once" media to archive the history of what was placed in the repository, to include a snapshot of the current status of applications on endpoints.

10. Privacy Considerations

The EPCP specifically addresses the collection of posture data from enterprise endpoints by an enterprise network. As such, privacy is not going to often arise as a concern for those deploying this solution.

A possible exception may be the concerns a user may have when attempting to connect a personal endpoint (such as a phone or mobile endpoint) to an enterprise network. The user may not want to share certain details, such as an endpoint identifier or SWID tags, with the enterprise. The user can configure their NEA client to reject requests for this information; however, it is possible that the enterprise policy will not allow the user’s endpoint to connect to the network without providing the requested data.

An enterprise network should limit access to endpoint posture and identification information to authorized users.

11. References

11.1. Informative References

[CIS]   http://www.cisecurity.org/controls/, "CIS Critical Security Controls".

[DSD] http://www.dsd.gov.au/publications/csocprotect/top_4_mitigations.htm, "Top 4 Mitigation Strategies to Protect Your ICT System", November 2012.

[ECP] Trusted Computing Group, "TCG Trusted Network Connect Endpoint Compliance Profile, Version 1.10", December 2014.

[IEEE-802-1ar]
Institute of Electrical and Electronics Engineers, "IEEE 802.1ar", December 2009.

[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>.

[TNC] Trusted Computing Group, "TCG Trusted Network Connect TNC Architecture for Interoperability, Version 1.5", February 2012.

11.2. Normative References

[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-04 (work in progress), October 2017.

[I-D.ietf-netconf-restconf-notif]
Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Dynamic subscription to YANG Events and Datastores over RESTCONF", draft-ietf-netconf-restconf-notif-15 (work in progress), June 2019.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-13 (work in progress), June 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-12 (work in progress), December 2017.

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

[IF-IMC] Trusted Computing Group, "TCG Trusted Network Connect TNC IF-IMC, Version 1.3", February 2013.

[IF-IMV] Trusted Computing Group, "TCG Trusted Network Connect TNC IF-IMV, Version 1.4", December 2014.

[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>.

[RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5792, DOI 10.17487/RFC5792, March 2010, <https://www.rfc-editor.org/info/rfc5792>.

[RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5793, DOI 10.17487/RFC5793, March 2010, <https://www.rfc-editor.org/info/rfc5793>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

[Server-Discovery] Trusted Computing Group, "DRAFT: TCG Trusted Network Connect PDP Discovery and Validation, Version 1.0", October 2015.

[SWID] "Information technology--Software asset management--Part 2: Software identification tag", ISO/IEC 9899:1999, 2009.

Appendix A. Rationale for an EPCP Solution

A.1. Preventative Posture Assessments

The value of continuous endpoint posture assessment is well established. Security experts have identified asset management and vulnerability remediation as a critical step for preventing intrusions. Application whitelisting, patching applications and operating systems, and using the latest versions of applications top the Defense Signals Directorate’s "Top 4 Mitigations to Protect Your ICT System". [DSD] "Inventory of Authorized and Unauthorized Endpoints", "Inventory of Authorized and Unauthorized Software", and "Continuous Vulnerability Assessment and Remediation" are Controls 1, 2, and 3, respectively, of the CIS Controls [CIS]. While there are commercially available solutions that attempt to address these security controls, these solutions do not run on all types of endpoints; consistently interoperate with other tools that could make use of the data collected; collect posture information from all types of endpoints in a consistent, standardized schema; or require vetted, standardized protocols that have been evaluated by the international community for cryptographic soundness.

As is true of most solutions offered today, the solution found in the EPCP does not attempt to solve the lying endpoint problem, or detect infected endpoints; rather, it focuses on ensuring that healthy endpoints remain healthy by keeping software up-to-date and patched.

A.2. All Network-Connected Endpoints are Endpoints

As defined by [I-D.ietf-sacm-terminology], an endpoint is any physical or virtual computing endpoint that can be connected to a network. Posture assessment against policy is equally, if not more, important for continuously connected endpoints, such as enterprise workstations and infrastructure endpoints, as it is for sporadically connected endpoints. Continuously connected endpoints are just as likely to fall out of compliance with policy, and a standardized posture assessment method is necessary to ensure they can be properly handled.

A.3. All Endpoints on the Network Must be Uniquely Identified

Many administrators struggle to identify what endpoints are connected to the network at any given time. By requiring a standardized method of endpoint identity, the EPCP will enable administrators to answer the basic question, "What is on my network?" In [I-D.ietf-sacm-terminology], SACM defines this set of endpoints on the network as the SACM domain. Unique endpoint identification also enables the comparison of current and past endpoint posture assessments, by allowing administrators to correlate assessments from the same endpoint. This makes it easier to flag suspicious changes in endpoint posture for manual or automatic review, and helps to swiftly identify malicious changes to endpoint applications.

A.4. Standardized Data Models

Meeting EPCP best practices requires the use of standardized data models for the exchange of posture information. This helps to ensure that the posture information sent from endpoints to the repository can be easily stored, due to their known format, and shared with authorized endpoints and users.

Posture information must be sent over standardized protocols to ensure the confidentiality and authenticity of this data while in transit. Implementations of the EPCP include [RFC6876] and [RFC6241] for communication between the target endpoint and the posture manager. These protocols allow networks that implement this solution to collect large amounts of posture information from an endpoint to make decisions about that endpoint's compliance with some policy. The EPCP offers a solution for all endpoints already connected to the network. Periodic assessments and automated reporting of changes to endpoint posture allow for instantaneous identification of connected endpoints that are no longer compliant to some policy.

A.5. Posture Information Must Be Stored

Posture information must be stored by the repository and must be exposed to an interface at the posture manager. Standard data models enable standard queries from an interface exposed to an administrator at the posture manager console. A repository must retain any current posture information retrieved from the target endpoint and store it indexed by the unique identifier for the endpoint. Any posture collection manager specified by this profile must be able to ascertain from its corresponding posture collection engine whether the posture information is up to date. An interface on the posture manager must support a request to obtain up-to-date information when an endpoint is connected. This interface must also support the ability to make a standard set of queries about the posture information stored by the repository. In the future, some forms of posture information might be retained at the endpoint. The interface on the posture manager must accommodate the ability to make a request to the corresponding posture collection engine about the posture of the target endpoint. Standard data models and protocols also enable the security of posture assessment results. By storing these results indexed under the endpoint's unique identification, secure storage itself enables endpoint posture information correlation, and ensures that the enterprise's repositories always offer the freshest, most up-to-date view of the enterprise's endpoint posture information possible.

A.6. Posture Information Can Be Shared

By exposing posture information using a standard interface and API, other security and operational components have a high level of insight into the enterprise's endpoints and the software installed on them. This will support innovation in the areas of asset management, vulnerability scanning, and administrative interfaces, as any authorized infrastructure endpoint can interact with the posture information.

A.7. Enterprise Asset Posture Information Belongs to the Enterprise

Owners and administrators must have complete control of posture information, policy, and endpoint mitigation. Standardized data models, protocols and interfaces help to ensure that this posture information is not locked in proprietary databases, but is made available to its owners. This enables administrators to develop as nuanced a policy as necessary to keep their networks secure. Of course, there may be exceptions to this such as the case with privacy-related information (e.g., personally identifiable information).

Appendix B. EPCP Supported Use Cases and Non-Supported Use Cases

B.1. Supported Use Cases

The following sections describe the different use cases supported by the EPCP.

B.1.1. Hardware Asset Management

Using the administrative interface on the posture manager, an authorized user can learn:

- what endpoints are connected to the network at any given time; and

- what SWID tags were reported for the endpoints.

The ability to answer these questions offers a standards-based approach to asset management, which is a vital part of enterprise processes such as compliance report generation for the Federal Information Security Modernization Act (FISMA), Payment Card Industry Data Security Standard (PCI DSS), Health Insurance Portability and Accountability Act (HIPAA), etc.

B.1.2. Software Asset Management

The administrative interface on the posture manager provides the ability for authorized users and infrastructure to know which software is installed on which endpoints on the enterprise's network. This allows the enterprise to answer questions about what software is installed to determine if it is licensed or prohibited. This information can also drive other use cases such as:

- vulnerability management: knowing what software is installed supports the ability to determine which endpoints contain vulnerable software and need to be patched.

- configuration management: knowing which security controls need to be applied to harden installed software and better protect endpoints.

B.1.3. Vulnerability Management

The administrative interface also provides the ability for authorized users or infrastructure to locate endpoints running software for which vulnerabilities have been announced. Because of

1. the unique IDs assigned to each endpoint; and

2. the rich application data provided in the endpoints' posture information,

the repository can be queried to find all endpoints running a vulnerable application. Endpoints suspected of being vulnerable can be addressed by the administrator or flagged for further scrutiny.

B.1.4. Threat Detection and Analysis

The repository's standardized API allows authorized infrastructure endpoints and software to search endpoint posture assessment information for evidence that an endpoint's software inventory has changed, and can make endpoint software inventory data available to other endpoints. This automates security data sharing in a way that expedites the correlation of relevant network data, allowing administrators and infrastructure endpoints to identify odd endpoint behavior and configuration using secure, standards-based data models and protocols.

B.2. Non-Supported Use Cases

Several use cases, including but not limited to these, are not covered by the EPCP:

- Gathering non-standardized types of posture information: The EPCP does not prevent administrators from collecting posture information in proprietary formats from the endpoint; however it does not set requirements for doing so.

- Solving the lying endpoint problem: The EPCP does not address the lying endpoint problem; the Profile makes no assertions that it can catch an endpoint that is, either maliciously or accidentally, reporting false posture information to the posture manager. However, other solutions may be able to use the posture information collected using the capabilities described in this profile to catch an endpoint in a lie. For example, a sensor may be able to compare the posture information it has collected on an endpoint's activity on the network to what the endpoint reported to the server and flag discrepancies. However, these capabilities are not described in this profile.

Appendix C. Endpoint Posture Collection Profile Examples

The following subsections provide examples of the EPCP as implemented using components from the NEA architecture.

C.1. Continuous Posture Assessment of an Endpoint

```
               Endpoint                 Posture Manager
               +---------------+        +---------------+
               |               |        |               |
               | +-----------+ |        | +-----------+ |
               | | SWID      | |        | | SWID      | |
               | | Posture   | |        | | Posture   | |
               | | Collector | |        | | Validator | |
               | +-----------+ |        | +-----------+ |
               |      |        |        |      |        |
               |      | IF-IMC |        |      | IF-IMV |
               |      |        |        |      |        |
               | +-----------+ |        | +-----------+ |
               | | PB Client | |        | | PB Server | |
               | +-----------+ |        | +-----------+ |
               |      |        |        |      |        |
               |      |        |        |      |        |
               |      |        |        |      |        |
               | +-----------+ |        | +-----------+ |
               | | PT Client | |<------>| | PT Server | |
               | +-----------+ | PT-TLS | +-----------+ |
               |               |        |               |
               +---------------+        +---------------+
```

Figure 4: Continuous Posture Assessment of an Endpoint

C.1.1. Change on Endpoint Triggers Posture Assessment

A new application is installed on the endpoint, and the SWID directory is updated. This triggers an update from the SWID posture collector to the SWID posture validator. The message is sent down the NEA stack, encapsulated by NEA protocols until it is sent by the posture transport client to the posture transport server. The posture transport server then forwards it up through the stack, where the layers of encapsulation are removed until the SWID Message arrives at the SWID posture validator.

```
              Endpoint                         Posture Manager
              +---------------+                +---------------+
              |               |                |               |
              | +-----------+ |                | +-----------+ |
              | | SWID      | |                | | SWID      | |
              | | Posture   | |                | | Posture   | |
              | | Collector | |                | | Validator | |
              | +-----------+ |                | +-----------+ |
              |      |        | SWID Message   |      |        |
              |      | IF-IMC | for PA-TNC     |      | IF-IMV |
              |      |        |                |      |        |
              | +-----------+ |                | +-----------+ |
              | | PB Client | |                | | PB Server | |
              | +-----------+ |                | +-----------+ |
              |      |        |                |      |        |
              |      |        | PB-TNC {SWID   |      |        |
              |      |        | Message for    |      |        |
              |      |        | PA-TNC}        |      |        |
              | +-----------+ |                | +-----------+ |
              | | PT Client | |<-------------->| | PT Server | |
              | +-----------+ | PT-TLS {PB-TNC | +-----------+ |
              |               | {SWID Message  |               |
              +---------------+ for PA-TNC}}   +---------------+
```

Figure 5: Compliance Protocol Encapsulation

The SWID posture validator stores the new tag information in the repository. If the tag indicates that the endpoint is compliant to the policy, then the process is complete until the next time an update is needed (either because policy states that the endpoint must submit posture assessment results periodically or because an install/uninstall/update on the endpoint triggers a posture assessment).

```
              Endpoint                 Posture Manager
              +---------------+        +---------------+
              |               |        |               |
              | +-----------+ |        | +-----------+ |
              | | SWID      | |        | | SWID      |-|-+
              | | Posture   | |        | | Posture   | | |
              | | Collector | |        | | Validator | | |
              | +-----------+ |        | +-----------+ | |
              |      |        |        |      |        | |     Repository
              |      | IF-IMC |        |      | IF-IMV | |     +--------+
              |      |        |        |      |        | |     |        |
              | +-----------+ |        | +-----------+ | |     |        |
              | | PB Client | |        | | PB Server | +---->|        |
              | +-----------+ |        | +-----------+ |       |        |
              |      |        |        |      |        |       +--------+
              |      |        |        |      |        |
              |      |        |        |      |        |
              | +-----------+ |        | +-----------+ |
              | | PT Client | |<------>| | PT Server | |
              | +-----------+ | PT-TLS | +-----------+ |
              |               |        |               |
              +---------------+        +---------------+
```

Figure 6: Storing SWIDs in the Repository

If the endpoint has fallen out of compliance with a policy, the posture manager can alert the administrator via the posture manager's administrative interface. The administrator can then take steps to address the problem. If the administrator has already established a policy for automatically addressing this problem, that policy will be followed.

```
                                                      (")
                                                     __|__
                                                   +-->|
         Endpoint                 Posture Manager   |   / \
         +---------------+        +---------------+ |
         |               |        |               | |
         | +-----------+ |        | +-----------+ | |
         | | SWID      | |        | | SWID      |-|-+
         | | Posture   | |        | | Posture   | |
         | | Collector | |        | | Validator | |
         | +-----------+ |        | +-----------+ |
         |       |       |        |       |       |       Repository
         |       | IF-IMC|        |       | IF-IMV|       +---------+
         |       |       |        |       |       |       |         |
         | +-----------+ |        | +-----------+ |       |         |
         | | PB Client | |        | | PB Server | |       |         |
         | +-----------+ |        | +-----------+ |       |         |
         |       |       |        |       |       |       +---------+
         |       |       |        |       |       |
         |       |       |        |       |       |
         | +-----------+ |        | +-----------+ |
         | | PT Client | |<------>| | PT Server | |
         | +-----------+ | PT-TLS | +-----------+ |
         |               |        |               |
         +---------------+        +---------------+
```

Figure 7: Server Alerts Network Admin

C.2. Administrator Searches for Vulnerable Endpoints

An announcement is made that a particular version of a piece of software has a vulnerability. The administrator uses the administrative interface on the server to search the repository for endpoints that reported the SWID tag for the vulnerable software.

```
                                                           (")
                                                          __|__
                                                        +-->|
             Endpoint                 Posture Manager   |  / \
             +---------------+        +---------------+ |
             |               |        |               | |
             | +-----------+ |        | +-----------+ | |
             | | SWID      | |        | | SWID      |-|-+
             | | Posture   | |        | | Posture   | |
             | | Collector | |        | | Validator | |
             | +-----------+ |        | +-----------+ |
             |      |        |        |      |        |       Repository
             |      | IF-IMC |        |      | IF-IMV |       +---------+
             |      |        |        |      |        |       |         |
             | +-----------+ |        | +-----------+ |       |         |
             | | PB Client | |        | | PB Server | |------>|         |
             | +-----------+ |        | +-----------+ |       |         |
             |      |        |        |      |        |       +---------+
             |      |        |        |      |        |
             |      |        |        |      |        |
             | +-----------+ |        | +-----------+ |
             | | PT Client | |<------>| | PT Server | |
             | +-----------+ | PT-TLS | +-----------+ |
             |               |        |               |
             +---------------+        +---------------+
```

Figure 8: Admin Searches for Vulnerable Endpoints

The repository returns a list of entries in the matching the administrator's search. The administrator can then address the vulnerable endpoints by taking some follow-up action such as removing it from the network, quarantining it, or updating the vulnerable software.

Appendix D. Change Log

D.1. -04 to -05

Updated the diagram so the Evaluator and Repository are "current work".

Clarified how the Posture Collection Engine can push data, respond to queries, and establish secure transport connectivity for fulfilling subscriptions.

Expanded on the future work around leveraging NETCONF, RESTCONF, and YANG Push for network devices.

Documented the need to reassess MAC addresses as a device identifier.

Made various typographical and editorial changes.

D.2. -03 to -04

Addressed various comments from the SACM WG.

Refactored the document to better focus it on the communications between endpoints and the posture manager and the best practices for EPCP implementations.

Made other editorial changes and improved consistency throughout the document.

D.3. -02 to -03

Addressed various comments from the SACM WG.

Added a reference to TCG ECP 1.0.

Removed text in the "SWID Posture Validator" section that states it performs evaluation. This was removed because it contradicts the posture manager not performing any evaluations.

Expanded the "Provisioning" section of the "EPCP Transactions" section to include examples of endpoint identifiers and the need to provision endpoints with components and data models.

Combined text for the capabilities of the Administrative Interface and API.

Removed superfluous and introductory text from the "Security Considerations" section.

Renamed section "Vulnerability Searches" to Vulnerability Management".

Changed I-D category to BCP.

Changed references to the NETMOD architecture to the NETCONF architecture because NETCONF represents the management protocol whereas NETMOD is focused on the definition of data models.

Addressed various editorial suggestions.

D.4. -01 to -02

Addressed various comments from the SACM WG.

Added a section for the collection of posture information from network devices using standards from the NETMOD WG.

Updated EPCP component diagrams so they were not specific to a NEA-based implementation.

Updated EPCP NEA example diagrams to reflect all the components in the NEA architecture.

D.5. -00 to -01

There are no textual changes associated with this revision. This revision simply reflects a resubmission of the document so that it remains in active status.

D.6. -01 to -02

Added references to the Software Inventory Message and Attributes (SWIMA) for PA-TNC I-D.

Replaced references to PC-TNC with IF-IMC.

Removed erroneous hyphens from a couple of section titles.

Made a few minor editorial changes.

D.7. -02 to -00

Draft adopted by IETF SACM WG.

D.8. -00 to -01

Significant edits to up-level the draft to describe SACM collection over multiple different protocols.

Replaced references to SANS with CIS.

Made other minor editorial changes.

Authors' Addresses

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

Email: dhaynes@mitre.org

Jessica Fitzgerald-McKay
Department of Defense
9800 Savage Road
Ft. Meade, Maryland
USA

Email: jmfitz2@nsa.gov

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

Email: llorenzin@pulsesecure.net

SACM C. Schmidt
Internet-Draft D. Haynes
Intended status: Standards Track C. Coffin
Expires: October 8, 2018 The MITRE Corporation
D. Waltermire
National Institute of Standards and Technology
J. Fitzgerald-McKay
United States National Security Agency
April 6, 2018

Software Inventory Message and Attributes (SWIMA) for PA-TNC
draft-ietf-sacm-nea-swima-patnc-05

Abstract

This document extends "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)" (RFC 5792) by providing specific attributes and message exchanges to allow endpoints to report their installed software inventory information to a NEA server, as defined in "Network Endpoint Assessment (NEA): Overview and Requirements" (RFC 5209).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Knowing the list of the software installed on endpoints is useful to understand and maintain the security state of a network. For example, if an enterprise policy requires the presence of certain software and prohibits the presence of other software, reported software installation information can be used to indicate compliance and non-compliance with these requirements. Endpoint software installation inventory lists (hereinafter "software inventories") can further be used to determine an endpoint's exposure to attack based on comparison to vulnerability or threat alerts against identified software's patch level data. These are some of the highly useful management use cases supported by software inventory data.

Software Inventory Message and Attributes (SWIMA) for PA-TNC provides a standardized method for exchanging software inventory data that includes a unique software identifier associated with a specific version of a software product. SWIMA can also convey metadata about software products beyond this identifier. SWIMA enables software identification, installation, and characterization information to be be transported to a central server from any endpoint that supports this specification. Such information can come from multiple sources, including tag files (such as ISO SWID tags [SWID15]), reports from third party inventory tools, output from package managers, and other sources. SWIMA does not standardize how software is detected, instead relying on a set of "data sources" to provide information about installed software. Instead, SWIMA provides a flexible transport capable of conveying this information regardless of how it is expressed.

This specification is designed to only report software that is installed on a target endpoint. In particular, it does not monitor or report information about what software is running on the endpoint. Likewise, it is not intended to report individual files, libraries, installation packages, or similar artifacts. While all of this information has its uses, this information requires different metadata and monitoring methods. As a result, this specification focuses solely on software inventory information, leaving reporting of other classes of endpoint information to other specifications.

Note that while this specification focuses on "software inventory", the mechanisms it describes could also be used to convey information about firmware and operating systems associated with an endpoint. The focus on software throughout this document should not be read as excluding the use of SWIMA for these other purposes.

This specification defines a new set of PA-TNC attributes, which are used to communicate requests for software inventory information and software installation change events. The exchange of these messages allows software inventory information to be sent to a NEA Server, which can make this information available to other applications.

Part of the motivation for the development of SWIMA was to support the IETF’s Security Automation and Continuous Monitoring (SACM) architecture. More details about SWIMA’s role in SACM appear in Section 7. However, SWIMA has no dependencies on any part of SACM and is usable wherever the NEA architecture is employed.

1.1. Network Endpoint Assessment (NEA)

SWIMA defines extensions to the PA-TNC specification, which is part of the Network Endpoint Assessment (NEA) architecture. The NEA specifications define an open solution architecture that enables network operators to collect and utilize information about endpoint configuration and state. This information can be used to enforce policies, monitor endpoint health, and for many other activities. Information about the software present on an endpoint is an important consideration for such activities. The new PA-TNC attributes defined in this document are used to communicate software inventory evidence, collected from a range of possible sources, from the posture collector on the endpoint to the posture validator on a NEA Server using the PA-TNC interface, as shown in Figure 1 below.

```
       +-------------+                           +--------------+
       |  Posture    |   <--------PA-------->    |   Posture    |
       |  Collectors |                           |   Validators |
       |  (1 .. N)   |                           |   (1 .. N)   |
       +-------------+                           +--------------+
             |                                          |
             |                                          |
             |                                          |
       +-------------+                           +--------------+
       |   Posture   |                           |   Posture    |
       |   Broker    |   <--------PB-------->    |   Broker     |
       |   Client    |                           |   Server     |
       +-------------+                           +--------------+
             |                                          |
             |                                          |
       +-------------+                           +--------------+
       |   Posture   |                           |   Posture    |
       |   Transport |   <--------PT-------->    |   Transport  |
       |   Client    |                           |   Server     |
       |   (1 .. N)  |                           |   (1 .. N)   |
       +-------------+                           +--------------+
          NEA CLIENT                                NEA SERVER
```

Figure 1: NEA Reference Model

To better understand this specification, the reader should review the NEA reference architecture as described in the Network Endpoint Assessment (NEA): Overview and Requirements [RFC5209]. The reader should also review the PA-TNC interfaces as defined in RFC 5792 [RFC5792].

This document is based on standards published by the Trusted Computing Group’s Trusted Network Communications (TNC) workgroup. The TNC and NEA architectures are interoperable and many components are equivalent.

1.2. Conventions Used in This Document

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Definitions

This section defines terms with special meaning within this document.

SWIMA-PC - A NEA Posture Collector (PC) that interprets SWIMA Attributes sent by SWIMA-PVs and which conforms to this specification. Note that such a posture collector might also support other PA-TNC exchanges beyond those defined herein.

SWIMA-PV - A NEA Posture Validator (PV) that interprets SWIMA Attributes sent by SWIMA-PCs and which conforms to this specification. Note that such a posture verifier might also support other PA-TNC exchanges beyond those defined herein.

SWIMA Attribute - This is a PA-TNC attribute (as defined in RFC 5792 [RFC5792] extension as defined in this specification.

Endpoint’s Software Inventory Evidence Collection - The set of information regarding the set of software installed on an endpoint. An endpoint’s software inventory evidence collection might include information created by or derived from multiple sources, including but not limited to SWID tag files deposited on the file system during software installation, information generated by software discovery tools, and information dynamically generated by a software or package management system on an endpoint.

Software Inventory Evidence Record - The endpoint’s Software Inventory Evidence Collection is composed of "records". Each record corresponds to one installed instance of a particular software product as reported by some data source. It is possible for a single installed instance to have multiple software inventory evidence records in an endpoint’s Software Inventory Evidence Collection - this can happen if multiple sources all report the same software installation instance.

Software Identifier - A string associated with a specific version of a specific software product. These identifiers are derived from the records used to describe software products. SWIMA does not limit the formats of these records, nor does it enforce that the same format be populated the same way by all data sources. As such, while each software identifier uniquely identifies a specific software product, the same software product might be associated with multiple software identifiers reflecting differences between different data sources and supported record formats.

2. Background

2.1. Supported Use Cases

This section describes the use cases supported by this specification. The primary use of exchanging software inventory information over the PA-TNC interface is to enable a challenger (e.g., NEA Server) to obtain inventory evidence about some system in a way that conforms to NEA procedures and expressed using a standard format. Collected software information can support a range of security activities including determining whether an endpoint is permitted to connect to the enterprise, determining which endpoints contain software that requires patching, and similar activities.

2.1.1. Use Software Inventory as an Access Control Factor

Some enterprises might define security policies that require connected endpoints to have certain pieces of security software installed. By contrast, some security policies might prevent access to resources by endpoints that have certain prohibited pieces of software installed, since such applications might pose a security risk. To support such policies, the NEA Server needs to collect software inventory evidence from a target endpoint that is seeking to initiate or continue connectivity to the enterprise resource.

Based on this specification, the SWIMA-PC can provide a complete or partial inventory to the SWIMA-PV as required to determine policy compliance. The SWIMA-PV can then use this as evidence of compliance or non-compliance to make a policy-based access decision.

2.1.2. Central Stores of Up-to-Date Endpoint Software Inventory Data

Many tools use information about an endpoint's software inventory to monitor and enforce the security of a network. For example, a software patching tool needs to determine if there is out-of-date software installed that needs to be updated. A vulnerability management tool needs to identify endpoints with known vulnerable software installed (patched or otherwise) to gauge an endpoint's relative exposure to attack. A license management tool needs to verify that all installed software within the enterprise is accounted for. A central repository representing an up-to-date understanding of each endpoint's software inventory facilitates these activities. Multiple tools can share such a repository ensuring that software inventory information is collected more frequently and efficiently, leading to a more complete and consistent understanding of installed software state as compared to each tool collecting the same inventory information from endpoints individually.

This specification supports these activities through a number of mechanisms. As noted above, a SWIMA-PC can provide a complete list of software present in an endpoint's Software Inventory Evidence Collection to the SWIMA-PV, which can then pass this information on to a central repository, such as a Configuration Management Database (CMDB) or similar application. In addition, SWIMA-PCs are required to be able to monitor for changes to an endpoint's Software Inventory Evidence Collection in near real-time and immediately push reports of detected changes to the SWIMA-PV. Thus, any central repository fed by a SWIMA-PV receiving inventory information can be updated quickly after a change occurs. Keeping a central repository synchronized with current software inventory information in this way allows tools to make efficient decisions based on up-to-date, consistent information.

2.1.3. PA-TNC Use Cases

SWIMA is intended to operate over the PA-TNC interface and, as such, are intended to meet the use cases set out in the PA-TNC specification.

2.2. Non-supported Use Cases

Some use cases not covered by this specification include:

- Addressing how the endpoint's Software Inventory Evidence Collection is populated. In particular, NEA components are not expected to perform software discovery activities beyond compiling information in an endpoint's Software Inventory Evidence Collection. This collection might come from multiple sources on the endpoint (e.g., information generated dynamically by package management tools or discovery tools, as well as SWID tag files discovered on the file system). While an enterprise might make use of software discovery capabilities to identify installed software, such capabilities are outside the scope of this specification.

- Converting inventory information expressed in a proprietary format into formats used in the attributes described in this specification. Instead, this specification focuses exclusively on defining interfaces for the transportation of software information expecting that reporting tools will converge around some set of standardized formats for this information.

- Mechanisms for a posture validator to request a specific list of software information based on arbitrary software properties. For example, requesting only information about software from a particular vendor is not supported. After the endpoint's Software

Inventory Evidence Collection has been copied to some central location, such as the CMDB, processes there can perform queries based on any criteria present in the collected information, but this specification does not address using such queries to constrain the initial collection of this information from the endpoint.

- Use of properties of certain sources of software information that might facilitate local tests (i.e., on the endpoint) of endpoint state. For example, the optional package_footprint field of an ISO SWID tag can contain a list of files and hash values associated with the software indicated by the tag. Tools on the endpoint can use the values in this field to test for the presence of the indicated files. Successful evaluation of such tests leads to greater assurance that the indicated software is present on the endpoint. Currently, most SWID tag creators do not provide values for tag fields that support local testing. For this reason, the added complexity of supporting endpoint testing using these fields is out of scope for this specification, but may be considered in a future version.

2.3. SWIMA Requirements

Below are the requirements that the SWIMA specification is required to meet in order to successfully play its role in the NEA architecture.

Efficient: The NEA architecture enables delay of network access until the endpoint is determined not to pose a security threat to the network based on its asserted integrity information. To minimize user frustration, SWIMA ought to minimize overhead delays and make PA-TNC communications as rapid and efficient as possible.

Scalable: SWIMA needs to be usable in enterprises that contain tens of thousands of endpoints or more. As such, it needs to allow a security tools to make decisions based on up-to-date information about an endpoint’s software inventory without creating an excessive burden on the enterprise’s network.

Support precise and complete historical reporting: This specification outlines capabilities that support real-time understanding of the state of endpoint in a network in a way that can be used by other tools. One means of facilitating such an outcome is for a CMDB to be able to contain information about all endpoints connected to the enterprise for all points in time between the endpoint’s first connection and the present. In such a scenario, it is necessary that any PC be able to report any changes to its software inventory evidence collection in near real-time while connected and, upon reconnection to the enterprise, be able to update the NEA Server (and through it the CMDB) with regard to the state of its software inventory evidence collection throughout the entire interval when it was not connected.

2.4. Non-SWIMA Requirements

There are certain capabilities that users of the SWIMA specification might require but which are beyond the scope of SWIMA itself and need to be addressed by other standards. This list is not exhaustive.

Confidentiality: The SWIMA specification does not define a mechanism for confidentiality, nor is confidentiality automatically provided by using the PA-TNC interface. In the NEA architecture, confidentiality is generally provided by the underlying transport protocols, such as the PT Binding to TLS [RFC6876] or PT-EAP Posture Transport for Tunneled EAP Methods [RFC7171] - see Section 7 for more information on related standards. The information conveyed by SWIMA is often sensitive in nature for both security (Section 8) and privacy (Section 9) reasons. Those who implement SWIMA need to ensure that appropriate NEA transport mechanisms are employed to meet confidentiality requirements.

2.5. Assumptions

The Posture Broker Client and Posture Broker Server are assumed to provide reliable delivery for PA-TNC messages and attributes sent between the SWIMA-PCs and the SWIMA-PVs. Reliable delivery means that either a message is delivered or the sender is made aware of the delivery failure. In the event that reliable delivery cannot be provided, the Posture Collector or Posture Validator is expected to terminate the connection.

2.6. Non-Assumptions

This specification explicitly does not assume that software inventory information exchanges reflect the software installation state of the endpoint. This specification does not attempt to detect when the endpoint is providing false information, either through malice or error, but instead focuses on correctly and reliably providing the reported Software Inventory Evidence Collection to the NEA Server. Tools that employ the SWIMA standard can include methods to help verify the accuracy of reports, but how those tools do so is beyond the scope of this specification.

Similarly, this specification makes no assumption about the completeness of the Software Inventory Evidence Collection’s coverage of the total set of software installed on the endpoint. It is possible, and even likely, that some installed software is not represented by a record in an endpoints Software Inventory Evidence Collection. Instead, SWIMA ensures that what does get reported is reported consistently and that the software products that are reported can be reliably tracked.

See Section 8 for more on this security consideration.

3. System Requirements

The SWIMA specification facilitates the exchange of software inventory and event information. Specifically, each application supporting SWIMA includes a component known as the SWIMA-PC that receives messages sent with the SWIMA Attributes component type. The SWIMA-PC is also responsible for sending appropriate SWIMA Attributes back to the SWIMA-PV in response. This section outlines what software inventories and events are and the requirements on SWIMA-PCs and SWIMA-PVs in order to support the stated use cases of this specification.

3.1. Data Sources

The records in an endpoint’s software inventory evidence collection come from one or more "sources". A source represents one collection of software inventory information about the endpoint. Examples of sources include, but are not limited to, ISO SWID tags deposited on the filesystem and collected therefrom, information derived from package managers (e.g., RPM or YUM), and the output of software inventory scanning tools.

There is no expectation that any one source of inventory information will have either perfect or complete software inventory information. For this reason, this specification supports the simultaneous use of multiple sources of software inventory information. Each source might have its own "sphere of expertise" and report the software within that sphere. For example, a package manager would have excellent understanding of the software that it managed, but would not necessarily have any information about software installed via other means.

A SWIMA-PC is not required to utilize every possible source of software information on its endpoint. Some SWIMA-PCs might be explicitly tied only to one or a handful of software inventory sources, or it could be designed to dynamically accommodate new sources. For all software inventory evidence sources that a particular SWIMA-PC supports, it MUST completely support all requirements of this specification with regard to those sources. A potential source that cannot support some set of required functionality (e.g., it is unable to monitor the software it reports for change events, as discussed in Section 3.6) MUST NOT be used as a source of endpoint software inventory information, even if it could provide some information. In other words, a source either supports full functionality as described in this specification, or it cannot be used at all.

When sending information about installed software the SWIMA-PC MUST include the complete set of relevant data from all supported sources of software inventory evidence. In other words, sources need to be used consistently. This is because, if a particular source is included in an initial inventory, but excluded from a later inventory, the SWIMA-PV receiving this information might reasonably conclude that the software reported by that source was no longer installed on the endpoint. As such, it is important that all supported sources be used every time the SWIMA-PC provides information to a SWIMA-PV.

Note that, if a SWIMA-PC collects data from multiple sources, it is possible that some software products might be "double counted". This can happen if both sources of inventory evidence provide a record for a single installation of a software product. When a SWIMA-PC reports information or records events from multiple inventory evidence sources, it MUST use the information those sources provide, rather than attempting to perform some form of reduction. In other words, if multiple sources report records corresponding to a single installation of a software product, all such records from each source are required to be part of the SWIMA-PC’s processing even if this might lead to multiple reporting, and the SWIMA-PC is not to ignore some records to avoid such multiple reporting.

All inventory records reported by a SWIMA-PC include a Source Identifier linking them to a particular source. Source Identifiers are discussed more in Section 3.4.5.

3.2. Data Models

SWIMA conveys records about software presence from a SWIMA-PC to a SWIMA-PV. SWIMA does not manage the actual generation or collection of such records on the endpoint. As a result, information available to SWIMA-PCs might come in a variety of formats, and a SWIMA-PC could have little control over the format of the data made available to it. Because of this, SWIMA places no constraints on the format of these generated records and supports an open set of record formats by which installed software instances can be described. The following terms are used in this document:

Data model - The format used to structure data within a given record. SWIMA does not constrain the data models it conveys.

Record - A populated instance of some data model that describes a software product.

Do not confuse the "data model" described here with the structure of the SWIMA messages and attributes used to convey information between SWIMA-PVs and PCs. The SWIMA specification dictates the structure of its messages and attributes. Some attributes, however, have specific fields used to convey inventory records, and those fields support an extensible list of data models for their values. In other words, SWIMA data models provide an extension point within SWIMA attributes that allows the structure of inventory records to evolve.

The data model used to structure software inventory information has very little impact on the behavior of the components defined in this specification. The SWIMA-PV has no dependency on the data model of records conveyed in SWIMA messages. For this reason, it MUST NOT reject a message or respond with a PA-TNC Error due to the data model used to structure records in attributes it receives. Similarly, it MUST NOT reject a message or respond with a PA-TNC Error if a record fails to comply with a stated format, unless that failure prevents correct parsing of the attribute itself. In short, the record bodies are effectively treated as "black boxes" by the SWIMA-PV. (Note that the SWIMA-PV might serve as the front-end of other functionality that does have a dependency on the data model used to structure software information, but any such dependency is beyond the scope of this specification and needs to be addressed outside the behaviors specified in this document. This specification is only concerned with collection and delivery of software inventory information; components that consume and use this information are a separate concern.)

The SWIMA-PC does have one functional dependency on the data models used in the software records it delivers, but only insofar as it is required to deterministically create a Software Identifier (described in Section 3.4.1) based on each record it delivers. The SWIMA-PC MUST be able to generate a Software Identifier for each record it delivers, and if the SWIMA-PC cannot do so the record cannot be delivered by the SWIMA-PC. All SWIMA-PCs MUST at least be able to generate Software Identifiers for the data model types specified in Section 6 of this document. A SWIMA-PC MAY include the ability to generate Software Identifiers for other data model types, and thus be able to support them as well.

3.3. Basic Attribute Exchange

In the most basic exchange supported by this specification, a SWIMA-PV sends a request to the SWIMA-PC requesting some type of information about the endpoint’s software inventory. This simple exchange is shown in Figure 2.

```
+-------------+                          +--------------+
|  SWIMA-PC   |                          |   SWIMA-PV   |  Time
+-------------+                          +--------------+   |
      |                                          |          |
      |<------------SWIMA Request--------------|          |
      |                                          |          |
      |-------------SWIMA Response------------->|          |
      |                                          |          V
```

Figure 2: Basic SWIMA Attribute Exchange

Upon receiving such a SWIMA Request from the SWIMA-PV, the SWIMA-PC is expected to collect all the relevant software inventory information from the endpoint’s software evidence collection and place it within its response attribute.

SWIMA-PVs MUST discard without error any SWIMA Response attributes that they receive for which they do not know the SWIMA Request parameters that led to this SWIMA Response. This is due to the fact that the SWIMA Request includes parameters that control the nature of the response (as will be described in the following sections) and without knowing those parameters the SWIMA Response cannot be reliably interpreted. Most often receiving an unsolicited SWIMA Response attribute happens when a NEA Server has multiple SWIMA-PVs; one SWIMA-PV sends a SWIMA Request but, unless exclusive delivery is set by the sender and honored by the recipient, both SWIMA-PVs receive copies of the resulting SWIMA Response. In this case, the SWIMA-PV that didn’t send the SWIMA Request would lack the context necessary to correctly interpret the SWIMA Response it received and would simply discard it. Note, however, that proprietary measures might allow a SWIMA-PV to discover the SWIMA Request parameters for a SWIMA Response even if that SWIMA-PV did not send the given SWIMA Request. As such, there is no blanket requirement for a SWIMA-PV to discard all SWIMA Responses to SWIMA Request the SWIMA-PV did not generate itself, only that SWIMA-PVs are required to discard SWIMA Responses for which they cannot get the necessary context to interpret.

In the case that it is possible to do so, the SWIMA-PC SHOULD send its SWIMA Response attribute to the SWIMA-PV that requested it using exclusive delivery as described in section 4.5 of RFC 5793 (PB-TNC) [RFC5793]. Exclusive delivery requests that only the sender of the SWIMA Request receives the resulting SWIMA Response. Note, however, that PB-TNC does not require the recipient to honor the exclusive delivery flag in messages that it receives, so setting the flag cannot be guaranteed to prevent a SWIMA-PV from receiving unsolicited SWIMA Responses.

All numeric values sent in SWIMA messages are sent in network (big endian) byte order.

3.4. Core Software Reporting Information

Different parameters in the SWIMA Request can influence what information is returned in the SWIMA Response. However, while each SWIMA Response provides different additional information about this installed software, they all share a common set of fields that support reliable software identification on an endpoint. These fields include: Software Identifiers, the Data Model Type, Record Identifiers, Software Locators, and Source Identifiers. These fields are present for each reported piece of software in each type of SWIMA Response. The following sections examine these information types in more detail.

3.4.1. Software Identifiers

A Software Identifier uniquely identifies a specific version of a specific software product. The SWIMA specification does not dictate the structure of a Software Identifier (beyond stating that it is a string) or define how it is created. Instead, each data model described in the Software Data Model IANA table (Section 10.4) includes its own rules for how a Software Identifier is created based on a record in the Endpoint's Software Inventory Evidence Collection expressed in that data model. Other data models will have their own procedures for the creation of associated Software Identifiers. Within the SWIMA specification, the Software Identifier is simply an opaque string and there is never any need to unpack any information that might be part of that identifier.

A Software Identifier is a fraction of the size of the inventory record from which it is derived. For some combinations of data models and sources, the full record might never be necessary as the identifier can be directly correlated to the contents of the full record. This is possible with authoritative SWID tags, since these tags always have the same contents and thus a Software Identifier derived from these tags can be used as a lookup to a local copy of the full tag. For other combinations of source and data model, a server might not be able to determine the specific software product and version associated with the identifier without requesting delivery of the full record. However, even in those cases, downstream consumers of this information might never need the full record as long as the Software Identifiers they receive can be tracked reliably. A SWIMA-PV can use Software Identifiers to track the presence of specific software products on an endpoint over time in a bandwidth-efficient manner.

There are two important limitations of Software Identifiers to keep in mind:

1. The identifiers do not necessarily change when the associated record changes. In some situations, a record in the endpoint's Software Inventory Evidence Collection will change due to new information becoming available or in order to correct prior errors in that information. Such changes might or might not result in changes to the Software Identifier, depending on the nature of the changes and the rules governing how Software Identifiers are derived from records of the appropriate data model.

2. It is possible that a single software product is installed on a single endpoint multiple times. If both of these installation instances are reported by the same source using the same data format, then this can result in identical Software Identifiers for each installation instances. In other words, Software Identifiers might not uniquely identify installation instances; they just are intended to uniquely identify software products (which might have more than one installation instance). Instead, to reliably distinguish between multiple instances of a single software product, one needs to make use of Record Identifiers, described in Section 3.4.3.

3.4.2. Data Model Type

The Data Model Type consists of two fields: the Data Model Type PEN and the Data Model Type. The combination of these fields is used to identify the type of data model of the associated software inventory record. The data model is significant not only because it informs the recipient of the data model of the associated record, but because the process for generation of the Software Identifier for the record depends on the record's data model. Clearly identifying the type of data model from which the Software Identifier was derived thus provides useful context for that value.

The PEN (or Private Enterprise Number) identifies the organization that assigns meaning to the Data Model Type field value. PENs are managed by IANA in the Private Enterprise Numbers registry. PENs allow vendors to designate their own set of data models for software inventory description. IANA reserves the PEN of 0x000000. Data Model Types associated with this PEN are defined in the Software Data Model IANA table, created in Section 10.4 of this specification. Note that this IANA table reserves all values greater than or equal to 0xC0 (192) for enterprise use. This means that local enterprises can use custom data formats and indicate them with the IANA PEN and a Data Model Type value between 0xC0 and 0XFF, inclusive. Those enterprises are responsible for configuring their SWIMA-PCs to correctly report those custom data models.

3.4.3. Record Identifiers

A Record Identifier is a 4-byte unsigned integer generated by the SWIMA-PC that is uniquely associated with a specific record within the Endpoint’s Software Inventory Evidence Collection. The SWIMA-PC MUST assign a unique identifier to each record when it is added to the Endpoint’s Software Inventory Evidence Collection. The Record Identifier SHOULD remain unchanged if that record is modified. (However, it is recognized that, in some circumstances, record modification might be hard to distinguish from record deletion followed by creation of a new record. For this reason, retaining a constant Record Identifier across record modification is recommended but not required.) The SWIMA-PC might wish to assign Record Identifiers sequentially, but any scheme is acceptable provided that no two records receive the same identifier.

Servers can use Record Identifiers to distinguish between multiple instances of a single software product installed on an endpoint. Since each installation instance of a software product is associated with a separate record, servers can use the record identifier to distinguish between instances. For example, if an event is reported (as described in Section 3.7), the record identifier will allow the server to discern which instance of a software product is involved.

3.4.4. Software Locators

In addition to the need to identify software products, many use cases of inventory information need to know where software is located on the endpoint. This information might be needed to direct remediation actions or similar processes. For this reason, every reported software product also includes a Software Locator to identify where the software is installed on the endpoint.

If the location is not provided directly by the record source the SWIMA-PC is responsible for attempting to determine the location of the software product. The "location" of a product SHOULD be the directory in which the software products’ executables are kept. The source and/or SWIMA-PC MUST be consistent in reporting the location of a software product (i.e., assuming a software product has not moved, the SWIMA-PC cannot use one location in one report and a different location for the same software product in another).

The location is expressed as a URI string. The string MUST conform to URI syntax requirements. [RFC3986] The URI schema describes the context of the described location. For example, in most cases the location of the installed software product will be expressed in terms of its path in the filesystem. For such locations, the location URI scheme MUST be "file" and the URI MUST conform to the "file" URI scheme standard [RFC8089] including percent-encoding of whitespace and other special characters. It is possible that other schemes could be used to represent other location contexts. Apart from specifying the use of the "file" scheme, this specification does not identify other schemes or define their use. When representing software products in other location contexts, tools MUST be consistent in their use of schemes, but the exact schemes are not normatively defined here. SWIMA implementations are not limited to the IANA list of URI schemes [1] and can define new schemes to support other types of application locations.

It is possible, that a SWIMA-PC is unable to determine the location of a reported software product. In this case, the SWIMA-PC MUST provide a zero-length Software Locator.

3.4.5. Source Identifiers

All SWIMA-PCs MUST track the source of each piece of software information they report. Each time a SWIMA-PC gets information to send to a given SWIMA-PV from a new source (from the perspective of that SWIMA-PV), the SWIMA-PC MUST assign that source a Source Identification Number, which is an 8-bit unsigned integer. Each item reported includes the Source Identification Number that provided that information. All information that is provided by that source, MUST be delivered with this same Source Identification Number. This MUST be done for each source used. If the SWIMA-PC ever is unclear as to whether a given source is new or not, it MUST assume that this is a new source and assign it a new Source Identification Number. Source Identification Numbers do not need to be assigned sequentially. SWIMA does not support the presence of more than 256 sources as the chance that a single endpoint will have more than 256 methods of collecting inventory information is vanishingly small. All possible values between 0 and 255 are valid; there are no reserved Source Identification Numbers

Source Identification Numbers can help with (although will not completely eliminate) the challenges posed by multiple reporting of a single software instance: since a single source would only report an instance or event once, if multiple reports of a similar instance come from multiple sources, this might be an instance of multiple reporting (although it still might not be so). On the other hand, if multiple instances are reported by a single source, this almost certainly means that there are actually multiple instances that are being legitimately reported.

The SWIMA-PC is responsible for tracking associations between Source Identifiers and the given data source. This association MUST remain consistent with regard to a given SWIMA-PV while there is an active PB-TNC session with that SWIMA-PV. The SWIMA-PC MAY have a different Source Identifier association for different SWIMA-PVs. Likewise, the SWIMA-PC MAY change the Source Identifier association for a given SWIMA-PV if the PB-TNC session terminates. However, implementers of SWIMA-PCs will probably find it easier to manage associations by maintaining the same association for all SWIMA-PVs and across multiple sessions.

Of special note, events records reported from the SWIMA-PC’s event log (discussed in Section 3.7) also need to be sent with their associated data source. The Source Identifier reported with events MUST be the current (i.e., at the time the event is sent) Source Identifier associated with the data source that produced the event, regardless of how long ago that event occurred. Event logs are likely to persist far longer than a single PB-TNC session. SWIMA-PCs MUST ensure that each event can be linked to the appropriate data source, even if the Source Identifiers used when the event was created have since been reassigned. In other words, when sending an event, it needs to be sent with the Source Identifier currently linked to the data source that produced it, regardless of whether a different Source Identifier would have been associated with the event when the event was first created.

Note that the Source Identification Number is primarily used to support recognition, rather than identification, of sources. That is to say, a Software Identification Number can tell a recipient that two events were reported by the same source, but will not necessarily help that recipient determine which source was used. Moreover, different SWIMA-PCs will not necessarily use the same Source Identification Numbers for the same sources. SWIMA-PCs MUST track the assignment of Source Identification Numbers to ensure consistent application thereof. SWIMA-PCs MUST also track which Source

Identification Numbers have been used with each SWIMA-PV with which they communicate.

3.4.6. Using Software and Record Identifiers in SWIMA Attributes

A SWIMA Attribute reporting an endpoint's Software Inventory Evidence Collection always contains the Software Identifiers associated with the identified software products. A SWIMA Attribute might or might not also contain copies of software inventory evidence records. The attribute exchange is identical to the diagram shown in Figure 2 regardless of whether software inventory evidence records are included. The SWIMA Request attribute indicates whether the response is required to include software inventory evidence records. Excluding software inventory evidence records can reduce the attribute size of the response by multiple orders of magnitude when compared to sending the same inventory with full records.

3.5. Targeted Requests

Sometimes a SWIMA-PV does not require information about every piece of software on an endpoint but only needs to receive updates about certain software instances. For example, enterprise endpoints might be required to have certain software products installed and to keep these updated. Instead of requesting a complete inventory just to see if these products are present, the SWIMA-PV can make a "targeted request" for the software in question.

Targeted requests follow the same attribute exchange described in Figure 2. The SWIMA-PV targets its request by providing one or more Software Identifiers in its SWIMA Request attribute. The SWIMA-PC MUST then limit its response to contain only records that match the indicated Software Identifier(s). This allows the network exchange to exclude information that is not relevant to a given policy question, thus reducing unnecessary bandwidth consumption. The SWIMA-PC's response might or might not include software inventory evidence records, depending on the parameters of the SWIMA Request.

Note that targeted requests identify the software relevant to the request only through Software Identifiers. This specification does not support arbitrary, parameterized querying of records. For example, one cannot request all records from a certain software publisher, or all records created by a particular record source. Targeted requests only allow a requestor to request specific software (as identified by their Software Identifiers) and receive a response that is limited to the named software.

There is no assumption that a SWIMA-PC will recognize "synonymous records" - that is, records from different sources for the same software. Recall that different sources and data models may use different Software Identifier strings for the same software product. The SWIMA-PC returns only records that match the Software Identifiers in the SWIMA Request, even if there might be other records in the endpoint’s Software Inventory Evidence Collection for the same software product. This is necessary because SWIMA-PCs might not have the ability to determine that two Software Identifiers refer to the same product.

Targeted requests do not include Record Identifiers or Software Locators. The response to a targeted request MUST include all records associated with the named Software Identifiers, including the case where there are multiple records associated with a single Software Identifier.

SWIMA-PCs MUST accept targeted requests and process them correctly as described above. SWIMA-PVs MUST be capable of making targeted requests and processing the responses thereto.

3.6. Monitoring Changes in an Endpoint’s Software Inventory Evidence Collection

The software collection on an endpoint is not static. As software is installed, uninstalled, patched, or updated, the Software Inventory Evidence Collection is expected to change to reflect the new software state on the endpoint. Different record sources might update the evidence collection at different rates. For example, a package manager might update its records in the Software Inventory Evidence Collection immediately whenever it is used to add or remove a software product. By contrast, sources that perform periodic examination of the endpoint would likely only update their records in the Software Inventory Evidence Collection after each examination.

All SWIMA-PCs MUST be able to be able to detect changes to the Software Inventory Evidence Collection. Specifically, SWIMA-PCs MUST be able to detect:

- The creation of records

- The deletion of records

- The alteration of records

An "alteration" is anything that modifies the contents of a record (or would modify it, if the record is dynamically generated on demand) in any way, regardless of whether the change is functionally meaningful.

SWIMA-PCs MUST detect such changes to the endpoint's Software Inventory Evidence Collection in close to real-time (i.e., within seconds) when the Posture Collector is operating. In addition, in the case where there is a period during which the SWIMA-PC is not operating, the SWIMA-PC MUST be able to determine the net change to the endpoint's Software Inventory Evidence Collection over the period it was not operational. Specifically, the "net change" represents the difference between the state of the endpoint's Software Inventory Evidence Collection when the SWIMA-PC was last operational and monitoring its state, and the state of the endpoint's software inventory evidence collection when the SWIMA-PC resumed operation. Note that a net change might not reflect the total number of change events over this interval. For example, if a record was altered three times during a period when the SWIMA-PC was unable to monitor for changes, the net change of this interval might only note that there was an alteration to the record, but not how many individual alteration events occurred. It is sufficient for a SWIMA-PC's determination of a net change to note that there was a difference between the earlier and current state rather than enumerating all the individual events that allowed the current state to be reached.

The SWIMA-PC MUST assign a time to each detected change in the endpoint's Software Inventory Evidence Collection. These timestamps correspond to the SWIMA-PC's best understanding as to when the detected change occurred. For changes to the endpoint's Software Inventory Evidence Collection that occur while the SWIMA-PC is operating, the SWIMA-PC ought to be able to assign a time to the event that is accurate to within a few seconds. For changes to the endpoint's Software Inventory Evidence Collection that occur while the SWIMA-PC is not operational, upon becoming operational the SWIMA-PC needs to make a best guess as to the time of the relevant events (possibly by looking at timestamps on files), but these values might be off. In the case of dynamically generated records, the time of change is the time at which the data from which the records are generate changes, not the time at which a changed record is generated. For example, if records are dynamically generated based on data in an RPM database, the time of change would be when the RPM database changed.

With regard to deletions of records, the SWIMA-PC needs to detect the deletion and MUST retain a copy of the full deleted record along with the associated Record Identifier and Software Locator so that the record and associated information can be provided to the SWIMA-PV upon request. This copy of the record MUST be retained for a reasonable amount of time. Vendors and administrators determine what "reasonable" means, but a copy of the record SHOULD be retained for as long as the event recording the deletion of the record remains in the SWIMA-PC's event log (as described in Section 3.7). This is recommended because, as long as the event is in the SWIMA-PC's change logs, the SWIMA-PC might send an event attribute (described in Section 3.7) that references this record, and a copy of the record is needed if the SWIMA-PV wants a full copy of the relevant records. In the case that a SWIMA-PC is called upon to report a deletion event that is still in the event log but where the record itself is no longer available, the SWIMA-PC will still return an entry corresponding to the deletion event, but the field of that entry that would normally contain the full copy of the record SHOULD be 0-length.

With regard to alterations to a record, SWIMA-PCs MUST detect any alterations to the contents of a record. Alterations need to be detected even if they have no functional impact on the record. A good guideline is that any alteration to a record that might change the value of a hash taken on the record's contents needs to be detected by the SWIMA-PC. A SWIMA-PC might be unable to distinguish modifications to the content of a record from modifications to the metadata the file system associates with the record. For example, a SWIMA-PC might use the "last modification" timestamp as an indication of alteration to a given record, but a record's last modification time can change for reasons other than modifications to the record contents. A SWIMA-PC is still considered compliant with this specification if it also reports metadata change events that do not change the record itself as alterations to the record. In other words, while SWIMA-PC authors are encouraged to exclude modifications that do not affect the bytes within the record, discriminating between modifications to file contents and changes to file metadata can be difficult and time consuming on some systems. As such, as long as the alterations detected by a SWIMA-PC always cover all modifications to the contents of record, the SWIMA-PC is considered compliant even if it also registers alterations that do not modify the contents of a record as well. When recording an alteration to a record, the SWIMA-PC is only required to note that an alteration occurred. The SWIMA-PC is not required to note or record how the record altered, nor is it possible to include such details in SWIMA Attributes reporting the change to a SWIMA-PV. There is no need to retain a copy of the original record prior to the alteration.

When a record changes it SHOULD retain the same Record Identifier. The Software Locator might or might not change, depending on whether the software changed locations during the changes that led to the record change. A record change MUST retain the same Software Identifier. This means that any action that changes a software product (e.g., application of a patch that results in a change to the product's version) MUST NOT be reflected by a record change but instead MUST result in the deletion of the old record and the creation of a new record. This reflects the requirement that a record in the endpoint’s Software Inventory Evidence Collection correspond directly with an instance of a specific software product.

3.7. Reporting Change Events

As noted in Section 3.6, SWIMA-PCs are required to detect changes to the endpoints Software Inventory Evidence Collection (creation, deletion, and alteration) in near real-time while the SWIMA-PC is operational, and MUST be able to account for any net change to the endpoint’s Software Inventory Evidence Collection that occurs when the SWIMA-PC is not operational. However, to be of use to the enterprise, the NEA Server needs to be able to receive these events and be able to understand how new changes relate to earlier changes. In SWIMA, this is facilitated by reporting change events. All SWIMA-PCs MUST be capable of receiving requests for change events and sending change event attributes. All SWIMA-PVs MUST be capable of requesting and receiving change event attributes.

3.7.1. Event Identifiers

To be useful, change events need to be correctly ordered. Ordering of events is facilitated by two pieces of information: an Event Identifier (EID) value and an EID Epoch value.

An EID is a 4-byte unsigned integer that the SWIMA-PC assigns sequentially to each observed event (whether detected in real-time or deduced by looking for net changes over a period of SWIMA-PC inactivity). All EIDs exist within the context of some "EID Epoch", which is also represented as a 4-byte unsigned integer. EID Epochs are used to ensure synchronization between the SWIMA-PC and any SWIMA-PVs with which it communicates. EID Epoch values MUST be generated in such a way as to minimize the chance that an EID Epoch will be reused, even in the case where the SWIMA-PC reverts to an earlier state. For this reason, sequential EID Epochs are discouraged, since loss of state could result in value reuse. There are multiple reasons that a SWIMA-PC might need to deliberately reset its EID counter, including exhaustion of available EID values, the need to purge entries from the event log to recover memory, or corruption of the event log. In all case where a SWIMA-PC needs to reset its EID counter, a new EID Epoch MUST be selected.

Within an Epoch, EIDs MUST be assigned sequentially, so that if a particular event is assigned an EID of N, the next observed event is given an EID of N+1. In some cases, events might occur simultaneously, or the SWIMA-PC might not otherwise be able to determine an ordering for events. In these cases, the SWIMA-PC creates an arbitrary ordering of the events and assigns EIDs according to this ordering. Two change events MUST NOT ever be assigned the same EID within the same EID Epoch. No meaningful comparison can be made between EID values of different Epochs.

The EID value of 0 is reserved and MUST NOT be associated with any event. Specifically, an EID of 0 in a SWIMA Request attribute indicates that a SWIMA-PV wants an inventory response rather than an event response, while an EID of 0 in a SWIMA Response is used to indicate the initial state of the endpoint's Software Inventory Evidence Collection prior to the observation of any events. Thus the very first recorded event in a SWIMA-PC's records within an EID Epoch MUST be assigned a value of 1. Note that EID and EID Epoch values are assigned by the SWIMA-PC without regard to whether events are being reported to one or more SWIMA-PVs. The SWIMA-PC records events and assigns EIDs during its operation. All SWIMA-PVs that request event information from the SWIMA-PC will have those requests served from the same event records and thus will see the same EIDs and EID Epochs for the same events.

If a SWIMA-PC uses multiple sources, a SWIMA-PC's assignment of EIDs MUST reflect the presence and order of all events on the endpoint (at least for supported sources) regardless of the source. This means that if source A experiences an event, and then source B experiences two events, and then source A experiences another two events, the SWIMA-PC is required to capture five events with consecutive EID values reflecting the order in which the events occur.

The SWIMA-PC MUST ensure there is no coverage gap (i.e., change events that are not recorded in the SWIMA-PC's records) in its change event records. This is necessary because a coverage gap might give a SWIMA-PV a false impression of the endpoint's state. For example, if a SWIMA-PV saw an event indicating that a particular record had been added to the endpoint's software inventory evidence collection, and saw no subsequent events indicating that record had been deleted, it might reasonably assume that this record was still present and thus that the indicated software was still installed (assuming the Epoch has not changed). If there is a coverage gap in the SWIMA-PC's event records, however, this assumption could be false. For this reason, the SWIMA-PC's event records MUST NOT contain gaps. In the case where there are periods where it is possible that changes occurred without the SWIMA-PC detecting or recording them, the SWIMA-PC MUST either compute a net change and update its event records appropriately, or pick a new EID Epoch to indicate a discontinuity with previous event records.

Within a given Epoch, once a particular event has been assigned an EID, this association MUST NOT be changed. That is, within an Epoch, once an EID is assigned to an event, that EID cannot be reassigned to a different event, and the event cannot be assigned a different EID.

When the SWIMA-PC’s Epoch changes, all of these associations between EIDs and events are cancelled, and EID values once again become free for assignment.

3.7.2. Core Event Tracking Information

Whether reporting events or full inventories it is important to know how the reported information fits into the overall timeline of change events. This is why all SWIMA Response attributes include fields to place that response within the sequence of detected events. Specifically, all SWIMA Responses include a Last EID and EID Epoch field. The EID Epoch field identifies the EID Epoch in which the SWIMA Response was sent. If the SWIMA Response is reporting events, all reported events occurred within the named EID Epoch. The Last EID (which is also always from the named EID Epoch) indicates the EID of the last recorded change event at the time that the SWIMA Response was sent. These two fields allow any response to be placed in the context of the complete set of detected change events within a given EID Epoch.

3.7.3. Updating Inventory Knowledge Based on Events

Modern endpoints can have hundreds of software products installed, most of which are unlikely to change from one day to the next. As such, instead of exchanging a complete list of an endpoint’s inventory on a regular basis, one might wish to only identify changes since some earlier known state of this inventory. This is readily facilitated by the use of EIDs to place change events in a context relative to earlier state.

As noted above, every SWIMA Response sent by a SWIMA-PC to a SWIMA-PV (as described in Section 3.3 through Section 3.5) includes the EID Epoch and EID of the last event recorded prior to that response being compiled. This allows the SWIMA-PV to place all subsequently received event records in context relative to this SWIMA Response attribute (since the EIDs represent a total ordering of all changes to the endpoint’s software inventory evidence collection). Specifically, a SWIMA-PV (or, more likely, a database that collects and records its findings) can record an endpoint’s full inventory and also the EID and Epoch of the most recent event reflected at the time of that inventory. From that point on, if change events are observed, the attribute describing these events indicates the nature of the change, the affected records, and the order in which these events occurred (as indicated by the sequential EIDs). Using this information, any remote record of the endpoint’s Software Inventory Evidence Collection can be updated appropriately.

3.7.4. Using Event Records in SWIMA Attributes

A SWIMA-PV MUST be able to request a list of event records instead of an inventory. The attribute flow in such an exchange looks the same as the basic flow shown in Figure 2. The only difference is that, in the SWIMA Request attribute, the SWIMA-PV provides an EID other than 0. (A value of 0 in these fields represents a request for an inventory.) When the SWIMA-PC receives such a request, instead of identifying records from the endpoint's Software Inventory Evidence Collection, it consults its list of detected changes. The SWIMA-PC MUST add an event record to the SWIMA Response attribute for each recorded change event with an EID greater than or equal to the EID in the SWIMA Request attribute (although targeting of requests, as described in the next paragraph, might limit this list). A list of event records MUST only contain events with EIDs that all come from the current Epoch.

SWIMA-PVs can target requests for event records by including one or more Software Identifiers, as described in Section 3.5, in the SWIMA Request that requests an event record list. A targeted request for event records is used to indicate that only events affecting software that matches one of the provided Software Identifiers are to be returned. Specifically, in response to a targeted request for event records, the SWIMA-PC MUST exclude any event records that are less than the indicated EID (within the current EID Epoch) and exclude any event records where the affected software does not match one of the provided Software Identifiers. This might mean that the resulting list of event records sent in the response attribute does not provide a continuous sequence of EIDs. Both SWIMA-PCs and SWIMA-PVs MUST support targeted request for event records.

3.7.5. Partial and Complete Lists of Event Records in SWIMA Attributes

Over time, a SWIMA-PC might record a large number of change events. If a SWIMA-PV requests all change events covering a large period of time, the resulting SWIMA Response attribute might be extremely large, especially if the SWIMA-PV requests inclusion of software inventory evidence records in the response. In the case that the resulting attribute is too large to send (either because it exceeds the 4GB attribute size limit imposed by the PA-TNC specification, or because it exceeds some smaller size limit imposed on the SWIMA-PC) the SWIMA-PC MAY send a partial list of event records back to the SWIMA-PV.

Generation of a partial list of events in a SWIMA Response attribute requires the SWIMA-PC to identify a "consulted range" of EIDs. A consulted range is the set of event records that are examined for inclusion in the SWIMA Response attribute and that are included in that attribute if applicable. Recall that, if a SWIMA Request is targeted, only event records that involve the indicated software would be applicable. (See Section 3.5 for more on Targeted Request.) If a request is not targeted, all event records in the considered range are applicable and included in the SWIMA Response attribute.

The lower bound of the consulted range MUST be the EID provided in the SWIMA Request. (Recall that a SWIMA Request indicates a request for event records by providing a non-0 EID value in the SWIMA Request. See Section 3.7.4.) The upper bound of the consulted range is the EID of the latest event record (as ordered by EID values) that is included in the SWIMA Response attribute if it is applicable to the request. The EID of this last event record is called the "Last Consulted EID". The SWIMA-PC chooses this Last Consulted EID based on the size of the event record list it is willing to provide to the SWIMA-PV.

A partial result list MUST include all applicable event records within the consulted range. This means that for any applicable event record (i.e., any event record in an un-targeted request, or any event record associated with software matching a requested Software Identifier in a targeted request) whose EID is greater than or equal to the EID provided in the SWIMA Request and whose EID is less than or equal to the Last Consulted EID, that event record MUST be included in the SWIMA Response conveying this partial list of event records. This ensures that every partial list of event records is always complete within its indicated range. Remember that, for targeted requests, "complete" doesn’t mean that all EIDs between the range endpoints are present, but only that every matching EID between the range endpoints is included.

All SWIMA Response attributes that convey event records include a Last Consulted EID field. This is in addition to the EID Epoch and Last EID fields that are present in all SWIMA Responses. Note that, if responding to a targeted SWIMA Request, the SWIMA Response attribute might not contain the event record whose EID matches the Last Consulted EID value. For example, the last consulted EID record might have been deemed inapplicable because it did not match the specified list of Software Identifiers in the SWIMA Request.

If a SWIMA-PV receives a SWIMA Response attribute where the Last EID and Last Consulted EID fields are identical, the SWIMA-PV knows that it has received a result list that is complete, given the parameters of the request, up to the present time.

On the other hand, if the Last EID is greater than the Last Consulted EID, the SWIMA-PV has received a partial result list. (The Last Consulted EID MUST NOT exceed the Last EID.) In this case, if the

SWIMA-PV wishes to try to collect the rest of the partially delivered result list it then sends a new SWIMA Request whose EID is one greater than the Last Consulted EID in the preceding response. Doing this causes the SWIMA-PC to generate another SWIMA Response attribute containing event records where the earliest reported event record is the one immediately after the event record with the Last Consulted EID (since EIDs are assigned sequentially). By repeating this process until it receives a SWIMA Response where the Last EID and Last Consulted EID are equal, the SWIMA-PV is able to collect all event records over a given range, even if the complete set of event records would be too large to deliver via a single attribute.

Implementers need to be aware that a SWIMA Request might specify an EID that is greater than the EID of the last event recorded by a SWIMA-PC. In accordance with the behaviors described in Section 3.7.4, a SWIMA-PC MUST respond to such a request with a SWIMA Response attribute that contains zero event records. This is because the SWIMA-PC has recorded no event records with EIDs greater than or equal to the EID in the SWIMA Request. In such a case, the Last Consulted EID field MUST be set to the same value as the Last EID field in this SWIMA Response attribute. This case is called out because the consulted range on a SWIMA-PC in such a situation is a negative range, where the "first" EID in the range (provided in the SWIMA Request) is greater than the "last" EID in the range (this being the EID of the last recorded event on the SWIMA-PC). Implementers need to ensure that SWIMA-PCs do not experience problems in such a circumstance.

Note that this specification only supports the returning of partial results when returning event records. There is no way to return a partial inventory list under this specification.

3.7.6. Synchronizing Event Identifiers and Epochs

Since EIDs are sequential within an Epoch, if a SWIMA-PV’s list of event records contains gaps in the EID values within a single Epoch, the SWIMA-PV knows that there are events that have not been accounted for. The SWIMA-PV can either request a new event list to collect the missing events or request a full inventory to re-sync its understanding of the state of the endpoint’s Software Inventory Evidence Collection. In either case, after the SWIMA-PV’s record of the endpoint’s Software Inventory Evidence Collection has been updated, the SWIMA-PV can record the new latest EID value and track events normally from that point on.

If the SWIMA-PV receives any attribute from a SWIMA-PC where the EID Epoch differs from the EID Epoch that was used previously, then SWIMA-PV or any entity using this information to track the endpoint’s

Software Inventory Evidence Collection knows that there is a discontinuity in their understanding of the endpoint’s state. To move past this discontinuity and reestablish a current understanding of the state of the endpoint’s Software Inventory Evidence Collection, the SWIMA-PV needs to receive a full inventory from the endpoint. The SWIMA-PV cannot be brought in sync with the endpoint’s state through the collection of any set of event records in this situation. This is because it is not possible to account for all events on the SWIMA-PC since the previous Epoch was used, because there is no way to query for EIDs from a previous Epoch. Once the SWIMA-PV has received a full inventory for the new Epoch, the SWIMA-PV records the latest EID reported in this new Epoch and can track further events normally.

A SWIMA-PC MUST NOT report events with EIDs from any Epoch other than the current EID Epoch. The SWIMA-PC MAY choose to purge all event records from a previous Epoch from memory after an Epoch change. Alternately, the SWIMA-PC MAY choose to retain some event records from a previous EID Epoch and assign them new EIDs in the current Epoch. However, in the case where a SWIMA-PC chooses the latter option it MUST ensure that the order of events according to their EIDs is unchanged and that there is no coverage gap between the first retained event recorded during the previous Epoch (now reassigned with an EID in the current Epoch) and the first event recorded during the current Epoch. In particular, the SWIMA-PC MUST ensure that all change events that occurred after the last recorded event from the previous Epoch are known and recorded. (This might not be possible if the Epoch change is due to state corruption on the SWIMA-PC.) A SWIMA-PC might choose to reassign EIDs to records from a preceding Epoch to create a "sliding window" of events, where each Epoch change represents a shift in the window of available events.

In the case where a SWIMA-PC suffers a crash and loses track of its current EID Epoch or current EID, then it MUST generate a new EID Epoch value and begin assigning EIDs within that Epoch. In this case, the SWIMA-PC MUST purge all event records from before the crash as it cannot ensure that there is not a gap between the last of those records and the next detected event. The process for generating a new EID Epoch MUST minimize the possibility that the newly generated EID Epoch is the same as a previously used EID Epoch.

The SWIMA-PV will normally never receive an attribute indicating that the latest EID is less than the latest EID reported in a previous attribute within the same EID Epoch. If this occurs, the SWIMA-PC has suffered an error of some kind, possibly indicative of at least partial corruption of its event log. In this case, the SWIMA-PV MUST treat the situation as if there was a change in Epoch and treat any local copy of the endpoint’s Software Inventory Evidence Collection as out-of-sync until a full inventory can be reported by the SWIMA-PC. In this case, the SWIMA-PV SHOULD log the occurrence so the SWIMA-PC can be examined to ensure it is now operating properly.

3.8. Subscriptions

Thus far, all attribute exchanges discussed assume that a SWIMA-PV sent an SWIMA Request attribute and the SWIMA-PC is providing a direct response to that request. The SWIMA specification also supports the ability for a SWIMA-PC to send a SWIMA Response to the SWIMA-PV in response to observed changes in the endpoint's software inventory evidence collection, instead of in direct response to a SWIMA Request. An agreement by a SWIMA-PC to send content when certain changes are detected to the endpoint's Software Inventory Evidence Collection is referred to in this specification as a "subscription", and the SWIMA-PV that receives this content is said to be "subscribed to" the given SWIMA-PC. All SWIMA-PCs and SWIMA-PVs MUST support the use of subscriptions.

3.8.1. Establishing Subscriptions

A SWIMA-PV establishes a subscription on a particular SWIMA-PC by sending a SWIMA Request attribute with the Subscription flag set. The SWIMA Request attribute is otherwise identical to the SWIMA Requests discussed in previous sections. Specifically, such a SWIMA Request might or might not request inclusion of software inventory evidence records, might or might not be targeted, and might request change event records or endpoint inventory. Assuming no error is encountered, a SWIMA-PC MUST send a SWIMA Response attribute in direct response to this SWIMA Request attribute, just as if the Subscription flag was not set. As such, the attribute exchange that establishes a new subscription in a SWIMA-PC has the same flow seen in the previous attribute exchanges, as depicted in Figure 2. If the SWIMA-PV does not receive a PA-TNC Error attribute (as described in Section 3.9 and Section 5.16) in response to their subscription request, the subscription has been successfully established on the SWIMA-PC. The SWIMA Request attribute that establishes a new subscription is referred to as the "establishing request" for that subscription.

When a subscription is established it is assigned a Subscription ID value. The Subscription ID is equal to the value of the Request ID of the establishing request. (For more about Request IDs, see Section 5.6.)

A SWIMA-PC MUST have the ability to record and support at least 8 simultaneous subscriptions and SHOULD have the ability to support more than this. These subscriptions might all come from a single

SWIMA-PV, might all be from different SWIMA-PVs, or might be a mix. In the case that a SWIMA-PC receives a subscription request but is unable to support an additional subscription, it MUST respond to the request with a PA-TNC Error attribute of type SWIMA_SUBSCRIPTION_DENIED_ERROR.

A SWIMA-PV MUST have the ability to record and support at least 256 simultaneous subscriptions and SHOULD have the ability to support more than this. Any number of these subscriptions might be to the same SWIMA-PC and any number of these subscriptions might be to different SWIMA-PCs.

3.8.2. Managing Subscriptions

The SWIMA-PC MUST record each accepted subscription along with the identity of the party to whom attributes are to be pushed in compliance with the subscription. This identity includes both the NEA Server's connection ID and the Posture Validator Identifier from the PB-PA message that delivered the request.

Likewise, SWIMA-PVs MUST record each accepted subscription for which they are the subscribing party along with the associated Subscription ID and the identity of the SWIMA-PC that will be fulfilling the subscription. The SWIMA-PV needs to retain this information in order to correctly interpret pushed SWIMA Response attributes sent in fulfillment of the subscription. The identity of the SWIMA-PC is given in the Posture Collector Identifier of the PB-PA message header in all messages from that SWIMA-PC.

3.8.3. Terminating Subscriptions

Subscriptions MAY be terminated at any time by the subscribing SWIMA-PV by setting the Clear Subscriptions flag in a SWIMA Request. (See Section 5.7 for more on using this flag.) In the case that a SWIMA Request with the Clear Subscriptions flag set is received the SWIMA-PC MUST only clear subscriptions that match both the NEA server connection ID and the SWIMA-PV ID for this SWIMA Request, and MUST clear all such subscriptions.

This specification does not give the SWIMA-PV the ability to terminate subscriptions individually - all subscriptions to the SWIMA-PV are cleared when the Clear Subscriptions flag is set.

This specification does not give the SWIMA-PC the ability to unilaterally terminate a subscription. However, if the SWIMA-PC experiences a fatal error fulfilling a subscription, resulting in sending a PA-TNC Error attribute of type SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR, then the subscription whose fulfillment led to the error MUST be treated as terminated by both the SWIMA-PC and the SWIMA-PV. Only the subscription experiencing the error is cancelled and other subscriptions are unaffected. See Section 3.9 for more on this error condition.

Finally, a subscription is terminated if the connection between the SWIMA-PC and SWIMA-PV is deleted. This occurs when the connection ID used in the messages between the SWIMA-PC and the SWIMA-PV becomes unbound. Loss of this connection ID would prevent the SWIMA-PC from sending messages in fulfillment of this subscription. As such, loss of the connection ID necessarily forces subscription termination between the affected parties.

3.8.4. Subscription Status

A SWIMA-PV can request that a SWIMA-PC report the list of active subscriptions for which the SWIMA-PV is the subscriber. A SWIMA-PV can use this to recover lost information about active subscriptions. A SWIMA-PV can also use this capability to verify that a SWIMA-PC has not forgotten any of its subscriptions. The latter is especially useful where a SWIMA-PC does not send any attributes in fulfillment of a given subscription for a long period of time. The SWIMA-PV can check the list of active subscriptions on the SWIMA-PC and verify whether the inactivity is due to a lack of reportable events or due to the SWIMA-PC forgetting its obligations to fulfill a given subscription.

A SWIMA-PV requests a list of its subscriptions on a given SWIMA-PC by sending that SWIMA-PC a Subscription Status Request. The SWIMA-PC MUST then respond with a Subscription Status Response (or a PA-TNC Error if an error condition is experienced). The Subscription Status Response MUST contain one subscription record for each of the active subscriptions for which the SWIMA-PV is the subscribing party.

3.8.5. Fulfilling Subscriptions

As noted in Section 3.6 SWIMA-PCs are required to automatically detect changes to an endpoint’s Software Inventory Evidence Collection in near real-time. For every active subscription, the SWIMA-PC MUST send an attribute to the subscribed SWIMA-PV whenever a change is detected to relevant records within the endpoint’s Software Inventory Evidence Collection. Such an attribute is said to be sent "in fulfillment of" the given subscription and any such attribute MUST include that subscription’s Subscription ID. If the establishing request for that subscription was a targeted request, then only records that match the Software Identifiers provided in that establishing request are considered relevant. Otherwise, (i.e., for non-targeted requests) any record is considered relevant for this purpose. Figure 3 shows a sample attribute exchange where a subscription is established and then later attributes are sent from the SWIMA-PC in fulfillment of the established subscription.

```
         +-------------+                   +--------------+
         |  SWIMA-PC   |                   |   SWIMA-PV   |  Time
         +-------------+                   +--------------+   |
                |                                 |           |
                |<----------SWIMA Request-----------|           |
                |                                 |           |
                |-----------SWIMA Response--------->|           |
                |                                 |           |
                .                                 .           .
                .                                 .           .
                .                                 .           .
 <Change Event>|                                 |           |
                |----------SWIMA Response---------->|           |
                |                                 |           |
                .                                 .           .
                .                                 .           .
                .                                 .           .
 <Change Event>|                                 |           |
                |----------SWIMA Response---------->|           |
                |                                 |           V
```

Figure 3: Subscription Establishment and Fulfillment

The contents of an attribute sent in fulfillment of a subscription depend on the parameters provided in the establishing request for that subscription. Specifically, the contents of an attribute sent in fulfillment of a subscription have the same format as would a direct response to the establishing request. For example, if the establishing request stipulated a response that contained an event record list that included software inventory evidence records, all attributes sent in fulfillment of this subscription will also consist of event record lists with software inventory evidence records. As such, all SWIMA Responses displayed in the exchange depicted in Figure 3 have the same format. A SWIMA Response generated in fulfillment of an active subscription MUST be a valid SWIMA Response attribute according to all the rules outlined in the preceding sections. In other words, an attribute constructed in fulfillment of a subscription will look the same as an attribute sent in direct response to an explicit request from a SWIMA-PV that had the same request parameters and which arrived immediately after the given change event. There are a few special rules that expand on this guideline:

3.8.5.1. Subscriptions Reporting Inventories

In the case that a SWIMA-PV subscribes to a SWIMA-PC requesting an inventory attribute whenever changes are detected (i.e., the EID in the establishing request is 0), then the SWIMA-PC MUST send the requested inventory whenever a relevant change is detected. (A "relevant change" is any change for untargeted requests, or a change to an indicated record in a targeted request.) Upon detection of a relevant change for an active subscription, the SWIMA-PC sends the appropriate inventory information as if it had just received the establishing request. Attributes sent in fulfillment of this subscription will probably have a large amount of redundancy, as the same records are likely to be present in each of these SWIMA Attributes. The role of an inventory subscription is not to report records just for the items that changed - that is the role of a subscription that reports events (see Section 3.8.5.2). A SWIMA-PC MUST NOT exclude a record from an attribute sent in fulfillment of an inventory subscription simply because that record was not involved in the triggering event (although a record might be excluded for other reasons, such as if the subscription is targeted - see Section 3.8.5.3).

3.8.5.2. Subscriptions Reporting Events

The way in which a SWIMA-PV indicates it wishes to establish a subscription requesting event records is by providing a non-zero EID in the SWIMA Request establishing the subscription (see Section 3.7.1). However, when the SWIMA-PC constructs an attribute in fulfillment of the subscription (other than the direct response to the establishing request), it MUST only include event records for the detected change(s) that precipitated this response attribute. In other words, it MUST NOT send a complete list of all changes starting with the establishing request’s EID, up through the latest change, every time a new event is detected. In effect, the EID in the establishing request is treated as being updated every time an attribute is sent in fulfillment of this subscription, such that a single event is not reported twice in fulfillment of a single subscription. As such, every SWIMA-PC MUST track the EID of the last event that triggered an attribute for the given subscription. When the next event (or set of events) is detected, the SWIMA-PC MUST only report events with EIDs after the last reported event. In the case that the EID Epoch of the SWIMA-PC changes, the SWIMA-PC MUST reset this EID tracker to 0 (if the event log is completely purged) or to the new EID of the last reported retained event (if the event log is partially purged to create a "sliding window"). Doing this ensures that the SWIMA-PC continues to only send events that have not been previously reported.

Note that while a subscription is active, the subscribing SWIMA-PV MAY make other requests for event records that overlap with events that are reported in fulfillment of a subscription. Such requests are unaffected by the presence of the subscription, nor is the subscription affected by such requests. In other words, a given request will get the same results back whether or not there was a subscription. Likewise, an attribute sent in fulfillment of a subscription will contain the same information whether or not other requests had been received from the SWIMA-PV.

A SWIMA-PV needs to pay attention to the EID Epoch in these attributes, as changes in the Epoch might create discontinuities in the SWIMA-PV's understanding of the endpoint's Software Inventory Evidence Collection state, as discussed in Section 3.7.6. In particular, once the EID Epoch changes, a SWIMA-PV is unable to have confidence that it has a correct understanding of the state of an endpoint's Software Inventory Evidence Collection until after the SWIMA-PV collects a complete inventory.

SWIMA-PCs MAY send partial lists of event records in fulfillment of a subscription. (See Section 3.7.5 for more on partial list of event records.) In the case that a SWIMA-PC sends a partial list of event records in fulfillment of a subscription, it MUST immediately send the next consecutive partial list, and continue doing so until it has sent the equivalent of the complete list of event records. In other words, if the SWIMA-PC sends a partial list it does not wait for another change event to send another SWIMA Response, but continues sending SWIMA Responses until it has sent all event records that would have been included in a complete fulfillment of the subscription. Note that the direct response to the establishing request is not considered to be sent in fulfillment of a subscription. However, in this case the SWIMA-PC MUST treat the presence of unreported events as a triggering event for pushing additional messages in fulfillment of the newly established subscription. As such, the net effect is that, if the direct response to the establishing request (i.e., the Subscription Fulfillment flag is unset) is partial, the SWIMA-PC will immediately follow this with additional attributes (with the Subscription Fulfillment flag set) until the complete set of events has been sent to the SWIMA-PV.

3.8.5.3. Targeted Subscriptions

Subscriptions MAY be targeted to only apply to records that match a given set of Software Identifiers. In the case where changes are detected that affect multiple records, some matching the establishing request's Software Identifiers and some not, the attribute sent in fulfillment of the subscription MUST only include inventory or events (as appropriate) for records that match the establishing request’s Software Identifiers. The SWIMA-PC MUST NOT include non-matching records in the attribute, even if those non-matching records experienced change events that were co-temporal with change events on the matching records.

In addition, a SWIMA-PC MUST send an attribute in fulfillment of a targeted subscription only when changes to the endpoint’s Software Inventory Evidence Collection impact one or more records matching the subscription’s establishing request’s Software Identifiers. A SWIMA-PC MUST NOT send any attribute in fulfillment of a targeted subscription based on detected change to the endpoint’s Software Inventory Evidence Collection that did not involve any of the records targeted by that subscription.

3.8.5.4. No Subscription Consolidation

A SWIMA-PV MAY establish multiple subscriptions to a given SWIMA-PC. If this is the case, it is possible that a single change event on the endpoint might require fulfillment by multiple subscriptions, and that the information included in attributes that fulfill each of these subscriptions might overlap. The SWIMA-PC MUST send separate attributes for each established subscription that requires a response due to the given event. Each of these attributes MUST contain all information required to fulfill that individual subscription, even if that information is also sent in other attributes sent in fulfillment of other subscriptions at the same time. In other words, SWIMA-PCs MUST NOT attempt to combine information when fulfilling multiple subscriptions simultaneously, even if this results in some redundancy in the attributes sent to the SWIMA-PV.

3.8.5.5. Delayed Subscription Fulfillment

A SWIMA-PC MAY delay the fulfillment of a subscription following a change event in the interest of waiting to see if additional change events are forthcoming and, if so, conveying the relevant records back to the SWIMA-PV in a single SWIMA Response attribute. This can help reduce network bandwidth consumption between the SWIMA-PC and the SWIMA-PV. For example, consider a situation where 10 changes occur a tenth of a second apart. If the SWIMA-PC does not delay in assembling and sending SWIMA Response attributes, the SWIMA-PV will receive 10 separate SWIMA Response attributes over a period of 1 second. However, if the SWIMA-PC waits half a second after the initial event before assembling a SWIMA Response, the SWIMA-PV only receives two SWIMA Response attributes over the same period of time.

Note that the ability to consolidate events for a single subscription over a given period of time does not contradict the rules in

Section 3.8.5.4 prohibiting consolidation across multiple subscriptions. When delaying fulfillment of subscriptions, SWIMA-PCs are still required to fulfill each individual subscription separately. Moreover, in the case that change events within the delay window cancel each other out (e.g., a record is deleted and then re-added), the SWIMA-PC MUST still report each change event rather than just reporting the net effect of changes over the delay period. In other words, delayed fulfillment can decrease the number of attributes send by the SWIMA-PC, but it does not reduce the total number of change events reported.

SWIMA-PCs are not required to support delayed fulfillment of subscriptions. However, in the case that the SWIMA-PC does support delayed subscription fulfillment, it MUST be possible to configure the SWIMA-PC to disable delayed fulfillment. In other words, parties deploying SWIMA-PCs need to be allowed to disable delayed subscription fulfillment in their SWIMA-PCs. The manner in which such configuration occurs is left to the discretion of implementers, although implementers MUST protect the configuration procedure from unauthorized tampering. In other words, there needs to be some assurance that unauthorized individuals are not able to introduce long delays in subscription fulfillment.

3.9. Error Handling

In the case where the SWIMA-PC detects an error in a SWIMA Request attribute that it receives it MUST respond with a PA-TNC Error attribute with an error code appropriate to the nature of the error. (See Section 4.2.8 of PA-TNC [RFC5792] for more details about PA-TNC Error attributes and error codes as well as Section 5.16 in this specification for error codes specific to SWIMA Attributes.) In the case that an error is detected in a SWIMA Request the SWIMA-PC MUST NOT take any action requested by this SWIMA Request, even if partial completion of the request is possible. In other words, a SWIMA Request that contains an error is completely ignored by the SWIMA-PC (beyond sending a PA-TNC Error attribute, and possibly logging the error locally) rather than being partially executed.

In the case where the SWIMA-PC receives a valid SWIMA Request attribute but experiences an error during the process of responding to that attribute’s instructions where that error prevents the SWIMA-PC from properly or completely fulfilling that request, the SWIMA-PC MUST send a PA-TNC Error attribute with an error code appropriate to the nature of the error. In the case where a PA-TNC Error attribute is sent, the SWIMA-PC MUST NOT take any of the actions requested by the SWIMA Request attribute which led to the detected error. This is the case even if some actions could have been completed successfully, and might even require the SWIMA-PC to reverse some successful actions already taken before the error condition was detected. In other words, either all aspects of a SWIMA Request complete fully and successfully (in which case the SWIMA-PC sends a SWIMA Response attribute), or no aspects of the SWIMA Request occur (in which case the SWIMA-PC sends a PA-TNC Error attribute). In the case that a SWIMA-PC sends a PA-TNC Error attribute in response to a SWIMA Request then the SWIMA-PC MUST NOT also send any SWIMA Response attribute in response to the same SWIMA Request. For this reason, the sending of a SWIMA Response attribute MUST be the last action taken by a SWIMA-PC in response to a SWIMA Request to avoid the possibility of a processing error occurring after that SWIMA Response attribute is sent.

In the case that the SWIMA-PC detects an error that prevents it from properly or completely fulfilling its obligations under an active subscription, the SWIMA-PC MUST send a PA-TNC Error attribute of type SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR to the SWIMA-PV that established this subscription. This type of PA-TNC Error attribute identifies the specific subscription that cannot be adequately honored due to the error condition as well as an error "sub-type". The error sub-type is used to indicate the type of error condition the SWIMA-PC experienced that prevented it from honoring the given subscription. In the case that the error condition cannot be identified or does not align with any of the defined error codes, the SWIMA_ERROR error code SHOULD be used in the sub-type. In the case that a SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR is sent, the associated subscription MUST be treated as cancelled by both the SWIMA-PC and SWIMA-PV.

The SWIMA-PV MUST NOT send any PA-TNC Error attributes to SWIMA-PCs. In the case that a SWIMA-PV detects an error condition, it SHOULD log this error but does not inform any SWIMA-PC's of this event. Errors might include, but are not limited to, detection of malformed SWIMA Response attributes sent from a given SWIMA-PC, as well as detection of error conditions when the SWIMA-PV processes SWIMA Responses.

Both SWIMA-PCs and SWIMA-PVs SHOULD log errors so that administrators can trace the causes of errors. Log entries SHOULD include the type of the error, the time it was detected, and additional descriptive information to aid in understanding the nature and cause of the error. Logs are an important debugging tool and implementers are strongly advised to include comprehensive logging capabilities in their products.

4. Protocol

The software inventory protocol supports two different types of message exchanges which are described the following subsections, along with implementation requirements for supporting these exchanges.

4.1. Direct Response to a SWIMA Request

The first type of exchange is used to provide the SWIMA-PV with a software inventory or event collection from the queried endpoint.

```
+-------------+                      +--------------+
|  SWIMA-PC   |                      |   SWIMA-PV   |  Time
+-------------+                      +--------------+    |
      |                                     |            |
      |<-----------SWIMA Request------------|            |
      |                                     |            |
      |            SWIMA Response*          |            |
      |----------------or------------------>|            |
      |              PA-TNC Error           |            |
      |                                     |            V
```

*SWIMA Response is one of the following: Software Identifier Inventory, Software Identifier Events, Software Inventory, or Software Events.

Figure 4: SWIMA Attribute Exchange (Direct Response to SWIMA Request)

In this exchange, the SWIMA-PV indicates to the SWIMA-PC, via a SWIMA Request, the nature of the information it wishes to receive (inventory vs. events, full or targeted) and how it wishes the returned inventory to be expressed (with or without software inventory evidence records). The SWIMA-PC responds with the requested information using the appropriate attribute type. A single SWIMA Request MUST only lead to a single SWIMA Response or PA-TNC Error that is in direct response to that request.

4.2. Subscription-Based Response

The second type of exchange allows change event-based reporting based on a subscription. If there is an active subscription on the endpoint, the SWIMA-PC sends a SWIMA Response to the SWIMA-PV following a change event on the endpoint in fulfillment of that subscription. Such an exchange is shown in Figure 5.

```
            +-------------+               +--------------+
            |  SWIMA-PC   |               |   SWIMA-PV   |  Time
            +-------------+               +--------------+   |
                  |                              |            |
    <Change Event>|                              |            |
                  |------SWIMA Response(s)*------>|            |
                  |                              |            |
```

*SWIMA Response is one of the following: Software Identifier Inventory, Software Identifier Events, Software Inventory, or Software Events.

Figure 5: SWIMA Attribute Exchange (In Fulfillment of an Active Subscription)

Note that, unlike direct responses to a SWIMA Request, a single change event can precipitate multiple SWIMA Responses for a single subscription, but only if all but the last of those SWIMA Responses convey partial lists of event records. When providing multiple SWIMA Responses in this way, the initial responses contain partial lists of event records and the last of those SWIMA Responses conveys the remainder of the relevant event records, completing the delivery of all relevant events in response to the change event. A single Change Event MUST NOT otherwise be followed by multiple SWIMA Response or PA-TNC Error attributes in any combination.

4.3. Required Exchanges

All SWIMA-PVs and SWIMA-PCs MUST support both types of exchanges. In particular, SWIMA-PCs MUST be capable of pushing a SWIMA Response to a SWIMA-PV immediately upon detection of a change to the endpoint's Software Inventory Evidence Collection in fulfillment of established SWIMA-PV subscriptions, as described in Section 3.8.

5. Software Inventory Messages and Attributes

This section describes the format and semantics of the SWIMA protocol. This protocol uses the PA-TNC message header format [RFC5792].

5.1. PA Subtype (AKA PA-TNC Component Type)

The NEA PB-TNC interface provides a general message-batching protocol capable of carrying one or more PA-TNC messages between the Posture Broker Client and Posture Broker Server. When PB-TNC is carrying a PA-TNC message, the PB-TNC message headers contain a 32 bit identifier called the PA Subtype. The PA Subtype field indicates the type of component associated with all of the PA-TNC attributes carried by the PB-TNC message. The core set of PA Subtypes is defined in the PA-TNC specification. This specification defines a new "SWIMA Attributes" PA Subtype, which is registered in Section 10.1 of this document, which is used as a namespace for the collection of SWIMA Attributes defined in this document.

For more information on PB-TNC and PA-TNC messages and message headers, see the PB-TNC [RFC5793] and PA-TNC [RFC5792] specifications, respectively.

5.2. SWIMA Attribute Overview

Each PA-TNC attribute described in this specification is intended to be sent between the SWIMA-PC and SWIMA-PV, so will be carried in a PB-TNC message indicating a PA Subtype of "SWIMA Attributes". PB-TNC messages MUST always include the SWIMA Attributes Subtype defined in Section 5.1 when carrying SWIMA Attributes over PA-TNC. The attributes defined in this specification appear below along with a short summary of their purposes.

PA-TNC attribute types are identified in the PA-TNC Attribute Header via the Attribute Type Vendor ID and Attribute Type fields. Table 1 identifies the appropriate values for these fields for each attribute type used within the SWIMA protocol. All attributes have a PEN value of 0x000000. For the Integer value field, both the hexadecimal and decimal values are provided. Each attribute is described in greater detail in subsequent sections identified in the attribute description.

Attribute Name	Integer	Description
SWIMA Request	0x0000000D (13)	Request sent from a SWIMA-PV to a SWIMA-PC for the SWIMA-PC to provide a software inventory or event list. It might also establish a subscription. (Section 5.7)
Software Identifier Inventory	0x0000000E (14)	An inventory sent without software inventory evidence records sent from a SWIMA-PC. (Section 5.8)
Software Identifier Events	0x0000000F (15)	A collection of events impacting the endpoint's Software Inventory Evidence Collection, where events do not include software inventory

		evidence records. (Section 5.9)
Software Inventory	0x00000010 (16)	An inventory including software inventory evidence records sent from a SWIMA-PC. (Section 5.10)
Software Events	0x00000011 (17)	A collection of events impacting the endpoint’s Software Inventory Evidence Collection, where events include software inventory evidence records. (Section 5.11)
Subscription Status Request	0x00000012 (18)	A request for a list of a SWIMA-PV’s active subscription on a SWIMA-PC. (Section 5.12)
Subscription Status Response	0x00000013 (19)	A list of a SWIMA-PV’s active subscriptions on a SWIMA-PC. (Section 5.13)
Source Metadata Request	0x00000014 (20)	A request for information about a SWIMA-PC’s data sources. (Section 5.14)
Subscription Metadata Response	0x00000015 (21)	Descriptive metadata about a SWIMA-PC’s data sources. (Section 5.15)
PA-TNC Error	0x00000008 (8)	An error attribute as defined in the PA-TNC specification [RFC5792].

Table 1: SWIMA Attribute Enumeration

Because one of the Software Identifier Inventory, Software Identifier Events, Software Inventory, or Software Events attributes are expected to be sent to a SWIMA-PV in direct response to a SWIMA Request attribute or in fulfillment of an active subscription, those four attribute types are referred to collectively in this document as "SWIMA Response attributes".

All SWIMA-PVs MUST be capable of sending SWIMA Request attributes and be capable of receiving and processing all SWIMA Response attributes as well as PA-TNC Error attributes. All SWIMA-PCs MUST be capable of receiving and processing SWIMA Request attributes and be capable of sending all types of SWIMA Response attributes as well as PA-TNC Error attributes. In other words, both SWIMA-PVs and SWIMA-PCs are required to support their role in exchanges using any of the attribute types defined in this section. SWIMA-PVs MUST ignore any SWIMA Request attributes that they receive. SWIMA-PCs MUST ignore any SWIMA Response attributes or PA-TNC Error attributes that they receive.

5.3. Message Diagram Syntax

This specification defines the syntax of new PA-TNC messages and attributes using diagrams. Each diagram depicts the format and size of each field in bits. Implementations MUST send the bits in each diagram as they are shown from left to right for each 32-bit quantity traversing the diagram from top to bottom. Fields representing numeric values MUST be sent in network (big endian) byte order.

Descriptions of bit fields (e.g., flags) values refer to the position of the bit within the field. These bit positions are numbered from the most significant bit through the least significant bit. As such, an octet with only bit 0 set would have a value of 0x80 (1000 0000), an octet with only bit 1 set would have a value of 0x40 (0100 0000), and an octet with only bit 7 set would have a value of 0x01 (0000 0001).

5.4. SWIMA Attribute Enumeration

5.5. Normalization of Text Encoding

In order to ensure consistency of transmitted attributes some fields require normalization of their format. When this is necessary, this is indicated in the field’s description. In such cases, the field contents MUST be normalized to Network Unicode format as defined in RFC 5198 [RFC5198]. Network Unicode format defines a refinement of UTF-8 [RFC3629] that ensures a normalized expression of characters. SWIMA-PCs and SWIMA-PVs MUST NOT perform conversion and normalization on any field values except those specifically identified as requiring normalization in the following sections. Note, however, that some data models require additional normalization before source information is added to an Endpoint’s Inventory Evidence Collection as a record. The references from the Software Data Model IANA table (see Section 10.4) will note where this is necessary.

5.6. Request IDs

All SWIMA Request attributes MUST include a Request ID value. The Request ID field provides a value that identifies a given request relative to other requests between a SWIMA-PV and the receiving SWIMA-PC. Specifically, the SWIMA-PV assigns each SWIMA Request attribute a Request ID value that is intended to be unique within the lifetime of a given network connection ID as assigned by the SWIMA-PV's Posture Broker Server.

In the case that a SWIMA Request requests the establishment of a subscription and the receiving SWIMA-PC agrees to that subscription, the Request ID of that SWIMA Request (i.e., the establishing request of the subscription) becomes that subscription's Subscription ID. All attributes sent in fulfillment of this subscription include a flag indicating that the attribute fulfills a subscription and the subscription's Subscription ID. A SWIMA-PV MUST NOT reuse a Request ID value in communicating to a given SWIMA-PC while that Request ID is also serving as a Subscription ID for an active subscription with that SWIMA-PC. In the case where a SWIMA-PC receives a SWIMA Request from a given SWIMA-PV where that Request ID is also the Subscription ID of an active subscription with that SWIMA-PV, the SWIMA-PC MUST respond with a PA-TNC Error attribute with an error code of SWIMA_SUBSCRIPTION_ID_REUSE_ERROR. Note that this error does not cancel the indicated subscription.

Subscription Status Requests and Subscription Status Responses do not include Request IDs.

In the case where all possible Request ID values have been exhausted within the lifetime of a single network connection ID, the sender MAY reuse previously used Request IDs within the same network connection if the ID is not being used as a Subscription ID. In such a case of Request ID reuse, Request IDs SHOULD be reused in the order of their original use. In other words, a SWIMA-PC SHOULD NOT use a given Request ID value more than once within a persistent connection between a given Posture Broker Client-Posture Broker Server pair. In the case where reuse is necessary due to exhaustion of possible ID values, the SWIMA-PC SHOULD structure the reuse to maximize the time between original and subsequent use. The Request ID value is included in a SWIMA Response attribute directly responding to this SWIMA Request to indicate which SWIMA Request was received and caused the response. Request IDs can be randomly generated or sequential, as long as values are not repeated per the rules in this paragraph. SWIMA-PCs are not required to check for duplicate Request IDs.

5.7. SWIMA Request

A SWIMA-PV sends this attribute to a SWIMA-PC to request that the SWIMA-PC send software inventory information to the SWIMA-PV. A SWIMA-PC MUST NOT send this attribute.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Flags        |        Software Identifier Count              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Request ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Earliest EID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|    SUB-BLOCK (Repeated "Software Identifier Count" times)     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: SWIMA Request Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Identifier Length  | Software Identifier (var len) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: SWIMA Request Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Clear Subscriptions	If set (1), the SWIMA-PC MUST delete all subscriptions established by the requesting SWIMA-PV (barring any errors).
Flags: Bit 1 - Subscribe	If set (1), in addition to responding to the request as described, the SWIMA-PC MUST establish a subscription with parameters matching those in the request attribute (barring any errors).
Flags: Bit 2 - Result Type	If unset (0), the SWIMA-PC’s response MUST include software inventory evidence records and thus the response MUST be a Software Inventory, a Software Events, or a PA-TNC Error attribute. If set (1), the response MUST NOT include software inventory evidence records and thus the response MUST be a Software Identifier Inventory, a Software Identifier Events, or a PA-TNC Error attribute.
Flags: Bit	Reserved for future use. This field MUST be set

3-7 - Reserved	to zero on transmission and ignored upon reception.
Software Identifier Count	A 3-byte unsigned integer indicating the number of Software Identifiers that follow. If this value is non-zero, this is a targeted request, as described in Section 3.5. The Software Identifier Length and Software Identifier fields are repeated, in order, the number of times indicated in this field. In the case where Software Identifiers are present, the SWIMA-PC MUST only report software that corresponds to the identifiers the SWIMA-PV provided in this attribute (or with a PA-TNC Error attribute). This field value MAY be 0, in which case there are no instances of the Software Identifier Length and Software Identifier fields. In this case, the SWIMA-PV is indicating an interest in all software inventory evidence records on the endpoint (i.e., this is not a targeted request).
Request ID	A value that uniquely identifies this SWIMA Request from a particular SWIMA-PV.
Earliest EID	In the case where the SWIMA-PV is requesting software events, this field contains the EID value of the earliest event the SWIMA-PV wishes to have reported. (Note - the report will be inclusive of the event with this EID value.) In the case where the SWIMA-PV is requesting an inventory, then this field MUST be 0. (0x00000000) In the case where this field is non-zero, the SWIMA-PV is requesting events and the SWIMA-PC MUST respond using a Software Events, Software Identifier Events, or a PA-TNC Error attribute. In the case where this field is zero, the SWIMA-PV is requesting an inventory and the SWIMA-PC MUST respond using a Software Inventory, a Software Identifier Inventory, or a PA-TNC Error attribute.
Software Identifier Length	A 2-byte unsigned integer indicating the length in bytes of the Software Identifier field.
Software Identifier	A string containing the Software Identifier value from a software inventory evidence record. This field value MUST be normalized to Network Unicode

	format, as described in Section 5.5. This string MUST NOT be NULL terminated.

Table 2: SWIMA Request Attribute Fields

The SWIMA-PV sends the SWIMA Request attribute to a SWIMA-PC to request the indicated information. Note that between the Result Type flag and the Earliest EID field, the SWIMA-PC is constrained to a single possible SWIMA Response attribute type (or a PA-TNC Error attribute) in its response to the request.

The Subscribe and Clear Subscription flags are used to manage subscriptions for the requesting SWIMA-PV on the receiving SWIMA-PC. Specifically, an attribute with the Subscribe flag set seeks to establish a new subscription by the requesting SWIMA-PV to the given SWIMA-PC, while an attribute with the Clear Subscription flag seeks to delete all existing subscriptions by the requesting SWIMA-PV on the given SWIMA-PC. Note that, in the latter case, only the subscriptions associated with the Connection ID and the Posture Validator ID of the requester are deleted as described in Section 3.8.3. A newly established subscription has the parameters outlined in the Request attribute. Specifically, the Result Type flag indicates the type of result to send in fulfillment of the subscription, the value of the Earliest EID field indicates whether the fulfillment attributes list inventories or events, and the fields describing Software Identifiers (if present) indicate if and how a subscription is targeted. In the case that the SWIMA-PC is unable or unwilling to comply with the SWIMA-PV’s request to establish or clear subscriptions, the SWIMA-PC MUST respond with a PA-TNC Error attribute with the SWIMA_SUBSCRIPTION_DENIED_ERROR error code. If the SWIMA-PV requests that subscriptions be cleared but has no existing subscriptions, this is not an error.

An attribute requesting the establishment of a subscription is effectively doing double-duty, as it is a request for an immediate response from the SWIMA-PC in addition to setting up the subscription. Assuming the SWIMA-PC is willing to comply with the subscription it MUST send an appropriate response attribute to a request with the Subscribe flag set containing all requested information. The same is true of the Clear Subscription flag - assuming there is no error the SWIMA-PC MUST generate a response attribute without regard to the presence of this flag in addition to clearing its subscription list.

Both the Subscribe and Clear Subscription flags MAY be set in a single SWIMA Request attribute. In the case where this request is successful, the end result MUST be equivalent to the SWIMA-PC clearing its subscription list for the given SWIMA-PV first and then creating a new subscription in accordance with the request parameters. In other words, do not first create the new subscription and then clear all the subscriptions including the one that was just created. In the case that the requested actions are successfully completed, the SWIMA-PC MUST respond with a SWIMA Response attribute. The specific type of SWIMA Response attribute depends on the Result Type and Earliest EID fields, as described above. In the case where there is a failure that prevents some part this request from completing, the SWIMA-PC MUST NOT add a new subscription, MUST NOT clear the old subscriptions, and the SWIMA-PC MUST respond with a PA-TNC Error attribute. In other words, the SWIMA-PC MUST NOT partially succeed at implementing such a request; either all actions succeed, or none succeed.

The Earliest EID field is used to indicate if the SWIMA-PV is requesting an inventory or event list from the SWIMA-PC. A value of 0 (0x00000000) represents a request for inventory information. Otherwise, the SWIMA-PV is requesting event information. For Earliest EID values other than 0, the SWIMA-PC's response MUST respond with event records, as described in Section 3.7. Note that the request does not identify a particular EID Epoch, since responses can only include events in the SWIMA-PC's current EID Epoch.

The Software Identifier Count indicates the number of Software Identifiers in the attribute. This number might be any value between 0 and 16,777,216, inclusive. A single Software Identifier is represented by the following fields: Software Identifier Length and Software Identifier. These fields are repeated a number of times equal to the Software Identifier Count, which may be 0. The Software Identifier Length field indicates the number of bytes allocated to the Software Identifier field. The Software Identifier field contains a Software Identifier as describe in Section 3.4.1. The presence of one or more Software Identifiers is used by the SWIMA-PV to indicate a targeted request, which seeks only inventories of or events affecting software corresponding to the given identifiers. The SWIMA-PC MUST only report software that matched the Software Identifiers provided in the SWIMA-PVs SWIMA Request attribute.

5.8. Software Identifier Inventory

A SWIMA-PC sends this attribute to a SWIMA-PV to convey the inventory of the endpoint's Software Inventory Evidence Collection without the inclusion of software inventory evidence records. This list might represent a complete inventory or a targeted list of records, depending on the parameters in the SWIMA-PV's request. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC either sends this attribute in fulfillment of an existing subscription where the establishing request has a Result Type of 1 and the Earliest EID is zero, or in direct response to a SWIMA Request attribute where the Result Type is 1 and the Earliest EID is zero.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Flags        |         Software Identifier Count             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Request ID Copy / Subscription ID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          EID Epoch                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Last EID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|    SUB-BLOCK (Repeated "Software Identifier Count" times)     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Software Identifier Inventory Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Record Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |   Software Identifier Length  |Software Id (v)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Software Identifier Inventory Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bit 1-7	Reserved for future use. This field MUST be set

- Reserved	to zero on transmission and ignored upon reception.
Software Identifier Count	The number of Software Identifiers that follow. This field is an unsigned integer. The Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is an unsigned 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field is an unsigned 4-byte unsigned integer.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a software inventory evidence record.
Data Model Type PEN	A 3-byte unsigned integer containing the Private Enterprise Number (PEN) of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identification Number associated with the source from which this software installation inventory instance was reported.

Software Identifier Length	A 2-byte unsigned integer indicating the length in bytes of the SW ID field.
Software Identifier	A string containing the Software Identifier value from a software inventory evidence record. This field value MUST be normalized to Network Unicode format as described in Section 5.5. This string MUST NOT be NULL terminated.
Software Locator Length	A 2-byte unsigned integer indicating the length in bytes of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.5, and then encoded as a URI [RFC3986]. This string MUST NOT be NULL terminated.

Table 3: Software Identifier Inventory Attribute Fields

In the case that this attribute is sent in fulfillment of a subscription, the Subscription Fulfillment bit MUST be set (1). In the case that this attribute is sent in direct response to a SWIMA Request, the Subscription Fulfillment bit MUST be unset (0). Note that the SWIMA Response attribute sent in direct response to a SWIMA Request that establishes a subscription (i.e., a subscription’s establishing request) MUST be treated as a direct response to that SWIMA Request (and thus the Subscription Fulfillment bit is unset). SWIMA Response attributes are only treated as being in fulfillment of a subscription (i.e., Subscription Fulfillment bit set) if they are sent following a change event, as shown in Figure 3.

The Software Identifier Count field indicates the number of Software Identifiers present in this inventory. Each Software Identifier is represented by the following set of fields: Record Identifier, Data Model Type, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator. These fields will appear once for each reported record.

When responding directly to a SWIMA Request attribute, the Request ID Copy / Subscription ID field MUST contain an exact copy of the Request ID field from that SWIMA Request. When this attribute is sent in fulfillment of an existing subscription on this Posture Collector, then this field MUST contain the Subscription ID of the fulfilled subscription.

The EID Epoch field indicates the EID Epoch of the Last EID value. The Last EID field MUST contain the EID of the last recorded change event (see Section 3.7 for more about EIDs and recorded events) at the time this inventory was collected. In the case where there are no recorded change events at the time that this inventory was collected, the Last EID field MUST contain 0. These fields can be interpreted to indicate that the provided inventory reflects the state of the endpoint after all changes up to and including this last event have been accounted for.

The Data Model Type PEN and Data Model Type fields are used to identify the data model associated with the given software record. These fields are discussed more in Section 3.4.2.

The Source Identification Number field is used to identify the source that provided the given record, as described in Section 3.1.

5.9. Software Identifier Events

A SWIMA-PC sends this attribute to a SWIMA-PV to convey events where the affected records are reported without software inventory evidence records. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC either sends this attribute in fulfillment of an existing subscription where the establishing request has a Result Type is 1 and the Earliest EID is non-zero, or in direct response to a SWIMA Request attribute where the Result Type is 1 and the Earliest EID is non-zero.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Flags        |               Event Count                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Request ID Copy / Subscription ID                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       EID Epoch                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Last EID                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Last Consulted EID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            SUB-BLOCK (Repeated "Event Count" times)           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Software Identifier Events Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              EID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                                                             -+
|                           Timestamp                           |
+-                                                             -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Record Identifier                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |  Action       |   Software Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Software Identifier (variable length)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Software Identifier Events Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bit 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Event Count	The number of events that are reported in this attribute. This field is a 3-byte unsigned integer. The EID, Timestamp, Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Action, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator fields are

	repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is an unsigned 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field contains the EID of the SWIMA-PC's last recorded change event (which might or might not be included as an event record in this attribute).
Last Consulted EID	The EID of the last event record that was consulted when generating the event record list included in this attribute. This is different from the Last EID field value if and only if this attribute is conveying a partial list of event records. See Section 3.7.5 for more on partial list of event records.
EID	The EID of the event in this event record.
Timestamp	The timestamp associated with the event in this event record. This timestamp is the SWIMA-PC's best understanding of when the given event occurred. Note that this timestamp might be an estimate. The Timestamp date and time MUST be represented as an RFC 3339 [RFC3339] compliant ASCII string in Coordinated Universal Time (UTC) time with the additional restrictions that the 'T' delimiter and the 'Z' suffix MUST be capitalized and fractional seconds (time-secfrac) MUST NOT be included. This field conforms to the date-time ABNF production from section 5.6 of RFC 3339 with the above restrictions. Leap seconds are permitted and

	SWIMA-PVs MUST support them. The Timestamp string MUST NOT be NULL terminated or padded in any way. The length of this field is always 20 octets.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a software inventory evidence record.
Data Model Type PEN	A 3-byte unsigned integer containing the Private Enterprise Number (PEN) of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identification Number associated with the source from which this software installation inventory instance was reported.
Action	The type of event that is recorded in this event record. Possible values are: 1 = CREATION - the addition of a record to the endpoint's Software Inventory Evidence Collection; 2 = DELETION - the removal of a record from the endpoint's Software Inventory Evidence Collection; 3 = ALTERATION - There was an alteration to a record within the endpoint's Software Inventory Evidence Collection. All other values are reserved for future use and MUST NOT be used when sending attributes. In the case where a SWIMA-PV receives an event record that uses an action value other than the ones defined here, it MUST ignore that event record but SHOULD process other event records in this attribute as normal.
Software Identifier Length	A 2-byte unsigned integer indicating the length in bytes of the SW ID field.
Software Identifier	A string containing the Software Identifier value from a software inventory evidence record. This field value MUST first be normalized to Network Unicode format as described in Section 5.5. This string MUST NOT be NULL terminated.

Software Locator Length	A 2-byte unsigned integer indicating the length in bytes of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.5, and then encoded as a URI [RFC3986]. This string MUST NOT be NULL terminated.

Table 4: Software Identifier Events Attribute Fields

The first few fields in the Software Identifier Events attribute mirror those in the Software Identifier Inventory attribute. The primary difference is that, instead of conveying an inventory, the attribute conveys zero or more event records, consisting of the EID, timestamp, Record Identifier, action type, data model type, Software Identifier, and Software Locator of the affected software inventory evidence record.

With regard to the Timestamp field, it is important to note that clock skew between the SWIMA-PC and SWIMA-PV as well as between different SWIMA-PCs within an enterprise might make correlation of timestamp values difficult. This specification does not attempt to resolve clock skew issues, although other mechanisms outside of this specification do exist to reduce the impact of clock skew and make the timestamp more useful for such correlation. Instead, SWIMA uses the Timestamp value primarily as a means to indicate the amount of time between two events on a single endpoint. For example, by taking the difference of the times for when a record was removed and then subsequently re-added, one can get an indication as to how long the system was without the given record (and, thus without the associated software). Since this will involve comparison of timestamp values all originating on the same system, clock skew between the SWIMA-PC and SWIMA-PV is not an issue. However, if the SWIMA-PC’s clock was adjusted between two recorded events, it is possible for such a calculation to lead to incorrect understandings of the temporal distance between events. Users of this field need to be aware of the possibility for such occurrences. In the case where the Timestamp values of two events appear to contradict the EID ordering of those events (i.e., the later EID has an earlier timestamp) the recipient MUST treat the EID ordering as correct.

All events recorded in a Software Identifier Events Attribute are required to be part of the same EID Epoch. Specifically, all reported events MUST have an EID from the same EID Epoch as each other, and which is the same as the EID Epoch of the Last EID and

Last Consulted EID values. The SWIMA-PC MUST NOT report events with EIDs from different EID Epochs.

The Last Consulted EID field contains the EID of the last event record considered for inclusion in this attribute. If this attribute contains a partial event set (as described in Section 3.7.5) this field value will be less than the Last EID value; if this attribute contains a complete event set, the Last EID and Last Consulted EID values are identical.

If multiple events are sent in a Software Identifier Events attribute, the order in which they appear within the attribute is not significant. The EIDs associated with them are used for ordering the indicated events appropriately. Also note that a single software record might be reported multiple times in an attribute, such as if multiple events involving the associated record were being reported.

5.10. Software Inventory

A SWIMA-PC sends this attribute to a SWIMA-PV to convey a list of inventory records. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC either sends this attribute in fulfillment of an existing subscription where the establishing request had a Result Type of 0 and the Earliest EID is zero, or in direct response to a SWIMA Request attribute where the Result Type is 0 and the Earliest EID is zero.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Flags         |               Record Count                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Request ID Copy / Subscription ID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           EID Epoch                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Last EID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|          SUB-BLOCK (Repeated "Record Count" times)            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Software Inventory Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Record Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |   Software Identifier Length  |Software Id (v)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Record Length                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Record (variable length)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: SWIMA Inventory Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bit 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Record Count	The number of records that follow. This field is a 3-byte unsigned integer. The Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Software Identifier Length, Software Identifier, Software Locator Length, Software Locator, Record Length, and Record fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0 in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this

	field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is an unsigned 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field is an unsigned 4-byte unsigned integer.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a software inventory evidence record.
Data Model Type PEN	A 3-byte unsigned integer containing the Private Enterprise Number (PEN) of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identification Number associated with the source from which this software installation inventory instance was reported.
Software Identifier Length	A 2-byte unsigned integer indicating the length in bytes of the SW ID field.
Software Identifier	A string containing the Software Identifier value from a software inventory evidence record. This field value MUST first be normalized to Network Unicode format as described in Section 5.5. This string MUST NOT be NULL terminated.
Software Locator Length	A 2-byte unsigned integer indicating the length in bytes of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.5, and then encoded as a URI [RFC3986]. This string MUST NOT be NULL terminated.
Record Length	This is a 4-byte unsigned integer indicating the

	length of the Record field in bytes.
Record	A software inventory evidence record as a string. The record MUST be converted and normalized to Network Unicode format as described in Section 5.5. This string MUST NOT be NULL terminated.

Table 5: Software Inventory Attribute Fields

The Software Inventory attribute contains some number of software inventory evidence records along with the core response attribute fields. Given that the size of records can vary considerably, the length of this attribute is highly variable and, if transmitting a complete inventory, can be extremely large. Enterprises might wish to constrain the use of Software Inventory attributes to targeted requests to avoid over-burdening the network unnecessarily.

When copying a software inventory evidence record into the Record field, the record MUST be converted and normalized to use Network Unicode format prior to its inclusion in the record field.

5.11. Software Events

A SWIMA-PC sends this attribute to a SWIMA-PV to convey a list of events that include software inventory evidence records. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC either sends this attribute in fulfillment of an existing subscription where the establishing request has a Result Type of 0 and the Earliest EID is non-zero, or in direct response to a SWIMA Request attribute where the Result Type is 0 and the Earliest EID is non-zero.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flags       |                 Event Count                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Request ID Copy / Subscription ID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           EID Epoch                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Last EID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Last Consulted EID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|             SUB-BLOCK (Repeated "Event Count" times)          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Software Events Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              EID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                                                             -+
|                          Timestamp                            |
+-                                                             -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Record Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |  Action       |   Software Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Software Identifier (variable length)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Record Length                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Record (variable Length)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: SWIMA Events Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bit 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Event Count	The number of events being reported in this attribute. This field is a 3-byte unsigned integer. The EID, Timestamp, Record Identifier,

	Data Model Type PEN, Data Model Type, Source Identification Number, Action, Software Identifier Length, Software Identifier, Software Locator Length, Software Locator, Record Length, and Record fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is an unsigned 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field contains the EID of the SWIMA-PC's last recorded change event (which might or might not be included as an event record in this attribute).
Last Consulted EID	The EID of the last event record that was consulted when generating the event record list included in this attribute. This is different from the Last EID field value if and only if this attribute is conveying a partial list of event records. See Section 3.7.5 for more on partial list of event records.
EID	The EID of the event in this event record.
Timestamp	The timestamp associated with the event in this event record. This timestamp is the SWIMA-PC's best understanding of when the given event occurred. Note that this timestamp might be an estimate. The Timestamp date and time MUST be represented as an RFC 3339 [RFC3339] compliant ASCII string in Coordinated Universal Time (UTC) time with the additional restrictions that the 'T' delimiter and the 'Z' suffix MUST be capitalized and fractional seconds (time-

	secfrac) MUST NOT be included. This field conforms to the date-time ABNF production from section 5.6 of RFC 3339 with the above restrictions. Leap seconds are permitted and SWIMA-PVs MUST support them. The Timestamp string MUST NOT be NULL terminated or padded in any way. The length of this field is always 20 octets.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a software inventory evidence record.
Data Model Type PEN	A 3-byte unsigned integer containing the Private Enterprise Number (PEN) of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identification Number associated with the source from which this software installation inventory instance was reported.
Action	The type of event that is recorded in this event record. Possible values are: 1 = CREATION - the addition of a record to the endpoint's Software Inventory Evidence Collection; 2 = DELETION - the removal of a record from the endpoint's Software Inventory Evidence Collection; 3 = ALTERATION - There was an alteration to a record within the endpoint's Software Inventory Evidence Collection. All other values are reserved for future use and MUST NOT be used when sending attributes. In the case where a SWIMA-PV receives an event record that uses an action value other than the ones defined here, it MUST ignore that event record but SHOULD process other event records in this attribute as normal.
Software Identifier Length	A 2-byte unsigned integer indicating the length in bytes of the Software Identifier field.
Software	A string containing the Software Identifier

Identifier	value from a software inventory evidence record. This field value MUST first be normalized to Network Unicode format as described in Section 5.5. This string MUST NOT be NULL terminated.
Software Locator Length	A 2-byte unsigned integer indicating the length in bytes of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.5, and then encoded as a URI [RFC3986]. This string MUST NOT be NULL terminated.
Record Length	This is a 4-byte unsigned integer indicating the length of the Record field in bytes.
Record	A software inventory evidence record as a string. The record MUST be converted and normalized to Network Unicode format as described in Section 5.5. This string MUST NOT be NULL terminated.

Table 6: Software Events Attribute Fields

The fields of this attribute are used in the same way as the corresponding fields of the previous attributes. As with the Software Inventory attribute, a Software Events attribute can be quite large if many events have occurred following the event indicated by a request's Earliest EID. As such, it is recommended that the SWIMA Request attributes only request full records be sent (Result Type set to 0) in a targeted request, thus constraining the response just to records that match a given set of Software Identifiers.

As with the Software Identifier Events Attribute, this attribute MUST only contain event records with EIDs coming from the current EID Epoch of the SWIMA-PC.

As with the Software Inventory Attribute, the SWIMA-PC MUST perform conversion and normalization of the record.

5.12. Subscription Status Request

A SWIMA-PV sends this attribute to a SWIMA-PC to request a list of active subscriptions for which the requesting SWIMA-PV is the subscriber. A SWIMA-PC MUST NOT send this attribute.

This attribute has no fields.

A SWIMA-PC MUST respond to this attribute by sending a Subscription Status Response attribute (or a PA-TNC Error attribute if it is unable to correctly provide a response).

5.13. Subscription Status Response

A SWIMA-PC sends this attribute to a SWIMA-PV to report the list of active subscriptions for which the receiving SWIMA-PV is the subscriber. A SWIMA-PV MUST NOT send this attribute.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Flags  |            Subscription Record Count          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|    SUB-BLOCK (Repeated "Subscription Record Count" times)     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Subscription Status Response Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Flags         |          Software Identifier Count            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Request ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Earliest EID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|  SUB-SUB-BLOCK (Repeated "Software Identifier Count" times)   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: SWIMA Status Response Attribute SUB-BLOCK

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Software Identifier Length   | Software Identifier (var len) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: SWIMA Status Response Attribute SUB-SUB-BLOCK

Field	Description
Status Flags: Bit 0-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Subscription Record Count	The number of subscription records that follow. This field is a 3-byte unsigned integer. The Flags, Software Identifier Count, Request ID, and Earliest EID fields, and 0 or more instances of Software Identifier Length and Software Identifier, are repeated, in order, the number of times indicated in this field. (The Software Identifier Length and Software Identifier fields within each of these sets of fields are repeated a number of times equal to the preceding Software Identifier Count value.) The Subscription Record Count field value MAY be 0 in which case there are no instances of these fields.
Flags, Software Identifier Count, Request ID, Earliest EID, Software Identifier Length, and Software Identifier	For each active subscription, these fields contain an exact copy of the fields with the same name as provided in the subscription’s establishing request.

Table 7: Subscription Status Response Fields

A Subscription Status Response contains zero or more subscription records. Specifically, it MUST contain one subscription record for each active subscription associated with the party that sent the Subscription Status Request to which this attribute is a response. As described in Section 3.8.2, the SWIMA-PC MUST use the requester's Connection ID and its Posture Validator ID to determine which subscriptions are associated with the requester.

A SWIMA-PC MUST send a Subscription Status Response attribute in response to a Subscription Status Request attribute. The only exception to this is if the SWIMA-PC experiences an error condition that prevents it from correctly populating the Subscription Status Response attribute, in which case it MUST respond with a PA-TNC Error attribute appropriate to the type of error experienced. If there are no active subscriptions associated with the requesting party, the Subscription Status Response attribute will consist of its Status Flags field, a Subscription Record Count field with a value of 0, and no additional fields.

Each subscription record included in a Subscription Status Response attribute duplicates the fields of a SWIMA Request attribute that was the establishing request of a subscription. Note that the Request ID field in the record captures the Subscription ID associated with the given subscription record (since the Subscription ID is the same as the Request ID of the establishing request). Note also that if the establishing request is targeted, then its Record Count field will be non-zero and, within that subscription record, the Software Identifier Length and Software Identifier fields are repeated, in order, the number of times indicated in the Record Count field. As such, each subscription record can be different sizes. If the establishing request is not targeted (Record Count field is 0), the subscription record has no Software Identifier Length or Software Identifier fields.

When a SWIMA-PV compares the information received in a Subscription Status Response to its own records of active subscriptions it should be aware that the SWIMA-PC might be unable to distinguish this SWIMA-PV from other SWIMA-PVs on the same NEA Server. As a result, it is possible that the SWIMA-PC will report more subscription records than the SWIMA-PV recognizes. For this reason, SWIMA-PVs SHOULD NOT automatically assume that extra subscriptions reported in a Subscription Status Response indicate a problem.

5.14. Source Metadata Request

A SWIMA-PV sends this attribute to a SWIMA-PC to request metadata about sources that the SWIMA-PC is using to collect software inventory information. A SWIMA-PC MUST NOT send this attribute.

This attribute has no fields.

A SWIMA-PC MUST respond to this attribute by sending a Sources Metadata Response attribute (or a PA-TNC Error attribute if it is unable to correctly provide a response).

5.15. Source Metadata Response

A SWIMA-PC sends this attribute to a SWIMA-PV to provide descriptive metadata about the sources of software inventory information used by the SWIMA-PC. A SWIMA-PV MUST NOT send this attribute.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Reserved             | Source Count  |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                |
|                                                                |
|           SUB-BLOCK (Repeated "Source Count" times             |
|                                                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Metadata Response Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source ID     |       Metadata Length         | Metadata (var)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Metadata Response Attribute SUB-BLOCK

Field	Description
Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Source Count	The number of source records that follow. The Source ID, Metadata Size, and Metadata fields are repeated, in order, the number of times indicated by this field. This field MAY be 0, in which case no fields follow (but this would only be done to indicate that the SWIMA-PC has no active sources, which would not be a usual situation).
Source ID	The Source ID number associated with the described source for any communications with the recipient SWIMA-PV.
Metadata Length	A 2-byte unsigned integer indicating the length in bytes of the Metadata field.
Metadata	A string containing descriptive metadata about the indicated data source. This string MUST NOT be NULL terminated.

Source Metadata Response Fields

A Source Metadata Response attribute contains 0 or more records, each describing one of the data sources the SWIMA-PC uses to collect software inventory information. It SHOULD contain one metadata record for each source that the SWIMA-PC uses. (There might be reasons not to inform certain SWIMA-PVs of the presence of certain data sources.) The attribute MUST contain a metadata record for each source that has been identified in inventory or event messages to the given SWIMA-PV.

A SWIMA-PC MUST send a Source Metadata Response attribute in response to a Source Metadata Request attribute. The only exception to this is if the SWIMA-PC experiences an error condition that prevents it from correctly populating the Source Metadata Response attribute, in which case it MUST respond with a PA-TNC Error attribute appropriate to the type of error experienced.

The Source Count field indicates how many source metadata records are included in the attribute. Each included record consists of a Source Identifier, Metadata Length, and Metadata field.

The Source Identifier corresponds to the Source Identifier field in inventory and event messages. In the case where the Source Identifier value in a Source Metadata Response attribute matches a Source Identifier associated with an event or inventory record and both the Source Metadata Response and the inventory/event record were sent to the same SWIMA-PV, the source described in the Metadata field MUST be the same source that provided the event or inventory record associated with the Source Identifier. Recall that a SWIMA-PC MAY use different Source Identifier associations with different SWIMA-PVs. As such, the association between a Source Identifier and the conveyed metadata is also only meaningful for communications between the sending SWIMA-PC and receiving SWIMA-PV. When sending to a given SWIMA-PV, the SWIMA-PC MUST use the recipient SWIMA-PVs Source Identifier associations.

The Metadata Length indicates the length, in bytes, if the Metadata field. The Metadata field contains information about the indicated data source. This specification does not dictate a format for the contents of the Metadata field. This field MAY include machine-readable information. For broadest utility, the Metadata field SHOULD include human-readable, descriptive information about the data source.

5.16. PA-TNC Error as Used by SWIMA

The PA-TNC Error attribute is defined in the PA-TNC specification [RFC5792], and its use here conforms to that specification. A PA-TNC Error can be sent due to any error in the PA-TNC exchange and might also be sent in response to error conditions specific to the SWIMA exchange. The latter case utilizes error codes defined below.

A PA-TNC Error MUST be sent by a SWIMA-PC in response to a SWIMA Request in the case where the SWIMA-PC encounters a fatal error (i.e., an error that prevents further processing of an exchange) relating to the attribute exchange. A SWIMA-PV MUST NOT send this attribute. In the case where the SWIMA-PV experiences a fatal error, it MUST handle the error without sending a PA-TNC Error attribute. The SWIMA-PV MAY take other actions in response to the error, such as logging the cause of the error, or even taking actions to isolate the endpoint.

A PA-TNC Error attribute is sent instead of a SWIMA Response attribute due to factors that prevent the reliable creation of a SWIMA Response. As such, a SWIMA-PC MUST NOT send both a PA-TNC Error attribute and a SWIMA Response attribute in response to a single SWIMA Request attribute.

Table 8 lists the Error Code values for the PA-TNC Error attribute specific to the SWIMA exchange. Error codes are shown in both hexadecimal and decimal format. In all of these cases, the Error Code Vendor ID field MUST be set to 0x000000, corresponding to the IETF SMI Private Enterprise Number. The Error Information structures for each error type are described in the following subsections.

Note that a message with a SWIMA attribute might also result in an error condition covered by the Standard PA-TNC Error Codes defined in PA-TNC. For example, a SWIMA Attribute might have an invalid parameter, leading to an error code of "Invalid Parameter". In this case, the SWIMA-PC MUST use the appropriate PA-TNC Error Code value as defined in Section 4.2.8 of PA-TNC specification.

Error Code Value	Description
0x00000004 (4)	SWIMA_ERROR. This indicates a fatal error (i.e., an error that precludes the creation of a suitable response attribute) other than the errors described below but still specific to the processing of SWIMA Attributes. The Description field SHOULD contain additional diagnostic information.
0x00000005 (5)	SWIMA_SUBSCRIPTION_DENIED_ERROR. This indicates that the SWIMA-PC denied the SWIMA-PV's request to establish a subscription. The Description field SHOULD contain additional diagnostic information.
0x00000006 (6)	SWIMA_RESPONSE_TOO_LARGE_ERROR. This indicates that the SWIMA-PC's response to the SWIMA-PV's request was too large to be serviced. The error information structure indicates the largest possible size of a response supported by the SWIMA-PC (see Section 5.16.2). The Description field SHOULD contain additional diagnostic information.
0x00000007 (7)	SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR. This indicates that the SWIMA-PC experienced an error fulfilling a given subscription. The error information includes the Subscription ID of the relevant subscription, as well as a sub-error that describes the nature of the error the SWIMA-PC experienced. The SWIMA-PC and SWIMA-PV MUST treat the identified subscription as cancelled.
0x00000008 (8)	SWIMA_SUBSCRIPTION_ID_REUSE_ERROR. This indicates that the SWIMA-PC received a SWIMA Request from a given SWIMA-PV where the Request ID of that SWIMA Request is currently used as the Subscription ID of an active subscription with that SWIMA-PV. This error does not cancel the identified subscription.

Table 8: PA-TNC Error Codes for SWIMA

The following subsections describe the structures present in the Error Information fields. Note that all error structures include a variable-length field, but do not include any fields indicating the length of those fields. Such a field is unnecessary because all other fields in the PA-TNC Error attribute are fixed-length, and thus the length of the variable-length field can be found by subtracting the size of these fixed-length fields from the PA-TNC Attribute Length field in the PA-TNC Attribute Header.

5.16.1. SWIMA_ERROR, SWIMA_SUBSCRIPTION_DENIED_ERROR and SWIMA_SUBSCRIPTION_ID_REUSE_ERROR Information

The SWIMA_ERROR error code indicates that the sender (the SWIMA-PC) has encountered an error related to the processing of a SWIMA Request attribute but which is not covered by more specific SWIMA error codes. The SWIMA_SUBSCRIPTION_DENIED_ERROR is used when the SWIMA-PV requests to establish a subscription or clear all subscriptions from the given SWIMA-PV, but the SWIMA-PC is unable or unwilling to comply with this request. The SWIMA_SUBSCRIPTION_ID_REUSE_ERROR is used when the SWIMA-PC receives a SWIMA Request whose Request ID duplicates a Subscription ID of an active subscription with the request’s sender. All of these error codes use the following error information structure.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Copy of Request ID / Subscription ID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Description (variable length)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: SWIMA Error, Subscription Error, and Subscription ID Reuse Information

Field	Description
Copy of Request ID / Subscription ID	In the case that this error condition is generated in direct response to a SWIMA Request attribute, this field MUST contain an exact copy of the Request ID field in the SWIMA Request attribute that caused this error. In the case that the attribute in question is generated in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription for which the attribute was generated. (This is only possible if the error code is SWIMA_ERROR as the other errors are not generated by subscription fulfillment.) Note that, in this case, the indicated error appears as a sub-error for a SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR, as described in Section 5.16.3.
Description	A UTF-8 [RFC3629] string describing the condition that caused this error. This field MAY be 0-length. However, senders SHOULD include some description in all PA-TNC Error attributes of these types. This field MUST NOT be NULL terminated.

Table 9: SWIMA Error, Subscription Error, and Subscription ID Reuse Information Fields

This error information structure is used with SWIMA_ERROR, SWIMA_SUBSCRIPTION_DENIED_ERROR, and SWIMA_SUBSCRIPTION_ID_REUSE_ERROR status codes to identify the SWIMA Request attribute that precipitated the error condition and to describe the error. The Description field contains text describing the error. The SWIMA-PC MAY encode machine-interpretable information in this field, but SHOULD also include a human-readable description of the error, since the receiving SWIMA-PV might not recognize the SWIMA-PC’s encoded information.

5.16.2. SWIMA_RESPONSE_TOO_LARGE_ERROR Information

The SWIMA_RESPONSE_TOO_LARGE_ERROR error code indicates that a response generated by a SWIMA-PC in response to a SWIMA-PV’s SWIMA Request attribute was too large to send.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Copy of Request ID / Subscription I              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Maximum Allowed Size                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Description (variable length)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: SWIMA Response Too Large Error Information

Field	Description
Copy of Request ID / Subscription ID	In the case that the attribute in question is generated in direct response to a SWIMA Request, this field MUST contain an exact copy of the Request ID field in the SWIMA Request attribute that caused this error. In the case that the attribute in question is generated in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription for which the attribute was generated. Note that, in the latter case, the SWIMA_RESPONSE_TOO_LARGE_ERROR appears as a sub-error for a SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR, as described in Section 5.16.3.
Maximum Allowed Size	This field MUST contain an unsigned integer indicating the largest permissible size, in bytes, of SWIMA Attribute that the SWIMA-PC is currently willing to send in response to a SWIMA Request attribute.
Description	A UTF-8 [RFC3629] string describing the condition that caused this error. This field MAY be 0-length. However, senders SHOULD include some description in all PA-TNC Error attributes of these types. This field MUST NOT be NULL terminated.

Table 10: SWIMA Response Too Large Error Information Fields

This error structure is used with the SWIMA_RESPONSE_TOO_LARGE_ERROR status code to identify the SWIMA Request attribute that precipitated the error condition and to describe the error. The Maximum Allowed Size field indicates the largest attribute the SWIMA-PC is willing to send in response to a SWIMA Request under the current circumstances. Note that under other circumstances, the SWIMA-PC might be willing to return larger or smaller responses than indicated (such as if the endpoint connects to the NEA Server using a different network protocol). The other fields in this error information structure have the same meanings as corresponding fields in the SWIMA_ERROR and SWIMA_SUBSCRIPTION_DENIED_ERROR information structure.

5.16.3. SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR Information

The SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR error code indicates that the SWIMA-PC encountered an error while fulfilling a subscription. The bytes after the first 4 octets duplicate a PA-TNC Error attribute (as described in Section 4.2.8 of PA-TNC) that is used to identify the nature of the encountered error.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Subscription ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |           Sub Error Code Vendor ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Sub Error Code                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Sub Error Information (variable Length)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: SWIMA Subscription Fulfillment Error Information

Field	Description
Subscription ID	This field MUST contain the Subscription ID of the subscription whose fulfillment caused this error.
Reserved	This field MUST contain the value of the Reserved field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.
Sub Error Code Vendor ID	This field MUST contain the value of the Error Code Vendor ID field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.
Sub Error Code	This field MUST contain the value of the Error Code field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.
Sub Error Information	This field MUST contain the value of the Error Information field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.

Table 11: SWIMA Subscription Fulfillment Error Information Fields

This error structure is used with the SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR status code. The first 4 octets of this error structure contain the Subscription ID of the subscription that was being fulfilled when the error occurred. The remaining fields of this error structure duplicate the fields of a PA-TNC Error attribute, referred to as the "sub-error". The error code of the sub-error corresponds to the type of error that the SWIMA-PC encountered while fulfilling the given subscription. The sub-error MUST NOT have an error code of SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR.

The SWIMA-PC sending a PA-TNC Error attribute with this error code, and the SWIMA-PV receiving it, MUST treat the subscription identified by the Subscription ID field as cancelled. All other subscriptions are unaffected.

6. Supported Data Models

SWIMA supports an extensible list of data models for representing and transmitting software inventory information. This list of data models appears in the Software Data Model IANA registry (see Section 10.4). This document provides guidance for an initial set of data models. Other documents might provide guidance on the use of new data models by SWIMA, and will be referenced by extensions to the Software Data Model IANA registry.

6.1. ISO 2015 SWID Tags using XML

The International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC) published the specification governing SWID tag construction and use in 2009 with a revised version published in 2015. [SWID15] Since that time, a growing number of vendors have integrated SWID tags into their software products. Doing so significantly simplifies the task of identifying these pieces of software: instead of relying on discovery processes that look for clues as to software presence, such as the presence of particular files or registry keys. A readily available list of SWID tags provides simple and immediate evidence as to the presence of the given piece of software.

SWIMA has no reliance on the presence or management of SWID tags on an endpoint as described in the ISO specification. However, the data model for describing software that is presented in the ISO specification provides a robust and comprehensive way of describing software and is adopted here as a means of representing and transmitting software information. It should be emphasized, the use of the ISO SWID tag data model makes no assumption as to whether the source of the recorded information was, in fact, an ISO SWID tag harvested from the endpoint or whether the information was created using some other source and normalized to the SWID format.

6.1.1. Guidance on Normalizing Source Data to ISO 2015 SWID Tags using XML

Any record associated with this Software Data Model Type MUST conform to the ISO/IEC 19770-2-2015 [SWID15] specification.

If generating a new ISO 2015 SWID tag, the software generating the tag MUST use a Tag Creator RegID that is associated with the generating software, unless this is impossible, in which case it MUST use the "http://invalid.unavailable" Tag Creator RegID value. (This conforms to conventions for an unknown Tag Creator in the ISO 2015 SWID tag specification.) Do not use a RegID associated with any other party. In particular, it is incorrect to use a Tag Creator

RegID associated with the software being described by the tag, the enterprise that is using the software, or any other entity except that of the party that built the tool that is generating the SWID tag. This reflects the requirement that the Tag Creator RegID identify the party that created the tag. Moreover, any generated tags SHOULD conform with guidance for tag creators provided NIST IR 8060 [NIST8060], which provides additional recommendations to increase interoperable use of SWID tags.

6.1.2. Guidance on Creation of Software Identifiers from ISO 2015 SWID Tags

A Software Identifier generated from an ISO 2015 SWID tag is expressed as a concatenation of the value of the Tag Creator RegID field and the Unique ID field. Specifically, it MUST be of the form: TAG_CREATOR_REGID "_" "_" UNIQUE_ID. Specifically, the Software Identifier consists of, the Tag Creator RegID and the Unique ID from the tag connected with a double underscore (_), without any other connecting character or whitespace.

6.2. ISO 2009 SWID Tags using XML

As noted above, ISO's SWID tag specification provides a useful data model for representation of software information. As of the writing of this specification, while the 2015 specification is considered more comprehensive and addresses some issues with the 2009 specification, 2009-format SWID tags remain far more common in deployments. For this reason, ISO 2009 SWID tags are included in the Software Data Model IANA table.

6.2.1. Guidance on Normalizing Source Data to ISO 2009 SWID Tags using XML

Any record associated with this Software Data Model Type MUST conform to the ISO/IEC 19770-2-2009 [SWID09] specification. Any such tag SHOULD use a UTF-8 encoding [RFC3629], but MUST NOT alter the existing encoding if doing so would invalidate digital signatures included in the tag.

If generating a new ISO 2009 SWID tag, the software generating the tag MUST use a Tag Creator RegID that is associated with the generating software unless this is impossible, in which case it MUST use "unknown", which indicates the Tag Creator is unknown. (This conforms to conventions for an unknown Tag Creator in the ISO 2009 SWID specification.) Do not use a RegID associated with any other party. In particular, it is incorrect to use a Tag Creator RegID associated with the software being described by the tag, the enterprise that is using the software, or any other entity except that of the party that built the tool that is generating the SWID tag. This reflects the requirement that the Tag Creator RegID identify the party that created the tag.

6.2.2. Guidance on Creation of Software Identifiers from ISO 2009 SWID Tags

A Software Identifier generated from an ISO 2009 SWID tag is expressed as a concatenation of the value of the Tag Creator RegID field and the Unique ID field. Specifically, it MUST be of the form: TAG_CREATOR_REGID "_" "_" UNIQUE_ID. Specifically, the Software Identifier consists of, the Tag Creator RegID and the Unique ID from the tag connected with a double underscore (_), without any other connecting character or whitespace.

7. Relationship to Other Specifications

This specification is expected to participate in a standard NEA architecture. As such, it is expected to be used in conjunction with the other protocols used in a NEA exchange. In particular, SWIMA Attributes are conveyed over PB-TNC [RFC5793], which is in turn conveyed over some variant of PT (either PT-TLS [RFC6876] or PT-EAP [RFC7171]). These protocols have an especially important role, as they are responsible for ensuring that attributes defined under this specification are delivered reliably, securely, and to the appropriate party.

It is important to note that the Product Information, Numeric Version, and String Version attributes defined in the PA-TNC specification [RFC5792] are also meant to convey information about installed applications and the versions thereof. As such, there is some conceptual overlap between those attributes and the intent of this specification. However, PA-TNC was designed to respond to very specific queries about specific classes of products, while the SWIMA specification is able to convey a broader query, resulting in a more comprehensive set of evidence regarding an endpoint’s installed software. As such, this specification provides important capabilities not present in the PA-TNC specification.

The NEA architecture is intended to support a broad range of activities and, as such, might be employed by other specifications. For example, requirement T-001 in the SACM Requirements document [RFC8248] notes that NEA can support data collection from endpoints within the broader SACM architecture. (Other parts of the NEA architecture, which SWIMA uses, meet the other SACM data transport requirements.) In the SACM architecture, a SWIMA-PV corresponds to a "SACM Collector" and a SWIMA-PC corresponds to a "SACM Internal Collector". In the SACM architecture, the SWIMA specification can support activities relating to software inventory collection. Specifically, SWIMA supports the SACM "Endpoint Posture Attribute Value Collection" use case (section 2.1.3 in [RFC7632]) by describing a collection mechanism that enables event driven, scheduled, and ad-hoc data collection of software inventory information. SWIMA's flexibility with regard to the format of inventory data records means that it is compatible with virtually any data format that implements SACM's "Define, Publish, Query, and Retrieve Security Automation Data" (section 2.1.1 in [RFC7632]). This is just one example of how SWIMA can support broader security solution standards. Note that, while SWIMA can support these SACM use cases, SWIMA has no dependencies upon the SACM architecture or any other context in which NEA might reasonably be applied.

8. Security Considerations

This section discusses some of the security threats facing Posture Collectors and Posture Validators that implement the SWIMA protocol. This section primarily notes potential issues for implementers to consider, although it does contain a handful of normative requirements to address certain security issues. The issues identified below focus on capabilities specific to this document. Implementers are advised to consult other relevant NEA specifications, particularly [RFC5209] (the NEA Architecture) and [RFC5792] (PA-TNC), for security issues that are applicable to such components in general.

8.1. Evidentiary Value of Software Inventory Evidence Records

The degree to which an endpoint's Software Inventory Evidence Collection accurately reflects the endpoint's actual software load and any changes made to this software load is dependent on the accuracy of the tools used to populate and manage the software inventory evidence records in this collection. While the SWIMA-PC is required to detect changes to an endpoint's Software Inventory Evidence Collection in near real-time, some tools might not be designed to update records in the Software Inventory Evidence Collection in real time. This can result in a collection that is out-of-sync with actual system state. Moreover, tools might inaccurately characterize software, or fail to properly record its removal. Finally, it is likely that there will be software on the endpoint that is not tracked by any source and thus is not reflected in the Software Inventory Evidence Collection. Tools that implement SWIMA ought to be aware of these potential issues and minimize them, but completely eliminating such issues is likely impossible. Users of collected software inventory evidence records need to understand that the information provided by the SWIMA capability cannot be treated as completely accurate. Nonetheless, having endpoints report this information can still provide useful insights into the state of the endpoint’s software load, and can alert administrators and policy tools of situations that require remediation.

8.2. Sensitivity of Collected Records

Collected software records can be sensitive in nature. This can include both security sensitivities and privacy sensitivities. Privacy sensitivities are discussed more in Section 9. With regard to security, inventory records represent a wealth of information about the endpoint in question and, for an adversary who does not already have access to the endpoint, a collection of the endpoint’s inventory records might provide many details that are useful for mounting an attack. A list of the inventory records associated with an endpoint reveals a list of software installed on the endpoint. This list can be very detailed, noting specific versions and even patch levels, which an adversary can use to identify vulnerable software and design efficacious attacks.

In addition, other information might also be gleaned from a collection of software inventory records:

- o An inventory record might include information about where the product was installed on a given endpoint. This can reveal details about the file organization of that endpoint that an attacker can utilize.

- o An inventory record might include information about how the software was provided to the endpoint, who in an organization signs off on the package release, and who packaged the product for installation. This information might be used as a starting point for the development of supply chain attacks.

- o Events affecting inventory records are reported with timestamps indicating when each given event occurred. This can give the attacker an indication of how quickly an organization distributes patches and updates, helping the attacker determine how long an attack window might remain open.

Any consolidated software inventory is a potential risk, because such an inventory can provide an adversary an insight into the enterprise’s configuration and management process. It is recommended that a centralized software inventory record collection be protected against unauthorized access. Mechanisms to accomplish this can include encrypting the data at rest, ensuring that access to the data is limited only to authorized individuals and processes, and other basic security precautions.

8.3. Integrity of Endpoint Records

SWIMA-PCs maintain records of detected changes to the endpoint’s Software Inventory Evidence Collection. These records are used to respond to a SWIMA-PV’s request for change events. The SWIMA-PV might use a list of reported events to update its understanding of the endpoint’s Software Inventory Evidence Collection without needing to receive a full inventory report from the SWIMA-PC. For this reason, preserving the integrity of the SWIMA-PC’s record of events is extremely important. If an attacker modifies the SWIMA-PC’s record of changes to the endpoint’s Software Inventory Evidence Collection, this might cause the SWIMA-PV’s understanding of the endpoint’s Software Inventory Evidence Collection to differ from its actual state. Results might include leading the SWIMA-PV to believe that absent software was present, that present software was absent, that patches have been installed even if this is not the case, or to be unaware of other changes to Software Inventory Evidence Records. As such, the SWIMA-PC MUST take steps to protect the integrity of its event records.

In addition, records of established SWIMA-PV subscriptions also require protection against manipulation or corruption. If an attacker is able to modify or delete records of an established subscription by a SWIMA-PV, the SWIMA-PC might fail to correctly fulfill this subscription. The SWIMA-PV would not be aware that its subscription was not being correctly fulfilled unless it received additional information that indicated a discrepancy. For example, the SWIMA-PV might collect a full inventory and realize from this that certain events had not been correctly reported in accordance with an established subscription. For this reason, the SWIMA-PC MUST protect the integrity of subscription records.

8.4. SWIMA-PC Access Permissions

A SWIMA-PC requires sufficient permissions to collect Software Inventory Evidence Records from all of its supported sources, as well as sufficient permissions to interact with the endpoint’s Posture Broker Client. With regard to the former, this might require permissions to read the contents of directories throughout the file system. Depending on the operating environment and other activities undertaken by a SWIMA-PC (or software that incorporates a SWIMA-PC as one of its capabilities) additional permissions might be required by the SWIMA-PC software. The SWIMA-PC SHOULD NOT be granted permissions beyond what it needs in order to fulfill its duties.

8.5. Sanitization of Record Fields

Not all sources of software inventory evidence are necessarily tightly controlled. For example, consider a source that gathers .swid files from the endpoint's file system. Any party could create a new .swid file that could be collected and turned into a Software Inventory Evidence Record. As a result, it is important that the contents of source information not be automatically trusted. In particular, tools that read source information and the Software Inventory Evidence Records derived therefrom, including SWIMA-PCs, need to be careful to sanitize input to prevent buffer overflow attacks, encoding attacks, and other weaknesses that might be exploited by an adversary who can control the contents of a record.

8.6. PA-TNC Security Threats

In addition to the aforementioned considerations the SWIMA protocol is subject to the same security threats as other PA-TNC transactions, as noted in Section 5.2 of PA-TNC [RFC5792]. These include, but are not limited to, attribute theft, message fabrication, attribute modification, attribute replay, attribute insertion, and denial of service. Implementers are advised to consult the PA-TNC specification to better understand these security issues.

9. Privacy Considerations

As noted in Section 8.2, if an adversary can gain an understanding of the software installed on an endpoint, they can utilize this to launch attacks and maintain footholds on this endpoint. For this reason, the NEA Server needs to ensure adequate safeguards are in place to prevent exposure of collected inventory records. For similar reasons, it is advisable that an endpoint only send records to a NEA Server that is authorized to receive this information and that can be trusted to safeguard this information after collection.

In addition, software inventory information can lead to insights about the endpoint's primary user if that user is able to install software. (Note that users might be "able" to install their own software even if they are not "allowed" to do so.) This is especially true on endpoints that support "apps", as individual apps can be closely tied to specific groups or activities. This could conceivably allow inferences about things such as a user's hobbies, the banks and other financial institutions that they use, and information about the user's race, sex, or sexual orientation.

Organizations that collect software inventory information from endpoints ought to make sure the endpoints' users are aware of this collection. In addition, organizations should be aware that a software inventory associated with an individual, such as the inventory of the individual's primary endpoint, could expose sensitive personal information. For this reason, privacy safeguards are necessary for collected inventory information. Such safeguards would require not only protection of the inventory's confidentiality, but also appropriate access controls so that only those trained in relevant privacy requirements are able to view the data.

10. IANA Considerations

This section extends multiple existing IANA registries. Specifically, it extends the PA-TNC Attribute Types and PA-TNC Error Codes defined in the PA-TNC specification [RFC5792] and the PA-Subtypes registry defined in the PB-TNC specification [RFC5793] and extended in PA-TNC. This specification only adds values to these registries and does not alter how these registries work or are maintained. Consult the appropriate specifications for details on the operations and maintenance of these registries.

10.1. PA Subtypes

The following is an extension to the PA Subtype registry [2] defined in section 7.2 of the PA-TNC specification [RFC5792].

PEN	Integer	Name	Defining Specification
0	9	SWIMA Attributes	[RFC-to-be]

10.2. Registry for PA-TNC Attribute Types

Section 5.4 of this specification defines several new PA-TNC attributes. The following values are added to the registry for PA-TNC Attribute Types defined in the PA-TNC specification. Note that Table 1 in Section 5.4 lists these attributes but uses a hexadecimal value to identify their associated integer. The integer values given in that table are identical to those provided here. Note also that Table 1 includes an entry for PA-TNC Error attributes, but the IANA information associated with that attribute is already defined in the PA-TNC specification and is not reproduced here.

PEN	Integer	Name	Defining Specification
0	13	SWIMA Request	[RFC-to-be]
0	14	Software Identifier Inventory	[RFC-to-be]
0	15	Software Identifier Events	[RFC-to-be]
0	16	Software Inventory	[RFC-to-be]
0	17	Software Events	[RFC-to-be]
0	18	Subscription Status Request	[RFC-to-be]
0	19	Subscription Status Response	[RFC-to-be]
0	20	Source Metadata Request	[RFC-to-be]
0	21	Source Metadata Response	[RFC-to-be]

10.3. Registry for PA-TNC Error Codes

Section 5.16 of this specification defines several new PA-TNC Error Codes. The following values are added to the registry for PA-TNC Error Codes defined in the PA-TNC specification. Note that Table 8 in Section 5.16 lists these codes but uses a hexadecimal value to identify their associated integer. The integer values given in that table are identical to those provided here.

PEN	Integer	Name	Defining Specification
0	4	SWIMA_ERROR	[RFC-to-be]
0	5	SWIMA_SUBSCRIPTION_DENIED_ERROR	[RFC-to-be]
0	6	SWIMA_RESPONSE_TOO_LARGE_ERROR	[RFC-to-be]
0	7	SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR	[RFC-to-be]
0	8	SWIMA_SUBSCRIPTION_ID_REUSE_ERROR	[RFC-to-be]

10.4. Registry for Software Data Models

The name for this registry is "Software Data Model Types". Each entry in this registry should include a human readable name, an SMI Private Enterprise Number, a decimal integer value between 1 and 2^8-1 (inclusive), and a reference to the specification where the use of this data model is defined. This specification needs to provide both a description of the format used by the data model and the procedures by which Software Identifiers are derived from a record expressed using this data model. Note that a specification that just defines the data model structure and its use is generally not sufficient as it would likely lack the procedures for constructing a Software Identifier. This is why the table below references the current specification for ISO SWID tags, rather than using the actual ISO SWID tag specification.

The following entries for this registry are defined in this document. They are the initial entries in the registry for Software Data Model Types. Additional entries to this registry are added by Expert Review with Specification Required, following the guidelines in [RFC8126].

PEN	Integer	Name	Defining Specification
0	0	ISO 2015 SWID Tags using XML	[RFC-to-be]
0	1	ISO 2009 SWID Tags using XML	[RFC-to-be]
0	192-255	Reserved for local enterprise use	N/A

11. References

11.1. Normative References

[NIST8060]
Waltermire, D., Cheikes, B., Feldman, L., and G. Witte, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", April 2016.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5792, DOI 10.17487/RFC5792, March 2010, <https://www.rfc-editor.org/info/rfc5792>.

[RFC8089] Kerwin, M., "The "file" URI Scheme", RFC 8089, DOI 10.17487/RFC8089, February 2017, <https://www.rfc-editor.org/info/rfc8089>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SWID09] The International Organization for Standardization/ International Electrotechnical Commission, "Information Technology - Software Asset Management - Part 2: Software Identification Tag, ISO/IEC 19770-2", November 2009.

[SWID15] The International Organization for Standardization/ International Electrotechnical Commission, "Information Technology - Software Asset Management - Part 2: Software Identification Tag, ISO/IEC 19770-2", October 2015.

11.2. Informative References

[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>.

[RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5793, DOI 10.17487/RFC5793, March 2010, <https://www.rfc-editor.org/info/rfc5793>.

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

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

[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>.

11.3. URIs

[1] https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

[2] https://www.iana.org/assignments/pb-tnc-parameters/pb-tnc-parameters.xhtml#pb-tnc-parameters-2

Authors' Addresses

Charles Schmidt
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: cmschmidt@mitre.org

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

Email: dhaynes@mitre.org

Chris Coffin
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: ccoffin@mitre.org

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

Email: david.waltermire@nist.gov

Jessica Fitzgerald-McKay
United States National Security Agency
9800 Savage Road
Ft. Meade, Maryland
USA

Email: jmfitz2@radium.ncsc.mil

SACM Working Group S. Banghart
Internet-Draft D. Waltermire
Intended status: Informational NIST
Expires: January 22, 2020 July 21, 2019

Definition of the ROLIE Software Descriptor Extension
draft-ietf-sacm-rolie-softwaredescriptor-08

Abstract

This document uses the "information-type" extension point as defined in the Resource-Oriented Lightweight Information Exchange (ROLIE) [RFC8322] Section 7.1.2 to better support Software Record and Software Inventory use cases. This specification registers a new ROLIE information-type, "software-descriptor", that allows for the categorization of information relevant to software description activities and formats. In particular, the usage of the ISO 19770-2:2015 Software Identification Tag (SWID Tag) and the Concise SWID (COSWID) formats in ROLIE are standardized. Additionally, this document discusses requirements and usage of other ROLIE elements in order to best syndicate software description information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines an extension to the Resource-Oriented Lightweight Information Exchange (ROLIE) [RFC8322] to support the publication of software descriptor information. Software descriptor information is information that characterizes static software components, packages, and installers; including identification, version, software creation and publication, and file artifact information.

Software descriptor information provides data about what might be installed, but doesn't describe a specific software installation's configuration or execution. This static approach to software description is a tightly limited scope that still covers the majority of current use cases for software inventory and record keeping.

Some possible use cases for software descriptor information ROLIE Feeds (Section 6.1 of [RFC8322]) include:

- o Software providers can publish software descriptor information so that software researchers, enterprises, and users of software can understand the collection of software produced by that software provider.

- o Organizations can aggregate and syndicate collections of software descriptor information provided by multiple software providers to support software-related analysis processes (e.g., vulnerability analysis) and to provide downsteam services (e.g., software configuration checklist repositories).

- o End user organizations can consume software descriptor information along with related software vulnerability and configuration information to provide the data needed to automate software asset, patch, and configuration management practices.

- o Organizations can use software descriptors to support verification of other entities through integrity measurement mechanisms.

This document supports these use cases by describing the content requirements for Feeds and Entries of software descriptor information that are to be published to or retrieved from a ROLIE repository.

2. Terminology

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

As an extension of [RFC8322], this document refers to many terms defined in that document. In particular, the use of "Entry" and "Feed" are aligned with the definitions presented in RFC8322.

Several places in this document refer to the "information-type" of a Resource (Entry or Feed). This refers to the "term" attribute of an "atom:category" element whose scheme is "urn:ietf:params:rolie:category:information-type". For an Entry, this value can be inherited from it's containing Feed as per [RFC8322].

3. Background

In order to effectively protect and secure an endpoint, it is vital to know what the software load of that endpoint is. Software load, the combination of software, patches and installers on a device, represents a significant portion of the endpoint's attack surface. Unfortunately, without a reliable and secure package manager, or a secured and managed operating system with strict software whitelisting, tracking what software is installed on an endpoint is currently not feasible without undue effort. Even attempting to whitelist software is difficult without a way of identifying software and its editions, versions and hotfixes.

Software descriptor information, such as that standardized in the ISO 19770-2:2015 Software Identification Tag (SWID) format or expressed in proprietary enterprise databases, attempts to provide as much data about this software as possible.

Once this information is expressed, it needs to be stored and shared to internal and external parties. ROLIE provides a mechanism to handle this sharing in an automation-friendly way.

4. The "software-descriptor" information type

When an "atom:category" element has a "scheme" attribute equal to "urn:ietf:params:rolie:category:information-type", the "term" attribute defines the information type of the associated resource. A new valid value for this "term": "software-descriptor", is described in this section and registered in Section 8.1. When this value is used, the resource in question is considered to have an information-type of "software-descriptor" as per [RFC8322] Section 7.1.2.

The "software-descriptor" information type represents any static information that describes a piece of software. This document uses the definition of software provided by [RFC4949]. Note that as per this definition, this information type pertains to static software, that is, code on the disc. The "software-descriptor" information type is intended to provide a category for information that does one or more of the following:

identifies and characterizes software: information that provides quantative and qualitative data describing software. This information identifies and charaterizes a given instance of software.

provides software installer metadata: information about software used to install other software. This metadata identifies, and characterizes a software installation package or media.

describes stateless installation metadata: information that describes the software post-deployment, such as files that may be deployed during an installation. It is expected that this metadata is produced generally for a given installation, and may not exactly match the actual installed files on a given endpoint.

Provided below is a non-exhaustive list of information that may be considered to be of a software-descriptor information type.

- o Naming information: IDs and names that aid in the identification of a piece of software

- o Version and patching information: Version numbers, patch identifiers, or other information that relates to software updates and patches.

- o Vendor and source information: Includes where the software was developed or distributed, as well as where the software installation media may be located.

- o Payload and file information: information that describes or enumerates the files and folders that make up the piece of software, and information about those files.

- o Descriptive information and data: Any information that otherwise characterizes a piece of software, such as libraries, runtime environments, target operating systems, intended purpose or audience, etc.

It is important to note that software descriptor information is static for a given piece of software. That is, the information expressed is the data that doesn't change from the publication of the software to its final install. Information about the current status (e.g. install location, memory usage, CPU usage, launch parameters, job progress, etc.), is out of scope of this information type.

5. rolie:property Extensions

This document registers new valid rolie:property names as follows:

5.1. urn:ietf:params:rolie:property:swd:swname

This property provides an exposure point for the plain text name of the software being described. Naming of software is not a well standardized process, and software names can change between product versions or editions. As such, care should be taken that this value is set as consistently as possible by generating it directly from an attached software descriptor resource.

5.2. urn:ietf:params:rolie:property:swd:swversion

This property provides an exposure point for the version of the software being described. This value should be generated or taken from the software descriptor linked to by the entry. This helps avoid, but does not prevent, inconsistent versioning schemes being shared.

5.3. urn:ietf:params:rolie:property:swd:swcreator

This property provides an exposure point for a plain text name of the creator of the software being described. This is in many cases an organization or company, but certainly could be a single person. Most software descriptor formats include this information, and where possible, this property should be set equal to that value.

6. Data format requirements

This section defines usage guidance and additional requirements related to data formats above and beyond those specified in [RFC8322]. The following formats are expected to be commonly used to express software descriptor information. For this reason, this document specifies additional requirements to ensure interoperability.

6.1. The ISO SWID 2015 format

6.1.1. Description

ISO/IEC 19770-2:2015 defines a software record data format referred to as a "SWID Tag". It provides several tag types:

- o primary: provides descriptive and naming information about software,

- o patch: describes non-standalone software meant to patch existing software,

- o corpus: describes the software installation media that installs a given piece of software,

- o supplemental: provides additional metadata to be deployed alongside a tag.

For a more complete overview as well as normative requirements, refer to ISO/IEC 19770-2:2015 [SWID].

For additional requirements and guidance around creation of SWID Tags, consult NIST Internal Report 8060 [NISTIR8060].

6.1.2. Requirements

For an Entry to be considered as a "SWID Tag Entry", it MUST fulfill the following conditions:

- The information-type of the Entry is "software-descriptor". For a typical Entry, this is derived from the information type of the Feed it is contained in. For a standalone Entry, this is provided by an "atom:category" element.

- The document linked to by the "href" attribute of the "atom:content" element is a 2015 SWID Tag per ISO/IEC 19770-2:2015.

A "SWID Tag Entry" MUST conform to the following requirements:

- The value of the "type" attribute of the "atom:content" element MUST be "application/xml".

- There MUST be one "rolie:property" with the "name" attribute equal to "urn:ietf:params:rolie:property:content-id" and the "value" attribute exactly equal to the "<tagid>" element in the attached SWID Tag. This allows ROLIE consumers to more easily search for SWID tags without needing to download the tag itself.

- There MUST be one "rolie:property" with the "name" attribute equal to "urn:ietf:params:rolie:property:swd:swname", and the "value" attribute equal to the value of the "<name>" element in the attached SWID Tag. As above, this helps ROLIE consumers search and filter Entries.

- There MAY be a property element with the "name" attribute equal to "urn:ietf:params:rolie:property:swd:swversion". When this property appears, its value MUST be equal to the value of the "version" element in the attached SWID Tag.

6.2. The Concise SWID format

6.2.1. Description

The Concise SWID (COSWID) format is an alternative representation of the SWID Tag format using a Concise Binary Object Representation (CBOR) encoding. CBOR provides the format with a reduced size that is more suitable for constrained devices. COSWID provides the same features and attributes as are specified in ISO 19770-2:2015, plus:

- a straight forward method to sign and encrypt using COSE, and

- additional attributes that provide an improved structure to include file hashes intended to be used as Reference Integrity Measurements (RIM).

For more information and the complete specification, refer to the COSWID internet draft [I-D.ietf-sacm-coswid].

6.2.2. Requirements

For an Entry to be considered as a "COSWID Tag Entry", it MUST fulfill the following conditions:

- The information-type of the Entry is "software-descriptor". For a typical Entry, this is derived from the information-type of the Feed it is contained in. For a standalone Entry, this is provided by an "atom:category" element.

- The document linked to by the "href" attribute of the "atom:content" element is a COSWID Tag per [I-D.ietf-sacm-coswid]

A "COSWID Tag Entry" MUST conform to the following requirements:

- The value of the "type" attribute of the atom:content element MUST be "application/swid+cbor".

- There MUST be one "rolie:property" with the "name" attribute equal to "urn:ietf:params:rolie:property:content-id" and the "value" attribute exactly equal to the decoded "tag-id" element in the attached COSWID Tag (mapped to integer 0). This allows ROLIE consumers to more easily search for COSWID tags without needing to download the tag itself.

- There MUST be one "rolie:property" with the "name" attribute equal to "urn:ietf:params:rolie:property:swd:swname", and the "value" attribute equal to the decoded value of the "swid-name" element in the attached COSWID Tag (mapped to the integer 1). As above, this helps ROLIE consumers search and filter Entries.

- There MAY be a property element with the "name" attribute equal to "urn:ietf:params:rolie:property:swd:swversion". When this property appears, it’s value MUST be equal to the decoded value of the tag-version element in the attached COSWID Tag (mapped to the integer 12).

7. atom:link Extensions

This section defines additional link relationships that implementations MUST support. These relationships are not registered in the Link Relation IANA table as their use case is too narrow. Each relationship is named and described.

Name	Description
ancestor	Links to a software descriptor resource that defines an ancestor of the software being described by this Entry. This is usually a previous version of the software.
descendent	Links to a software descriptor resource that defines an descendent of the software being described by this Entry. This is usually a more recent version or edition of the software.
patches	Links to a software descriptor resource that defines the software being patched by this software
patchedby	Links to a software descriptor resource that defines the patch or update itself that can be or has been applied to this software.
requires	Links to a software descriptor resource that defines a piece of software required for this software to function properly, i.e., a dependency.
requiredBy	Links to a software descriptor resource that defines a piece of software that requires this software to function properly.
installs	Links to a software descriptor resource that defines the software that is installed by this software.
installedBy	Links to a software descriptor resource that defines the software package that installs this software.

patchesVulnerability	Links to a vulnerability that this software update fixes. Used for software descriptors that describe software patches or updates.
hasVulnerability	Links to a vulnerability description object that details a vulnerability that this software has.

Table 1: Link Relations for Resource-Oriented Lightweight Indicator Exchange

8. IANA Considerations

8.1. software-descriptor information-type

IANA has added an entry to the "ROLIE Security Resource Information Type Sub-Registry" registry located at <https://www.iana.org/assignments/rolie/category/information-type> .

The entry is as follows:

name: software-descriptor

index: TBD

reference: This document, Section 4

8.2. swd:swname property

IANA has added an entry to the "ROLIE URN Parameters" registry located in <https://www.iana.org/assignments/rolie/>.

The entry is as follows:

name: property:swd:swname

Extension IRI: urn:ietf:params:rolie:property:swd:swname

Reference: This document, Section 5.1

Subregistry: None

8.3. swd:swversion property

IANA has added an entry to the "ROLIE URN Parameters" registry located in <https://www.iana.org/assignments/rolie/>.

The entry is as follows:

name: property:swd:swversion

Extension IRI: urn:ietf:params:rolie:property:swd:swversion

Reference: This document, Section 5.1

Subregistry: None

8.4. swd:swcreator property

IANA has added an entry to the "ROLIE URN Parameters" registry located in <https://www.iana.org/assignments/rolie/>.

The entry is as follows:

name: property:swd:swcreator

Extension IRI: urn:ietf:params:rolie:property:swd:swcreator

Reference: This document, Section 5.1

Subregistry: None

9. Security Considerations

Use of this extension implies dealing with the security implications of both ROLIE and of software descriptors in general. As with any data, care should be taken to verify the trustworthiness and veracity of the descriptor information to the fullest extent possible.

Ideally, software descriptors should be signed by the software manufacturer, or signed by whichever agent processed the source code. Software descriptor documents from these sources are more likely to be accurate than those generated by scraping installed software.

These "authoritative" sources of software descriptor content should consider additional security for their ROLIE repository beyond the typical recommendations, as the central importance of the repository is likely to make it a target.

Version information is often represented differently across manufacturers and even across product releases. If using software version information for low fault tolerance comparisons and searches, care should be taken that the correct version scheme is being used.

10. Normative References

[I-D.ietf-sacm-coswid]
Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", draft-ietf-sacm-coswid-11 (work in progress), June 2019.

[NISTIR8060]
Waltermire, D., Cheikes, B., Feldman, L., and G. Witte, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", NISTIR 8060, April 2016, <https://doi.org/10.6028/NIST.IR.8060>.

[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>.

[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>.

[RFC5070] Danyliw, R., Meijer, J., and Y. Demchenko, "The Incident Object Description Exchange Format", RFC 5070, DOI 10.17487/RFC5070, December 2007, <https://www.rfc-editor.org/info/rfc5070>.

[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>.

[SWID] "Information technology - Software asset management - Part 2: Software identification tag", ISO/IEC 19770-2:2015, October 2015, <https://www.iso.org/standard/65666.html>.

Appendix A. Schema

This document does not require any schema extensions.

Appendix B. Examples of Use

Use of this extension in a ROLIE repository will not typically change that repository's operation. As such, the general examples provided by the ROLIE core document would serve as examples. Provided below is a sample software descriptor ROLIE entry:


```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
  xmlns:rolie="urn:ietf:params:xml:ns:rolie-1.0">
  <id>dd786dba-88e6-440b-9158-b8fae67ef67c</id>
  <title>Sample Software Descriptor</title>
  <published>2015-08-04T18:13:51.0Z</published>
  <updated>2015-08-05T18:13:51.0Z</updated>
  <summary>A descriptor for a piece of software published by this
  organization. </summary>
  <link rel="self" href="http://www.example.org/rolie/SWD/123456"/>
  <link rel="feed" href="http://www.example.org/rolie/SWD/"/>
  <link rel="requires" href="http://www.example.org/rolie/SWD/78430"/>
  <rolie:property name=urn:ietf:params:rolie:property:swd:swname
      value="Example Software Name"/>
  <category
      scheme="urn:ietf:params:rolie:category:information-type"
      term="software-descriptor"/>
  <rolie:format
      ns="http://standards.iso.org/iso/19770/-2/2015/schema.xsd"/>
  <content type="application/xml"
      src="http://www.example.org/rolie/SWD/123456/data"/>
</entry>
```


Authors' Addresses

Stephen Banghart
NIST
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Email: stephen.banghart@nist.gov

David Waltermire
NIST
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Email: david.waltermire@nist.gov

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

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

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

2. Terms and Definitions

This section describes terms that have been defined by other RFC's and defines new ones. The predefined terms will reference the RFC and where appropriate will be annotated with the specific context by which the term is used in SACM. Note that explanatory or informational augmentation to definitions are segregated from the definitions themselves. The definition for the term immediately follows the term on the same line, whereas expositional text is contained in subsequent paragraphs immediately following the definition.

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

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

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

Assessment: Defined in [RFC5209] as "the process of collecting posture for a set of capabilities on the endpoint (e.g., host-based firewall) such that the appropriate validators may evaluate the posture against compliance policy."

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

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

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

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

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

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

The SACM Vulnerability Assessment Scenario [I-D.ietf-sacm-vuln-scenario] defines the terms Endpoint Management Capabilities, Vulnerability Management Capabilities, and Vulnerability Assessment Capabilities, which illustrate specific sets of SACM capabilities on an enterprise IT department's point of view and therefore compose sets of declarative guidance.

Collection Result: Is a composition of one or more content elements carrying information about a target endpoint, that is produced by a collector when conducting a collection task.

Collection Task: A targeted task that collects attributes and/or corresponding attribute values from target endpoint.

There are four types of frequency collection tasks can be conducted with:

ad-hoc, e.g. triggered by a unsolicited query

conditional, e.g. triggered in accordance with policies included in the compositions of workflows

scheduled, e.g. in regular intervals, such as every minute or weekly

continuously, e.g. a network behavior observation

There are three types of collection methods, each requiring an appropriate set of functions to be included in the SACM component conducting the collection task:

Self-Reporting: A SACM component located on the target endpoint itself conducts the collection task.

Remote-Acquisition: A SACM component located on an Endpoint different from the target endpoint conducts the collection task via interfaces available on the target endpoint, e.g. SNMP/ NETCONF or WMI.

Behavior-Observation: A SACM component located on an Endpoint different from the target endpoint observes network traffic related to the target endpoint and conducts the collection task via interpretation of that network traffic.

Collector: A piece of software that acquires information about one or more target endpoints by conducting collection tasks.

A collector can be distributed across multiple endpoints, e.g. across a target endpoint and a SACM component. The separate parts of the collector can communicate with a specialized protocol, such as PA-TNC [RFC5792]. At least one part of a distributed collector has to take on the role of a provider of information by providing SACM interfaces to propagate capabilities and to provide SACM content in the form of collection results.

Configuration: A non-volatile subset of the endpoint attributes of a endpoint that is intended to be unaffected by a normal reboot-cycle.

Configuration is a type of imperative guidance that is stored in files (files dedicated to contain configuration and/ or files that are software components), directly on block devices, or on specific hardware components that can be accessed via corresponding software components. Modification of configuration can be conducted manually or automatically via management (plane) interfaces that support management protocols, such as SNMP or WMI. A change of configuration can occur during both run-time and down-time of an endpoint. It is common practice to scheduled a change of configuration during or directly after the completion of a boot-cycle via corresponding software components located on the target endpoint itself.

Examples: The static association of an IP address and a MAC address in a DHCP server configuration, a directory-path that identifies a log-file directory, a registry entry.

Configuration Drift: The disposition of endpoint characteristics to change over time.

Configuration drift exists for both hardware components and software components. Typically, the frequency and scale of configuration drift of software components is significantly higher than the configuration drift of hardware components.

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

Content Element: Content elements constitute the payload data (SACM content) transferred via statement Subjects emitted by providers of information. Every content element Subject includes a specific content Subject and a corresponding content metadata Subject.

Content Metadata: Data about content Subjects. Every content-element includes a content metadata Subject. The Subject can include any information element that can annotate the content transferred. Examples include time stamps or data provenance Subjects.

Control Plane: An architectural component that provides common control functions to all SACM components.

Typically used as a term in the context of routing, e.g. [RFC6192]. SACM components may include authentication, authorization, (capability) discovery or negotiation, registration and subscription. The control plane orchestrates the flow on the data plane according to imperative guidance (i.e. configuration) received via the management plane. SACM components with interfaces to the control plane have knowledge of the capabilities of other SACM components within a SACM domain.

Controller: A controller is a SACM Role that is assigned to a SACM component containing control plane functions managing and facilitating information sharing or execute on security functions.

There are three types of SACM controllers: Broker, Proxy, and Repository. Depending on its type, a controller can also contain functions that have interfaces on the data plane.

Data Confidentiality: Defined in [RFC4949] as "the property that data is not disclosed to system entities unless they have been authorized to know the data."

Data In Motion: Data that is being transported via a network; also referred to as "Data in Transit" or "Data in Flight".

Data in motion requires a data model to transfer the data using a specific encoding. Typically, data in motion is serialized (marshalling) into a transport encoding by a provider of information and deserialized (unmarshalling) by a consumer of information. The termination points of provider of information and consumer of information data is transferred between are interfaces. In regard to data in motion, the interpretation of the roles consumer of information and provider of information depends on the corresponding OSI layer (e.g. on layer2: between interfaces connected to a broadcast domain, on layer4: between interfaces that maintain a TCP connection). In the context of SACM, consumer of information and provider of information are SACM components.

Data At Rest: Data that is stored.

Data at rest requires a data model to encode the data to be stored. In the context of SACM, data at rest located on a SACM component can be provided to other SACM components via discoverable capabilities.

Data Integrity: Defined in [RFC4949] as "the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner."

Data Origin: The SACM Component that initially acquired or produced data about an endpoint.

Data Origin enables a SACM component to identify the SACM component that initially acquired or produced data about a (target) endpoint (e.g. via collection from a data source) and made it available to a SACM domain via a SACM statement. Data Origin can be expressed by an endpoint label information element (e.g. to be used as metadata in statement).

Data Plane: Is an architectural component providing operational functions enabling information exchange that is not command and control or management related.

Typically used as a term in the context of routing (and used as a synonym for forwarding plane, e.g. [RFC6192]). In the context of SACM, the data plane is an architectural component providing operational functions to enable a SACM component to provide and consume SACM statements and therefore SACM content, which composes the actual SACM content. The data plane in a SACM domain is used to conduct distributed SACM tasks by transporting SACM content via specific transport encodings and corresponding operations defined by SACM data models.

Data Provenance: An historical record of the sources, origins and evolution, as it pertains to data, that is influenced by inputs, entities, functions and processes.

Additional Information - In the context of SACM, data provenance is expressed as metadata that identifies SACM statements and corresponding content elements a new statement is created from. In a downstream process, this references can cascade, creating a data provenance tree that enables SACM components to trace back the original data sources involved in the creation of SACM statements and take into account their characteristics and trustworthiness.

Data Source: Is an endpoint from which a particular set of attributes and/or attribute values have been collected.

Data Source enables a SACM component to identify - and potentially characterize - a (target) endpoint that is claimed to be the original source of endpoint attributes in a SACM statement. Data Source can be expressed as metadata by an endpoint label information element or a corresponding subject of identifying endpoint attributes.

Endpoint: Defined in [RFC5209] as "any computing device that can be connected to a network."

Additional Information - The [RFC5209] definition continues, "Such devices normally are associated with a particular link layer address before joining the network and potentially an IP address once on the network. This includes: laptops, desktops, servers, cell phones, or any device that may have an IP address."

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

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

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

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

Endpoint Attribute: Is a discreet endpoint characteristic that is computably observable.

Endpoint Attributes typically constitute Attributes that can be bundled into Subject (e.g. information about a specific network interface can be represented via a set of multiple AVP).

Endpoint Characteristics: The state, configuration and composition of the software components and (virtual) hardware components a target endpoint is composed of, including observable behavior, e.g. sys-calls, log-files, or PDU emission on a network.

In SACM work-flows, (Target) Endpoint Characteristics are represented via Information Elements.

Endpoint Characterization Task: The task of endpoint characterization that uses endpoint attributes that represent distinct endpoint characteristics.

Endpoint Classification: The categorization of of the endpoint into one or more taxonomic structures.

Endpoint classification requires declarative guidance in the form of an endpoint profile, discovery results and potentially collection results. Types, classes or the characteristics of an individual target endpoint are defined via endpoint profiles.

Endpoint Classification Task: The task of endpoint classification that uses an endpoint's characteristics to determine how to categorize the given endpoint into one or more taxonomic structures.

Endpoint Label: A unique label associated with a unique endpoint.

Endpoint specializations have corresponding endpoint label specializations. For example, an endpoint label used on a SACM Component is a SACM Component Label.

Endpoint Management Capabilities: Enterprise IT management capabilities that are tailored to manage endpoint identity, endpoint information, and associated metadata.

Evaluation Task: A task by which an endpoint's asserted attribute value is evaluated against a policy-compliant attribute value.

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

Expected Endpoint Attribute State: The policy-compliant state of an endpoint attribute that is to be compared against.

Sets of expected endpoint attribute states are transported as declarative guidance in target endpoint profiles via the management plane. This, for example, can be a policy, but also a recorded past state. An expected state is represented by an Attribute or a Subject that represents a set of multiple attribute value pairs.

Guidance: Machine-processable input directing SACM processes or tasks.

Examples of such processes/tasks include automated device management, remediation, collection, evaluation. Guidance influences the behavior of a SACM Component and is considered content of the management plane. In the context of SACM, guidance is machine-readable and can be manually or automatically generated or provided. Typically, the tasks that provide guidance to SACM components have a low-frequency and tend to be sporadic.

There are two types of guidance:

Declarative Guidance: Guidance that defines the configuration or state an endpoint is supposed to be in, without providing specific actions or methods to produce that desired state. Examples include Target Endpoint Profiles or network topology based requirements.

Imperative Guidance: Guidance that prescribes specific actions to be conducted or methods to be used in order to achieve an outcome. Examples include a targeted Collection Task or the IP-Address of a SACM Component that provides a registration function.

Prominent examples include: modification of the configuration of a SACM component or updating a target endpoint profile that resides on an evaluator. In essence, guidance is transported via the management plane.

Endpoint Hardware Inventory: The set of hardware components that compose a specific endpoint representing its hardware configuration.

Hardware Component: A distinguishable physical component used to compose an endpoint.

The composition of an endpoint can be changed over time by adding or removing hardware components. In essence, every physical endpoint is potentially a composite of multiple hardware components, typically resulting in a hierarchical composition of hardware components. The composition of hardware components is based on interconnects provided by specific hardware types (e.g. FRU in a chassis are connected via redundant busses). In general, a hardware component can be distinguished by its serial number. Occasionally, hardware components are referred to as power sucking aliens.

Information Element: A representation of information about physical and virtual "objects of interest".

Information elements are the building blocks that constitute the SACM information model. In the context of SACM, an information element that expresses a single value with a specific name is referred to as an Attribute (analogous to an attribute-value-pair). A set of attributes that is bundled into a more complex composite information element is referred to as a Subject. Every information element in the SACM information model has a unique name. Endpoint attributes or time stamps, for example, are represented as information elements in the SACM information model.

Information Model: An abstract representation of data, their properties, relationships between data and the operations that can be performed on the data.

While there is some overlap with a data model, [RFC3444] distinguishes an information model as being protocol and implementation neutral whereas a data model would provide such details. The purpose of the SACM information model is to ensure interoperability between SACM data models (that are used as transport encoding) and to provide a standardized set of information elements for communication between SACM components.

Interaction Model: The definition of specific sequences regarding the exchange of messages (data in motion), including, for example, conditional branching, thresholds and timers.

An interaction model, for example, can be used to define operations, such as registration or discovery, on the control plane. A composition of data models for data in motion and a corresponding interaction model is a protocol.

Internal Collector: A collector that runs on a target endpoint to acquire information from that target endpoint.

Management Plane: An architectural component providing common functions to steer the behavior of SACM components, e.g. their behavior on the control plane.

Typically, a SACM component can fulfill its purpose without continuous input from the management plane. In contrast, without continuous availability of control plane functions a typical SACM component could not function properly. In general, interaction on the management plane is less frequent and less regular than on the control plane. Input via the management plane can be manual (e.g. via a CLI), or can be automated via management plane functions that are part of other SACM components.

Network Address: A layer-specific address that follows a layer-specific address scheme.

The following characteristics are a summery derived from the Common Information Model and ITU-T X.213. Each Network Interface of a specific layer can be associated with one or more addresses appropriate for that layer. There is no guarantee that a network address is globally unique. A dedicated authority entity can provide a level of assurance that a network address is unique in its given scope. In essence, there is always a scope to a network address, in which it is intended to be unique.

Examples include: physical Ethernet port with a MAC address, layer 2 VLAN interface with a MAC address, layer 3 interface with multiple IPv6 addresses, layer 3 tunnel ingress or egress with an IPv4 address.

Network Interface: An Endpoint is connected to a network via one or more Network Interfaces. Network Interfaces can be physical (Hardware Component) or logical (virtual Hardware component, i.e. a dedicated Software Component). Network Interfaces of an Endpoint can operate on different layers, most prominently what is now commonly called layer 2 and 3. Within a layer, interfaces can be nested.

In SACM, the association of Endpoints and Network Addresses via Network Interfaces is vital to maintain interdependent autonomous processes that can be targeted at Target Endpoints, unambiguously.

Examples include: physical Ethernet port, layer 2 VLAN interface, a MC-LAG setup, layer 3 Point-to-Point tunnel ingress or egress.

Metadata: Data about data.

In the SACM information model, data is referred to as Content. Metadata about the content is referred to as Content-Metadata, respectively. Content and Content-Metadata are combined into Subjects called Content-Elements in the SACM information model. Some information elements defined by the SACM information model can be part of the Content or the Content-Metadata. Therefore, if an information element is considered data or data about data depends on which kind of Subject it is associated with. The SACM information model also defines metadata about the data origin via the Subject Statement-Metadata. Typical examples of metadata are time stamps, data origin or data source.

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

This term is used within the scope of SACM to represent the configuration and state information that is collected from a target endpoint in the form of endpoint attributes (e.g. software/ hardware inventory, configuration settings, dynamically assigned addresses). This information may constitute one or more posture attributes.

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

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

Provider: A provider is a SACM role assigned to a SACM component that provides role-specific functions to provide information to other SACM components.

Repository: A repository is a controller that contains functions to consume, store and provide information of a particular kind.

Such information is typically data transported on the data plane, but potentially also data and metadata from the control and management plane. A single repository may provide the functions of more than one specific repository type (i.e. configuration baseline repository, assessment results repository, etc.)

SACM Broker Controller: A SACM Broker Controller is a controller that contains control plane functions to provide and/or connect services on behalf of other SACM components via interfaces on the control plane.

A broker may provide, for example, authorization services and find, upon request, SACM components providing requested services.

SACM Component: Is a component, as defined in [I-D.ietf-i2nsf-terminology], that is composed of SACM capabilities.

In the context of SACM, a set of SACM functions composes a SACM component. A SACM component conducts SACM tasks, acting on control plane, data plane and/or management plane via corresponding SACM interfaces. SACM defines a set of standard components (e.g. a collector, a broker, or a data store). A SACM component contains at least a basic set of control plane functions and can contain data plane and management plane functions. A SACM component residing on an endpoint assigns one or more SACM roles to the corresponding endpoint due to the SACM functions it is composed of. A SACM component "resides on" an endpoint and an endpoint "contains" a SACM component, correspondingly. For example, a SACM component that is composed solely of functions that provide information would only take on the role of a provider.

SACM Component Discovery: The task of discovering the capabilities provided by SACM components within a SACM domain.

This is likely to be performed via an appropriate set of control plane functions.

SACM Component Label: A specific endpoint label that is used to identify a SACM component.

In content-metadata, this label is called data origin.

SACM Content: The payload provided by SACM components to the SACM domain on the data plane.

SACM content includes the SACM data models.

SACM Domain: Endpoints that include a SACM component compose a SACM domain.

(To be revised, additional definition content TBD, possible dependencies to SACM architecture)

SACM Function: A behavioral aspect of a SACM component that provides external SACM Interfaces or internal interfaces to other SACM Functionse.

For example, a SACM Function with SACM Interfaces on the Control Plane can provide a brokering function to other SACM Components. Via Data Plane interfaces, a SACM Function can act as a provider and/or as a consumer of information. SACM Functions can be propagated as the Capabilities of a SACM Component and can be discovered by or negotiated with other SACM Components.

SACM Interface: An interface, as defined in [I-D.ietf-i2nsf-terminology], that provides SACM-specific operations.

[I-D.ietf-i2nsf-terminology] defines interface as a "set of operations one object knows it can invoke on, and expose to, another object," and further defines interface by stating that an interface "decouples the implementation of the operation from its specification. An interface is a subset of all operations that a given object implements. The same object may have multiple types of interfaces to serve different purposes."

In the context of SACM, SACM Functions provide SACM Interfaces on the management, control, or data plane. Operations a SACM Interface provides are based on corresponding data model defined by SACM. SACM Interfaces are used for communication between SACM components.

SACM Proxy Controller: A SACM Proxy Controller is a controller that provides data plane and control plane functions, information, or services on behalf of another component, which is not directly participating in the SACM architecture.

SACM Role: Is a role, as defined in [I-D.ietf-i2nsf-terminology], that requires the SACM Component assuming the role to bear a set of SACM functions or interfaces.

SACM Roles provide three important benefits. First, it enables different behavior to be supported by the same Component for different contexts. Second, it enables the behavior of a Component to be adjusted dynamically (i.e., at runtime, in response)to changes in context, by using one or more Roles to define the behavior desired for each context. Third, it decouples the Roles of a Component from the Applications that use that Component."

In the context of SACM, SACM roles are associated with SACM components and are defined by the set of functions and interfaces a SACM component includes. There are three SACM roles: provider, consumer, and controller. The roles associated with a SACM component are determined by the purpose of the SACM functions and corresponding SACM interfaces the SACM component is composed of.

SACM Statement: Is an assertion that is made by a SACM Component.

Security Automation: The process of which security alerts can be automated through the use of different components to monitor, analyze and assess endpoints and network traffic for the purposes of detecting misconfigurations, misbehaviors or threats.

Security Automation is intended to identify target endpoints that cannot be trusted (see "trusted" in [RFC4949]. This goal is achieved by creating and processing evidence (assessment statements) that a target endpoint is not a trusted system [RFC4949].

Software Package: A generic software package (e.g. a text editor).

Software Component: A software package installed on an endpoint.

The software component may include a unique serial number (e.g. a text editor associated with a unique license key).

Software Instance: A running instance of a software component.

For example, on a multi-user system, one logged-in user has one instance of a text editor running and another logged-in user has another instance of the same text editor running, or on a single-user system, a user could have multiple independent instances of the same text editor running.

State: A volatile set of endpoint attributes of a (target) endpoint that is affected by a reboot-cycle.

Local state is created by the interaction of components with other components via the control plane, via processing data plane payload, or via the functional properties of local hardware and software components. Dynamic configuration (e.g. IP address distributed dynamically via an address distribution and management services, such as DHCP) is considered state that is the result of the interaction with another component (e.g. provided by a DHCP server with a specific configuration).

Examples: The static association of an IP address and a MAC address in a DHCP server configuration, a directory-path that identifies a log-file directory, a registry entry.

Statement: A statement is the root/top-level subject defined in the SACM information model.

A statement is used to bundle Content Elements into one subject and includes metadata about the data origin.

Subject: A semantic composite information element pertaining to a system entity that is a target endpoint.

Like Attributes, subjects have a name and are composed of attributes and/or other subjects. Every IE that is part of a subject can have a quantitiy associated with it (e.g. zero-one, none-unbounded). The content IE of a subject can be an unordered or an ordered list.

In contrast to the definitions of subject provided by [RFC4949], a subject in the scope of SACM is neither "a system entity that causes information to flow among objects or changes the system state" nor "a name of a system entity that is bound to the data items in a digital certificate".

In the context of SACM, a subject is a semantic composite of information elements about a system entity that is a target endpoint. Every acquirable subject-as defined in the scope of SACM-about a target endpoint represents and therefore identifies every subject-as defined by [RFC4949]-that is a component of that target endpoint. The semantic difference between both definitions can be subtle in practice and is in consequence important to highlight.

Supplicant: A component seeking to be authenticated via the control plane for the purpose of participating in a SACM domain.

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

Target Endpoint: Is an endpoint that is under assessment at some point in, or region of, time.

Every endpoint that is not specifically designated as an excluded endpoint is a target endpoint. A target endpoint is not part of a SACM domain unless it contains a SACM component (e.g. a SACM component that publishes collection results coming from an internal collector).

A target endpoint is similar to a device that is a Target of Evaluation (TOE) as defined in Common Criteria and as referenced by {{RFC4949}.

Target Endpoint Address: An address that is layer specific and which follows layer specific address schemes.

Each interface of a specific layer can be associated with one or more addresses appropriate for that layer. There is no guarantee that an address is globally unique. In general, there is a scope to an address in which it is intended to be unique.

Examples include: physical Ethernet port with a MAC address, layer 2 VLAN interface with a MAC address, layer 3 interface with multiple IPv6 addresses, layer 3 tunnel ingress or egress with an IPv4 address.

Target Endpoint Characterization: The description of the distinctive nature of a target endpoint, that is based on its characteristics.

Target Endpoint Characterization Record: A set of endpoint attributes about a target endpoint that was encountered in a SACM domain, which are associated with that target endpoint as a result of a Target Endpoint Characterization Task.

A characterization record is intended to be a representation of an endpoint. It cannot be assured that a record distinctly represents a single target endpoint unless a set of one or more endpoint attributes that compose a unique set of identifying endpoint attributes are included in the record. Otherwise, the set of identifying attributes included in a record can match more than one target endpoints, which are - in consequence - indistinguishable to a SACM domain until more qualifying endpoint attributes can be acquired and added to the record. A characterization record is maintained over time in order to assert that acquired endpoint attributes are either about an endpoint that was encountered before or an endpoint that has not been encountered before in a SACM domain. A characterization record can include, for example, acquired configuration, state or observed behavior of a specific target endpoint. Multiple and even conflicting instances of this information can be included in a characterization record by using timestamps and/or data origins to differentiate them. The endpoint attributes included in a characterization record can be used to re-identify a distinct target endpoint over time. Classes or profiles can be associated with a characterization record via the Classification Task in order to guide collection, evaluation or remediation tasks.

Target Endpoint Characterization Task: An ongoing task of continuously adding acquired endpoint attributes to a corresponding record. The TE characterization task manages the representation of encountered target endpoints in the SACM domain in the form of characterization records. For example, the output of a target endpoint discovery task or a collection task can be processed by the characterization task and added to the record. The TE characterization Task also manages these representations of target endpoints encountered in the SACM domain by splitting or merging the corresponding records as new or more refined endpoint attributes become available.

Target Endpoint Classification Task: The task of associating a class from an extensible list of classes with an endpoint characterization record. TE classes function as imperative and declarative guidance for collection, evaluation, remediation and security posture assessment in general.

Target Endpoint Discovery Task: The ongoing task of detecting previously unknown interaction of a potential target endpoint in the SACM domain. TE Discovery is not directly targeted at a specific target endpoint and therefore an un-targeted task. SACM Components conducting the discovery task as a part of their function are typically distributed and located, for example, on infrastructure components or collect from those remotely via appropriate interfaces. Examples of infrastructure components that are of interest to the discovery task include routers, switches, VM hosting or VM managing components, AAA servers, or servers handling dynamic address distribution.

Target Endpoint Identifier: The target endpoint discovery task and the collection tasks can result in a set of identifying endpoint attributes added to a corresponding Characterization Record. This subset of the endpoint attributes included in the record is used as a target endpoint identifier, by which a specific target endpoint can be referenced. Depending on the available identifying attributes, this reference can be ambiguous and is a "best-effort" mechanism. Every distinct set of identifying endpoint attributes can be associated with a target endpoint label that is unique in a SACM domain.

Target Endpoint Label: An endpoint label that identifies a specific target endpoint.

Target Endpoint Profile: A bundle of expected or desired component composition, configurations and states that is associated with a target endpoint.

The corresponding task by which the association with a target endpoint takes places is the endpoint classification task. The task by which an endpoint profile is created is the endpoint characterization task. A type or class of target endpoints can be defined via a target endpoint profile. Examples include: printers, smartphones, or an office PC.

In respect to [RFC4949], a target endpoint profile is a protection profile as defined by Common Criteria (analogous to the target endpoint being the target of evaluation).

SACM Task: Is a task conducted within the scope of a SACM domain by one or more SACM functions that achieves a SACM-defined outcome.

A SACM task can be triggered by other operations or functions (e.g. a query from another SACM component or an unsolicited push on the data plane due to an ongoing subscription). A task is part of a SACM process chain. A task starts at a given point in time and ends in a deterministic state. With the exception of a collection task, a SACM task consumes SACM statements provided by other SACM components. The output of a task is a result that can be provided (e.g. published) on the data plane.

The following tasks are defined by SACM:

Target Endpoint Discovery

Target Endpoint Characterization

Target Endpoint Classification

Collection

Evaluation [TBD]

Information Sharing [TBD]

SACM Component Discovery

SACM Component Authentication [TBD]

SACM Component Authorization [TBD]

SACM Component Registration [TBD]

Timestamps : Defined in [RFC4949] as "with respect to a data object, a label or marking in which is recorded the time (time of day or other instant of elapsed time) at which the label or marking was affixed to the data object".

A timestamp always requires context, i.e. additional information elements that are associated with it. Therefore, all timestamps wrt information elements are always metadata. Timestamps in SACM Content Elements may be generated outside a SACM Domain and may be encoded in an unknown representation. Inside a SACM domain the representation of timestamps is well-defined and unambiguous.

Virtual Endpoint: An endpoint composed entirely of logical system components (see [RFC4949]).

The most common example is a virtual machine/host running on a target endpoint. Effectively, target endpoints can be nested and at the time of this writing the most common example of target endpoint characteristics about virtual components is the EntLogicalEntry in [RFC6933].

Vulnerability Assessment: An assessment specifically tailored to determining whether a set of endpoints is vulnerable according to the information contained in the vulnerability description information.

Vulnerability Description Information: Information pertaining to the existence of a flaw or flaws in software, hardware, and/or firmware, which could potentially have an adverse impact on enterprise IT functionality and/or security.

Vulnerability description information should contain enough information to support vulnerability detection.

Vulnerability Detection Data: A type of imperative guidance extracted or derived from vulnerability description information that describes the specific mechanisms of vulnerability detection that is used by an enterprise's vulnerability management capabilities to determine if a vulnerability is present on an endpoint.

Vulnerability Management Capabilities: An IT management capability tailored toward managing endpoint vulnerabilities and associated metadata on an ongoing basis by ingesting vulnerability description information and vulnerability detection data, and performing vulnerability assessments.

Vulnerability assessment capabilities: An assessment capability that is tailored toward determining whether a set of endpoints is vulnerable according to vulnerability description information.

Workflow: A workflow is a modular composition of tasks that can contain loops, conditionals, multiple starting points and multiple endpoints.

The most prominent workflow in SACM is the assessment workflow.

3. IANA Considerations

This memo includes no request to IANA.

4. Security Considerations

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

5. Acknowledgements

6. Change Log

Changes from version 00 to version 01:

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

Changes from version 01 to version 02:

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

Changes from version 02 to version 03:

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

Changes from version 03 to version 04:

- TODO

Changes from version 04 to version 05:

- TODO

Changes from version 05 to version 06:

- Updated author information.
- Combined "Pre-defined Terms" with "New Terms and Definitions".
- Removed "Requirements language".
- Removed unused reference to use case draft; resulted in removal of normative references.
- Removed introductory text from Section 1 indicating that this document is intended to be temporary.
- Added placeholders for missing change log entries.

Changes from version 06 to version 07:

- Added Contributors section.

- Updated author list.

- Changed title from "Terminology for Security Assessment" to "Secure Automation and Continuous Monitoring (SACM) Terminology".

- Changed abbrev from "SACM-Terms" to "SACM Terminology".

- Added appendix The Attic to stash terms for future updates.

- Added Authentication, Authorization, Data Confidentiality, Data Integrity, Data Origin, Data Provenance, SACM Component, SACM Component Discovery, Target Endpoint Discovery.

- Major updates to Building Block, Function, SACM Role, Target Endpoint.

- Minor updates to Broker, Capability, Collection Task, Evaluation Task, Posture.

- Relabeled Role to SACM Role, Endpoint Target to Target Endpoint, Endpoint Discovery to Endpoint Identification.

- Moved Asset Targeting, Client, Endpoint Identification to The Attic.

- Endpoint Attributes added as a TODO.

- Changed the structure of the Change Log.

Changes from version 07 to version 08:

- Added Assertion, Collection Result, Collector, Excluded Endpoint, Internal Collector, Network Address, Network Interface, SACM Domain, Statement, Target Endpoint Identifier, Target Endpoint Label, Timestamp.

- Major updates to Attributes, Broker, Collection Task, Consumer, Controller, Control Plane, Endpoint Attributes, Expected Endpoint State, SACM Function, Provider, Proxy, Repository, SACM Role, Target Endpoint.

- Minor updates to Asset, Building Block, Data Origin, Data Source, Data Provenance, Endpoint, Management Plane, Posture, Posture Attribute, SACM Component, SACM Component Discovery, Target Endpoint Discovery.

- o Relabeled Function to SACM Function.

Changes from version 08 to version 09:

- o Updated author list.
- o Added Data Plane, Endpoint Characterization, Endpoint Classification, Guidance, Interaction Model, Software Component, Software Instance, Software Package, Statement, Target Endpoint Profile, SACM Task.
- o Removed Building Block.
- o Major updates to Control Plane, Endpoint Attribute, Expected Endpoint State, Information Model, Management Plane.
- o Minor updates to Attribute, Capabilities, SACM Function, SACM Component, Collection Task.
- o Moved Asset Characterization to The Attic.

Changes from version 09 to version 10:

- o Added Configuration Drift, Data in Motion, Data at Rest, Endpoint Management Capability, Hardware Component, Hardware Inventory, Hardware Type, SACM Interface, Target Endpoint Characterization Record, Target Endpoint Characterization Task, Target Endpoint Classification Task, Target Endpoint Discovery Task, Vulnerability Description Information, Vulnerability Detection Data, Vulnerability Management Capability, Vulnerability Assessment
- o Added references to i2nsf definitions in Capability, SACM Component, SACM Interface, SACM Role.
- o Added i2nsf Terminology I-D Reference.
- o Major Updates to Endpoint, SACM Task, Target Endpoint Identifier.
- o Minor Updates to Guidance, SACM Component Discovery, Target Endpoint Label, Target Endpoint Profile.
- o Relabeled SACM Task
- o Removed Target Endpoint Discovery

Changes from version 10 to version 11:

- Added Content Element, Content Metadata, Endpoint Label, Information Element, Metadata, SACM Component Label, Workflow.

- Major Updates to Assessment, Capability, Collector, Endpoint Management Capabilities, Guidance, Vulnerability Assessment Capabilities, Vulnerability Detection Data, Vulnerability Assessment Capabilities.

- Minor updates to Collection Result, Control Plane, Data in Motion, Data at Rest, Data Origin, Network Interface, Statement, Target Endpoint Label.

- Relabeled Endpoint Management Capability, Vulnerability Management Capability, Vulnerability Assessment.

Changes from version 11 to version 12:

- Added Configuration, Endpoint Characteristic, Event, SACM Content, State, Subject.

- Major Updates to Assertion, Data in Motion, Data Provenance, Data Source, Interaction Model.

- Minor Updates to Attribute, Control Plane, Data Origin, Data Provenance, Expected Endpoint State, Guidance, Target Endpoint Classification Task, Vulnerability Detection Data.

Changes from version 12 to version 13:

- Added Virtual Component.

- Major Updates to Capability, Collection Task, Hardware Component, Hardware Type, Security Automation, Subject, Target Endpoint, Target Endpoint Profile.

- Minor Updates to Assertion, Data Plane, Endpoint Characteristics.

Changes from version 13 to version 14:

- Handled a plethora of issues listed in GitHub.

- Pruned some commonly understood terms.

- Narrowing term labels per their definitions.

- In some cases, excised expositional text.

- o Where expositional text was left intact, it has been separated from the actual definition of a term.

Changes from version 14 to version 16:

- o moved obsolete definitions into the Appendix (attic).

7. Contributors

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

Email: david.waltermire@nist.gov

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

Email: adam.w.montville@gmail.com

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

Email: ietfdbh@comcast.net

Brian Ford
Lancope
3650 Brookside Parkway, Suite 500
Alpharetta, GA 30022
USA

Email: bford@lancope.com

Merike Kaeo
Double Shot Security
3518 Fremont Avenue North, Suite 363
Seattle, WA 98103
USA

Email: merike@doubleshotsecurity.com

8. References

8.1. Normative References

[RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5792, DOI 10.17487/RFC5792, March 2010, <https://www.rfc-editor.org/info/rfc5792>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <https://www.rfc-editor.org/info/rfc6933>.

8.2. Informative References

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-06 (work in progress), July 2018.

[I-D.ietf-netmod-entity]
Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-08 (work in progress), January 2018.

[I-D.ietf-sacm-vuln-scenario]
Coffin, C., Cheikes, B., Schmidt, C., Haynes, D., Fitzgerald-McKay, J., and D. Waltermire, "SACM Vulnerability Assessment Scenario", draft-ietf-sacm-vuln-scenario-02 (work in progress), September 2016.

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

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

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

[RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <https://www.rfc-editor.org/info/rfc6192>.

[X.1252] "ITU-T X.1252 (04/2010)", n.d..

Appendix A. The Attic

The following terms are stashed for now and will be updated later:

Asset: Is a system resource, as defined in [RFC4949], that may be composed of other assets.

Examples of Assets include: Endpoints, Software, Guidance, or X.509 public key certificates. An asset is not necessarily owned by an organization.

Asset Management: The IT process by which assets are provisioned, updated, maintained and deprecated.

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

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

Client: An architectural component receiving services from another architectural component.

Endpoint Identification (TBD per list; was "Endpoint Discovery"): The process by which an endpoint can be identified.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Jarrett Lu
Oracle Corporation
4180 Network Circle
Santa Clara, CA 95054
USA

Email: jarrett.lu@oracle.com

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95138
USA

Email: john.sc.strassner@huawei.com

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

Email: ncamwing@cisco.com

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

Email: adam.w.montville@gmail.com

The Data Model of Network Infrastructure Device Management Plane Security Baseline
draft-lin-sacm-nid-mp-security-baseline-04

Abstract

This document provides security baseline for network device management plane, which is represented by YANG data model. The corresponding configuration values and status values of the YANG data model can be transported between Security Automation and Continuous Monitoring (SACM) components and used for network device security posture assessment.

Table of Contents

1. Introduction

Besides user devices and servers, network devices such as routers, switches, and firewalls are crucial to enterprise network security. The security baseline defined in this document refers to a minimal set of security controls that are essential to provide network security. Organizations can define additional security controls based on the security baseline. Then the security posture of network devices can be assessed by comparing the configuration values and status values with the required security controls.

Network devices typically perform three planes of operation: management plane, control plane and data plane. All the planes should be protected and monitored. This document focuses on security baseline for management plane. Management plane provides configuration and monitoring services to network administrators or device owners. Unauthorized access, insecure access channels, weak cryptographic algorithms are common security issues that break management plane security. A number of security best practices have been proposed to deal with these security issues, such as disabling unused services and ports, discarding insecure access channels, and enforcing strong user authentication and authorization. In this document, we provide a minimal set of security controls that are expected to be widely applicable to common network devices. To assess security posture of network devices, the configurations that are effective on network devices and the current status of the networks devices will be compared with the reference values defined by an organization or a third party.

YANG data model is used to describe the security baseline defined in this document. [I-D.birkholz-sacm-yang-content] defines a method to construct the YANG data model scheme for network device security posture assessment by brokering YANG push telemetry through SACM statements. In this document, we follow the same way to define the YANG output for network device security posture based on the [I-D.ietf-sacm-information-model].

Besides management plane, the security baselines for control plane, data plane, and infrastructure layer of network infrastructure devices are described in [I-D.dong-sacm-nid-cp-security-baseline], [I-D.xia-sacm-nid-dp-security-baseline] and [I-D.dong-sacm-nid-infra-security-baseline] respectively.

2. Requirements Language

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

3. Terminology

This document uses the terms defined in [RFC7950] and [RFC8342].

4. Tree Diagrams

Tree diagram defined in [RFC8340] is used to represent the YANG data model of network device management plane security. The meaning of the symbols used in the tree diagram and the syntax are as follows:

- o A module is identified by "module:" followed the module-name. The top-level data nodes defined in the module, offset by 2 spaces. Submodules are represented in the same fashion as modules, but are identified by "submodule:" followed the (sub)module-name.

- o Groupings, offset by 2 spaces, and identified by the keyword "grouping" followed by the name of the grouping and a colon (":") character.

- o Each node in the tree is prefaces with "+--". Schema nodes that are children of another node are offset from the parent by 3 spaces.

- o Brackets "[" and "]" enclose list keys.

- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" means state data (read-only), and "-u" indicates the use of a predefined grouping.

- o Symbols after data node names: "?" means an optional leaf, choice, anydata, or anyxml, "!" means a presence container, and "*" denotes a "list" or "leaf-list".

- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- o At times when the composition of the nodes within a module schema is not important in the context of the presented tree, sibling nodes and their children can be collapsed using the notation "..." in place of the text lines used to represent the summarized nodes.

- o Curly brackets and a question mark "{...}?" are combined to represent the features that node depends on.

5. Data Model Structure

The security baseline defined in this document consists of security configuration and runtime security status for administration, system management, port management, log, files.

- o Administration security

- System management security

- Port management security

- Log security

- File security

A multitude of YANG modules for network devices and network protocols have been defined in IETF. Several RFCs and drafts model some parts of management plane security. But an overall data model of management plane security is still missing. New modules, groupings, and nodes are defined in this document as supplements. And the existing YANG modules are reused. Appendix A provides a summary of existing YANG modules and the relationship to the security baseline defined in this document.

5.1. Administration Security

5.1.1. Administrative Account Security

In order to provide administrative accounts, security controls on account properties and passwords should be applied. The commonly applied security controls include limiting the length of account name, checking the password complied to the complexity policy, forbidding the use of some strings in password, blocking accounts after several login fails, etc. The following data model illustrates these kinds of security controls.

```
module: ietf-admin-account-security
  +--rw ietf-admin-account-security
     +--rw account-security-policy {account-security}?
     |  +--rw policy-status?          boolean
     |  +--rw account-aging-period?   uint64
     |  +--rw account-name-minlen?    uint64
     +--rw pwd-security-policy {pwd-security}?
     |  +--rw expire-days?           uint64
     |  +--rw prompt-days?           uint64
     |  +--rw change-check?          boolean
     |  +--rw complexity-check?      boolean
     |  +--ro history-pwd-num?       uint64
     |  +--rw pwd-minlen?            uint64
     |  +--rw forbidden-word-rules?
     |     +--rw forbidden-word-rule* [forbidden-word]
     |        +--rw forbidden-word   string
     +--rw login-failed-limit {login-failed-block}?
        +--rw failed-times?          uint64
        +--rw period?                uint64
        +--rw reactive-time?         uint64
```

5.1.2. Administrator Access Security

Network devices typically can be managed through command line interface (CLI) or web user interface. Insecure access channels (e.g., Telnet), can expose the devices to threats and attacks. Therefore, SSH-based access channels and HTTPS-based web channels should be used. Besides, the right version of the protocols should be chosen. For example, SSHv1 is considered not secure, SSHv2 is recommended. And draft [I-D.ietf-tls-oldversions-deprecate] will formally deprecates Transport Layer Security (TLS) versions 1.0 [RFC2246] and 1.1 [RFC4346] and moves these documents to the historic state.

```
module: ietf-admin-access-security
  +--rw ietf-admin-access-security
     +--rw console
     |  +--rw auth-mode?               auth-mode-type
     |  +--rw privilege-level?         uint8
     +--rw vtys
     |  +--rw vty* [vty-number]
     |     +--rw vty-number            uint8
     |     +--rw auth-mode             auth-mode-type
     |     +--rw privilege-level       uint8
     |     +--rw acl-name-list*        string
     |     +--rw ip-block-enable       boolean
     |     +--rw ip-block-limit {ip-block-config}?
     |        +--rw failed-times?      uint64
     |        +--rw period?            uint64
     |        +--rw reactive-time?     uint64
     +--rw ssh
     |  +--rw ssh-enable?              boolean
     |  +---u ssh-server-attribute-grouping
     |  +---u ssh-security-harden-grouping
     |  +--rw ip-block-enable          boolean
     |  +--rw ip-block-limit {ip-block-config}?
     |     +--rw failed-times?         uint64
     |     +--rw period?               uint64
     |     +--rw reactive-time?        uint64
     +--rw web  {web-interface}?
        +--rw privilege-level?         uint8
        +--rw http-server-interface?   string
        +--rw https-ipv4-enable?       boolean
        +--rw https-ipv6-enable?       boolean
        +--rw https-source-port?       inet:port-number
        +--rw https-timeout?           uint32
        +--rw acl-name-list*?          string
        +--rw ip-block-enable          boolean
        +--rw ip-block-limit {ip-block-config}?
        |  +--rw failed-times?         uint64
        |  +--rw period?               uint64
        |  +--rw reactive-time?        uint64
        +---u tls-server-attribute-grouping
```

[I-D.ietf-netconf-ssh-client-server] defines "ssh-server-grouping" for configuring SSH server and does not consider the underlying transport parameters. And it reuses the groupings defined in [I-D.ietf-netconf-keystore]. Because this document focuses on the security configurations that are actively in use when the network device acts as a SSH server, the "ssh-server-attribute-grouping" defined here tailors the "private-key" node and the "certificate-expiration" notification of "ssh-server-grouping". The tree diagram of grouping "ssh-server-attribute-grouping":

```
  grouping ssh-server-attribute-grouping:
    +--rw server-identity
    |  +--rw host-key* [name]
    |     +--rw name   string
    |     +--rw (host-key-type)
    |        +--:(public-key)
    |        |  +--rw (local-or-keystore)
    |        |     +--:(local)
    |        |     |  +---u ks:public-key-grouping
    |        |     +--:(keystore) {ks:keystore-implemented}?
    |        |        +--rw ref?  ks:asymmetric-key-certificate-ref
    |        +--:(certificate) {sshcmn:ssh-x509-certs}?
    |           +--rw (local-or-keystore)
    |              +--:(local)
    |              |  +---u ks:public-key-grouping
    |              |  +---u ks:trust-anchor-cert-grouping
    |              +--:(keystore) {ks:keystore-implemented}?
    |                 +--rw ref?  ks:asymmetric-key-certificate-ref
    +--rw client-cert-auth {sshcmn:ssh-x509-certs}?
    |  +--rw pinned-ca-certs?       ta:pinned-certificates-ref
    |  +--rw pinned-client-certs?   ta:pinned-certificates-ref
    +--rw transport-params {ssh-server-transport-params-config}?
       +---u sshcmn:transport-params-grouping
```

Besides the security configurations defined "ssh-server-attribute-grouping", there are several other features related the secure use and configuration of SSH, such as which SSH version is used, whether the network device support to be compatible with earlier SSH versions, whether the port number has been changed, etc. The "ssh-security-harden-grouping" includes these kind of security configurations and state. The tree diagram of grouping "ssh-security-harden-grouping":

```
  grouping ssh-security-harden-grouping:
    +--ro ssh-version                uint32
    +--rw ssh-server-port?           inet:port-number
    +--rw ssh-rekey-interval?        uint32
    +--rw ssh-timeout?               uint32
    +--rw ssh-retry-times?           uint32
    +--rw ssh1x-compatible?          boolean
    +--rw ssh-server-interface?      string
```

[I-D.ietf-netconf-tls-client-server] defines "tls-server-grouping" for configuring TLS server and does not consider the underlying transport parameters. And it reuses the groupings defined in

[I-D.ietf-netconf-keystore]. Because this document focuses on the security configurations that are actively in use when the network device acts as a web server and build connections through HTTPS, the "tls-server-attribute-grouping" defined here tailors the "private-key" node and the "certificate-expiration" notification of "tls-server-grouping". The tree diagram of grouping "tls-server-attribute-grouping":

```
grouping tls-server-attribute-security-grouping:
  +--rw server-identity
  |  +--rw (local-or-keystore)
  |     +--:(local)
  |     |  +---u ks:public-key-grouping
  |     |  +---u ks:trust-anchor-cert-grouping
  |     +--:(keystore) {ks:keystore-implemented}?
  |        +--rw ref?   ks:asymmetric-key-certificate-ref
  +--rw client-auth
  |  +--rw pinned-ca-certs?       ta:pinned-certificates-ref
  |  +--rw pinned-client-certs?   ta:pinned-certificates-ref
  +--rw hello-params {tls-server-hello-params-config}?
     +--rw tls-versions
     |  +--rw tls-version*     identityref
     +--rw cipher-suites
        +--rw cipher-suite*    identityref
```

5.1.3. AAA

Authentication, Authorization, and Accounting (AAA) provides user management for network devices. RADIUS (Remote Authentication Dial In User Service) and TACACS+ (Terminal Access Controller Access Control System) are the commonly used AAA mechanisms. In order to implement AAA, network devices act as AAA clients to communicate with AAA servers. [RFC7317] defined YANG module for client to configure the RADIUS authentication server information. In this document, authentication, authorization and accounting schemes, as well as AAA server lists are all included.

```
module: ietf-aaa-security
  +--rw ietf-aaa-security
     +--rw authentication-scheme* [authen-scheme-name]
     |  +--rw authen-scheme-name   string
     |  +--rw authen-mode*         aaa-authen-mode
     |  +--rw authen-type?         radius-authen-type
     |  +--rw authen-fail-policy?  boolean
     +--rw authorization-scheme* [author-scheme-name]
     |  +--rw author-scheme-name   string
     |  +--rw author-mode*         aaa-author-mode
     |  +--rw cmd-author-mode*     aaa-cmd-author-mode
     +--rw accounting-scheme* [account-scheme-name]
     |  +--rw account-scheme-name  string
     |  +--rw account-mode?        aaa-account-name
     +--rw radius-security
     |  +--rw radius-authen-servers* [address]
     |  |  +--rw address         inet:host
     |  |  +--rw port?           inet:port-number
     |  +--rw radius-author-servers*? [address]
     |  |  +--rw address         inet:host
     |  |  +--rw port?           inet:port-number
     |  +--rw radius-account-servers* [address]
     |     +--rw address         inet:host
     |     +--rw port?           inet:port-number
     +--rw tacacs-security {tacacs-supported}?
        +--rw tacacs-authen-servers* [address]
        |  +--rw address         inet:host
        |  +--rw port?           inet:port-number
        +--rw tacacs-author-servers*? [address]
        |  +--rw address         inet:host
        |  +--rw port?           inet:port-number
        +--rw tacacs-account-servers* [address]
           +--rw address         inet:host
           +--rw port?           inet:port-number
```

5.1.4. Administrator Access Statistics

The statistics of the current online administrators, the failed login attempts and the blocked addresses are useful for the monitoring of network infrastructure devices.

```
module: ietf-admin-access-statistics
  +--ro ietf-admin-access-statistics
     +--ro online
     |  +--ro total-online-users      uint32
     |  +--ro online-admin-list  {display-online-info}?
     |     +--ro online-users* [account-name]
     |        +--ro account-name      string
     |        +--ro ip-address        inet:ip-address-no-zone
     |        +--ro mac-address       yang:mac-address
     +--ro ip-block-list
        +--ro blocked-ip* [ip-address]
           +--ro ip-address           inet:ip-address-no-zone
           +--ro vpn-instance         string
           +--ro state                ip-block-state-type
           +--ro authen-fail-account  uint32
```

5.2. System Management Security

5.2.1. SNMP Management Security

Simple Network Management Protocol (SNMP) is a network management standard to monitor network devices. Three SNMP versions are available: SNMPv1, SNMPv2c, and SNMPv3. [RFC7407] defines community-based security model for SNMPv1 and SNMPv2c, view-based access control model and user-based security model, transport security model for SNMPv3. SNMPv1 and SNMPv2c are lack of authentication and message encryption, which could facilitate unauthorized access to network devices. SNMPv3 needs to be used to authenticate and encrypt payloads. The "ietf-snmp-security" module defined in this section reuses the definitions in [RFC7407], but some modifications and eliminations are made. As this module only focuses on security controls and status of SNMP, the detailed transport information such as IP address and port are not included, while the transport protocol used is under consideration. And the subtree for key configuration is also not needed for user-based security model, but the authentication protocol or encryption protocol used is included.

```
module: ietf-snmp-security
  +--rw ietf-snmp-security
     +--rw snmp-enable?       boolean
     +--rw engine
     |  +--rw enabled?        boolean
     |  +--rw listen* [name]
     |  |  +--rw name        snmp:identifier
     |  |  +--rw transport   snmp-transport-type
     |  +--rw version        snmp-version-type
     |  +--rw enable-authen-traps?  boolean
     +--rw target* [name]
```

```
        |  +--rw name       snmp:identifier
        |  +--rw transport  snmp-transport-type
        |  +--rw target-params   snmp:identifier
        +--rw target-params* [name]
        |  +--rw name        snmp:identifier
        |  +--rw (params)?
        |     +--:(usm)
        |     |  +---u snmp:usm-target-params
        |     +--:(tsm) {snmp:tsm}?
        |     |  +---u snmp:tsm-target-params
        +--rw vacm
        |  +--ro vacm-enable?     boolean
        |  +--rw group* [name]
        |  |  +--rw name                snmp:group-name
        |  |  +--rw member*  [security-name]
        |  |  |  +--rw security-name    snmp:security-name
        |  |  |  +--rw security-model*  snmp:security-model
        |  |  +--rw access* [context security-model security-level]
        |  |     +--rw context          snmp:context-name
        |  |     +--rw context-match?   enumeration
        |  |     +--rw security-model   snmp:security-model-or-any
        |  |     +--rw security-level   snmp:security-level
        |  |     +--rw read-view?       snmp:view-name
        |  |     +--rw write-view?      snmp:view-name
        |  |     +--rw notify-view?     snmp:view-name
        |  +--rw view* [name]
        |     +--rw name      vacm:view-name
        |     +--rw include*  snmp:wildcard-object-identifier
        |     +--rw exclude*  snmp:wildcard-object-identifier
        +--rw usm
        |  +--ro usm-enable?    boolean
        |  +--rw local
        |  |  +---u user-auth-priv
        |  +--rw remote
        |     +---u user-auth-priv
        +--rw tsm   {tsm}?
           +--ro tsm-enable?    boolean
```

The tree diagram of grouping "user-auth-priv":

```
  grouping user-auth-priv:
    +--rw user* [name]
          +--rw name              snmp:identifier
       +--rw auth-protocol     auth-pro-type
       +--rw priv-protocol     priv-pro-type
```

5.2.2. NETCONF Management Security

The NETCONF server model defined in [I-D.ietf-netconf-netconf-client-server] supports both the SSH and TLS transport protocols. The "ietf-netconf-security" module defined in this section only reused the security related subtrees and replaces the SSH and TLS related groupings with those defined in "ietf-admin-access-security" module.

```
module: ietf-netconf-security
  +--rw ietf-netconf-security
     +--rw netconf-enable?     boolean
     +--rw listen {ncs:listen}?
     |  +--rw endpoint* [name]
     |     +--rw name          string
     |     +--rw (transport)
     |        +--:(ssh) {ssh-listen}?
     |        |  +--rw port    inet:port-number
     |        |  +---u accsec:ssh-server-attribute-grouping
     |        +--:(tls) {tls-listen}?
     |           +--rw port    inet:port-number
     |           +---u accsec:tls-server-attribute-grouping
     +--rw call-home {call-home}?
        +--rw netconf-client* [name]
           +--rw name         string
           +--rw endpoints
              +--rw endpoint* [name]
                 +--rw name   string
                 +--rw (transport)
                    +--:(ssh) {ssh-call-home}?
                    |  +--rw port    inet:port-number
                    |  +---u accsec:ssh-server-attribute-grouping
                    +--:(tls) {tls-call-home}?
                       +--rw port    inet:port-number
                       +---u accsec:tls-server-attribute-grouping
```

5.3. Port Management Security

As it is suggested to disable unused service and ports, the current status (open or shut-down) of the ports that are available on the network devices can be retrieved and compared with the communication matrix to check the device security posture.

```
module: ietf-port-management-security
  +--rw ietf-port-management-security
     +--rw port-list*  [port-number]
        +--rw port-number       inet:port-number
        +--rw port-status        boolean
```

5.4. Log Security

To monitor the running status and diagnose faults or attacks on network devices, the activities of network administrators, the operations conducted on devices, and the security notification of abnormal events need to be recorded. Besides, policy should be defined to deal with log overflow. Log records can be outputted to console, or stored locally, or outputted to remote Syslog server. The following defined "ietf-log-security" module reuses the security configuration of log remote transfer in [I-D.ietf-netmod-syslog-model], and adds access control for locally stored log files.

```
module: ietf-log-security
  +--rw ietf-log-security
     +--rw alert-notification
     |  +--rw login-fail-threshold         uint8
     |  +--rw system-abnormal              boolean
     |  +--rw attack                       boolean
     |  +--rw log-overflow-lost            boolean
     +--rw (log-overflow-action)
     |  +--:(rewrite-when-overflow)        boolean
     |  |  +--ro rewrite-numbers           uint16
     |  +--:(discard-new-logs)             boolean
     |     +--ro discard-numbers           uint16
     +--rw (log-mode)
        +--:(file) {file-action}?
        |  +--rw user-level-for-read       uint8
        |  +--rw user-level-for-delete     uint8
        +--:(remote) {remote-action}?      [I-D.ietf-netmod-syslog-model]
           +--rw destination* [name]
              +--rw name                   string
              +--rw (transport)
              |  ...
              +--rw signing! {signed-messages}?
                 ...
```

5.5. File Security

Patches, packages, configuration files, password files are critical system files for network infrastructure devices. Only administrators with certain security privilege levels are allowed to access or operate on these files. For file transfer security, secure protocol should be used.

```
   module: ietf-file-security
     +--rw ietf-file-security
        +--rw role-based-access-control   boolean
        +--rw transport-protocol          file-pro-type
        +--rw (transport)
        |  +--:(sftp) {sftp}?
        |  |  +--rw sftp-enable              boolean
        |  |  +--rw sftp-server-port         inet:port-number
        |  |  +---u accsec:ssh-server-attribute-grouping
        |  |  +---u accsec:ssh-security-harden-grouping
        |  +--:(scp) {scp}?
        |  |  +--rw scp-enable               boolean
        |  |  +--rw scp-server-port          inet:port-number
        |  |  +---u accsec:ssh-server-attribute-grouping
        |  |  +---u accsec:ssh-security-harden-grouping
        |  +--:(ftps) {ftps}?
        |  |  +--rw ftps-enable              boolean
        |  |  +--rw ftps-server-port         inet:port-number
        |  |  +---u accsec:tls-server-attribute-grouping
        +--rw ip-block-enable                boolean
        +--rw ip-block-limit {ip-block-config}?
           +--rw failed-times                uint64
           +--rw period                      uint64
           +--rw reactive-time               uint64
```

6. Network Infrastructure Device Security Baseline Yang Module

6.1. Module 'ietf-admin-account-security'

```
<CODE BEGINS> file "ietf-admin-account-security@2018-10-16.yang"
module ietf-admin-account-security {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-admin-account-security";
  prefix acsec;

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";
```

```
  description
    "This YANG module defines ietf-admin-account-security YANG module, which con
tains configurations that are actively in use for account security control, pass
word security control and administrative account block.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  /*
  * features
  */
  feature account-security {
    description
      "If the network device supports this feature, then several security contro
ls on administrative accounts can be conducted.";
  }

  feature pwd-security {
    description
      "If the network device supports this feature, then several security contro
ls on password can be conducted.";
  }

  feature login-failed-block {
    description
      "If the network device supports this feature, an adminstrative account wil
l be blocked for a certain time range when this account login failed several tim
es in a certain period.";
  }

  /*
  * containers
  */

  container account-security-policy {
    if-feature account-security;
    leaf policy-status {
      type boolean;
      description
        "The status of account security policy: enabled, or disabled.";
    }
    leaf account-aging-period {
      type uint64;
      description
        "The aging period of an administrative account.";
    }
    leaf account-name-minlen {
      type uint64;
      description
        "The minimum length of an administrative account name.";
    }
```

```
    description
      "If the network device supports some security controls on administrative a
ccounts, the configuration that is actively in use will be collected.";
  }

  container pwd-security-policy {
    if-feature pwd-security;
    leaf expire-days {
      type uint64;
      description
        "The password validity period.";
    }
    leaf prompt-days {
      type uint64;
      description
        "The period for warning before the password expires.";
    }
    leaf change-check {
      type boolean;
      description
        "Whether it is mandatory to change the password when logining for the fi
rst time: enabled, or disabled.";
    }
    leaf complexity-check {
      type boolean;
      description
        "The status of password complexity check: enabled, or disabled.";
    }
    leaf history-pwd-num {
      type uint64;
      config false;
      description
        "The newly configured password should not be the same as the several pas
t passwords.";
    }
    leaf pwd-minlen {
      type uint64;
      description
        "The minimum length of a password.";
    }
    container forbidden-word-rules {
      list forbidden-word-rule {
        key "forbidden-word";
        leaf forbidden-word {
          type string;
          description
            "A forbidden word in password.";
        }
        description
          "A list of forbidden words that are not allowed to be used in password
.";
      }
```

```
      description
        "Password blacklist.";
    }
    description
      "If the network device supports some security controls on administrative p
asswords, the configuration that is actively in use will be collected.";
  }

  container login-failed-limit {
    if-feature login-failed-block;
    leaf failed-times {
      type uint64;
      description
        "The failed time in a certain period.";
    }
    leaf peroid {
      type uint64;
      description
        "The certain period in which the failed times are counted.";
    }
    leaf reactive-time {
      type uint64;
      description
        "The reactive time after which the account is not blocked.";
    }
    description
      "If the network device suppor this feature, an account will be blocked for
 a certain time range when it failed to login for several times in a certain per
iod.";
  }
}
<CODE ENDS>
```

6.2. Module 'ietf-admin-access-security'

```
module ietf-admin-access-security {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-admin-access-security";
  prefix accsec;

  import ietf-inet-types {
    prefix inet;
      reference
        "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-ssh-common {
    prefix sshcmn;
      reference
        "draft-ietf-netconf-ssh-client-server - YANG Groupings for SSH Clients a
nd SSH Servers";
  }
```

```
  import ietf-tls-common {
    prefix tlscmn;
      reference
        "draft-ietf-netconf-tls-client-server - YANG Groupings for TLS Clients a
nd SSH Servers";
  }

  import ietf-keystore {
    prefix ks;
      reference
        "draft-ietf-netconf-keystore - YANG Data Model for a Centralized Keystor
e Mechanism";
  }

  import ietf-trust-anchors {
    prefix ta;
      reference
        "draft-ietf-netconf-trust-anchors - YANG Data Model for Global Trust Anc
hors";
  }

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";

  description
    "This YANG module defines ietf-admin-access-security YANG module, which cont
ains security configurations that are actively in use for different access chann
els.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  /*
  * features
  */
  feature web-interface {
    description
      "If the network device supports web interface for administration, then adm
inistrative account can access this device through web interface.";
  }
```

```
  feature ip-block-config {
    description
      "If the network device supports the configuration of ip block function, th
en it can be configured to block the access from a list of IP addresses.";
  }

  feature ssh-server-transport-params-config {
    description
      "SSH transport layer parameters are configurable on an SSH server.";
  }

  feature tls-server-hello-params-config {
    description
      "TLS hello message parameters are configurable on a TLS server.";
  }

  /*
  * typedefs
  */
  typedef auth-mode-type {
    type enumeration {
      enum "none" {
        description
          "Authentication mode: none.";
      }
      enum "password" {
        description
          "Authentication mode: password.";
      }
      enum "aaa" {
         description
           "Authentication mode: aaa.";
      }
    }
    description
      "The Authentication mode of console and vty interface.";
   }

  /*
  * groupings
  */
  grouping ssh-server-attribute-grouping {
    container server-identity {
      list host-key {
        key "name";
        leaf name {
          type string;
          description
            "The name of the host-key.";
```

```
                }
        choice host-key-type {
          mandatory true;
          case public-key {
            choice local-or-keystore {
              case local {
                uses ks:public-key-grouping;
                description
                  "The public key and the corresponding algorithm.";
                          }
              case keystore {
                if-feature ks:keystore-implemented;
                leaf ref {
                  type ks:asymmetric-key-certificate-ref;
                  description
                    "A reference to a value that exists in the keystore.";
                }
                description
                  "The reference of the key pair that stored in the keystore. ";
              }
              description
                "The key pair is locally stored or can be referenced from the ke
ystore.";
            }
            description
              "The host key type is asymmetric key pair.";
          }
          case certificate {
            if-feature sshcmn:ssh-x509-certs;
            choice local-or-keystore {
              case local {
                uses ks:public-key-grouping;
                uses ks:trust-anchor-cert-grouping;
                description
                  "The certificate and the corresponding public key are stored l
ocally.";
              }
              case keystore {
                if-feature ks:keystore-implemented;
                leaf ref {
                  type ks:asymmetric-key-certificate-ref;
                  description
                    "The certificate is referenced by a value that exists in the
 keystore.";
                }
                description
                  "The reference of the certificate that stored in the keystore.
";
              }
              description
                "The certificate is stored locally or can be referenced from the
 keystore.";
            }
```

```
            description
              "The host key type is certificate.";
          }
          description
            "Two types of host key: asymmetric key pair, certificate.";
        }
        description
          "A list of host keys of the network device";
      }
      description
        "The list of host keys the network device (acts as SSH server) will use
to construct its list of algorithms, when sending its SSH-MSG-KEXINIT message, a
se defined in Section 7.1 of RFC 4253.";
        }
    container client-cert-auth {
      if-feature sshcmn:ssh-x509-certs;
      leaf pinned-ca-certs {
        type ta:pinned-certificates-ref;
        description
          "A reference to a list of certificate authority (CA) certificates used
 by the SSH server to authenticate SSH client certificates.";
        reference
          "draft-ietf-netconf-trust-anchors: YANG Data Model for Global Trust An
chors";
      }
      leaf pinned-client-certs {
        type ta:pinned-certificates-ref;
        description
          "A reference to a list of client certificates used by the SSH server t
o authenticate SSH client certificates.";
        reference
          "draft-ietf-netconf-trust-anchors: YANG Data Model for Global Trust An
chors";
      }
      description
        "A reference to a list of pinned certificate authority (CA) certificates
 and a reference to a list of pinned client certificates.";
    }
    container transport-params {
      if-feature ssh-server-transport-params-config;
      uses sshcmn:transport-params-grouping;
      description
        "Configurable parameters of the SSH transport layer.";
    }
    description
      "A reusable grouping of configurations that are actively in use for networ
k devices which act as SSH servers.";
  }

  grouping ssh-security-harden-grouping {
    leaf ssh-version {
      type uint32;
      config false;
      mandatory true;
      description
        "The SSH version that the network device supports.";
```

```
    }
    leaf ssh-server-port {
      type inet:port-number;
      description
        "The port number of SSH server.";
    }
    leaf ssh-rekey-interval {
      type uint32;
      description
        "The interval for updating the key pair of the SSH server.";
    }
    leaf ssh-timeout {
      type uint32;
      description
        "The authentication timeout period of SSH.";
    }
    leaf ssh-retry-times {
      type uint32;
      description
        "The authentication retry times.";
    }
    leaf ssh1x-compatible {
      type boolean;
      description
        "The status of version-compatible function on the SSH server: enabled, d
isabled.";
    }
    leaf ssh-server-interface {
      type string;
      description
        "The source interface of SSH server.";
    }
        description
      "A set of SSH configuration status to enhance security.";
  }

  grouping tls-server-attribute-grouping {
    container server-identity {
      choice local-or-keystore {
          case local {
            uses ks:public-key-grouping;
            uses ks:trust-anchor-cert-grouping;
            description
              "The certificate and the corresponding public key are stored local
ly.";
          }
          case keystore {
            if-feature ks:keystore-implemented;
            leaf ref {
              type ks:asymmetric-key-certificate-ref;
```

```
              description
                "The certificate is referenced by a value that exists in the key
store.";
            }
            description
              "The reference of the certificate that stored in the keystore.";
          }
          description
            "The certificate is stored locally or can be referenced from the key
store.";
      }
      description
        "A locally-defined or referenced end-entity certificate, including any c
onfigured intermediate certificates, the TLS server will present when establishi
ng a TLS connection in its Certificate message, as defined in Section 7.4.2 in R
FC5246.";
        }
    container client-auth {
      leaf pinned-ca-certs {
        type ta:pinned-certificates-ref;
        description
          "A reference to a list of certificate authority (CA) certificates used
 by the TLS server to authenticate TLS client certificates.";
        reference
          "draft-ietf-netconf-trust-anchors: YANG Data Model for Global Trust An
chors";
      }
      leaf pinned-client-certs {
        type ta:pinned-certificates-ref;
        description
          "A reference to a list of client certificates used by the TLS server t
o authenticate TLS client certificates.";
        reference
          "draft-ietf-netconf-trust-anchors: YANG Data Model for Global Trust An
chors";
      }
      description
        "A reference to a list of pinned certificate authority (CA) certificates
 and a reference to a list of pinned client certificates.";
    }
    container hello-params {
      if-feature tls-server-hello-params-config;
      uses tlscmn:hello-params-grouping;
      description
        "Configurable parameters for the TLS hello message.";
    }
    description
      "A reusable grouping of configurations that are actively in use for networ
k devices which act as TLS servers.";
  }


  /*
  * containers
  */
  container console {
    leaf auth-mode {
      type auth-mode-type;
      description
```

```
        "The authentication mode used when administrative accounts login through
 console interface: none, password, AAA.";
    }
    leaf privilege-level {
      type uint8;
      description
        "User privilege level.";
    }
    description
      "Security configurations that are actively in use for console interface.";
  }

  container vtys {
    list vty {
      key "vty-number";
      leaf vty-number {
        type uint8;
        description
          "The number of the vty interface.";
      }
      leaf auth-mode {
        type auth-mode-type;
        mandatory true;
        description
          "The authentication mode used when administrator login through vty int
erface: none, password, AAA.";
      }
      leaf privilege-level {
        type uint8;
        mandatory true;
        description
          "User privilege level.";
      }
      leaf-list acl-name-list {
        type string;
        description
          "The name of the acl.";
      }
      leaf ip-block-enable {
        type boolean;
        mandatory true;
        description
          "The status of ip block function: enabled, or disabled.";
      }
      container ip-block-limit {
        if-feature ip-block-config;
                leaf failed-times {
          type uint64;
                  description
            "The failed times in a certain perid.";
```

```
        }
        leaf peroid {
          type uint64;
                  description
            "The certain period in which the failed times are counted.";
        }
        leaf reactive-time {
          type uint64;
          description
            "The reactive time after which the address is not blocked.";
        }
        description
          "If the login from an address failed several times in a certain period
, this address will be blocked for a certain time range.";
      }
      description
        "Security configurations that are actively in use for a vty interface.";
    }
    description
      "A list of security configurations that are actively in use for each vty i
nterface.";
  }

  container ssh {
    uses ssh-server-attribute-grouping;
    uses ssh-security-harden-grouping;
    leaf ssh-enable {
      type boolean;
      description
        "The status of SSH server: enabled, or disabled.";
    }
    leaf ip-block-enable {
      type boolean;
      description
        "The status of ip block function: enabled, or disabled.";
    }
    container ip-block-limit {
      if-feature ip-block-config;
      leaf failed-times {
        type uint64;
                description
          "The failed times in a certain perid.";
      }
      leaf peroid {
        type uint64;
                description
          "The certain period in which the failed times are counted.";
      }
      leaf reactive-time {
        type uint64;
```

```
                description
              "The reactive time after which the address is not blocked.";
          }
          description
            "If the login from an address failed several times in a certain period,
this address will be blocked for a certain time range.";
        }
        description
          "Security configurations that are actively in use for SSH-based access cha
nnel.";
      }

      container web {
        if-feature web-interface;
        uses tls-server-attribute-grouping;
        leaf auth-mode {
          type auth-mode-type;
          description
            "The authentication mode used when administrator login through web inter
face: none, password, AAA.";
        }
        leaf privilege-level {
          type uint8;
          description
            "User privilege level.";
        }
        leaf http-server-interface {
          type string;
          description
            "The source interface of web server.";
        }
        leaf https-ipv4-enable {
          type boolean;
          description
            "The status of ipv4 https server: enabled, disabled.";
        }
        leaf https-ipv6-enable {
          type boolean;
          description
            "The status of ipv6 https server: enabled, disabled.";
        }
        leaf https-source-port {
         type inet:port-number;
         description
           "The port number of web server.";
        }
        leaf https-timeout {
          type uint32;
          description
            "The authentication timeout period of https.";
        }
```

```
    leaf ip-block-enable {
      type boolean;
      description
        "The status of ip block function: enabled, or disabled.";
    }
    container ip-block-limit {
      if-feature ip-block-config;
      leaf failed-times {
        type uint64;
        description
          "The failed times in a certain perid.";
      }
      leaf peroid {
        type uint64;
        description
          "The certain period in which the failed times are counted.";
      }
      leaf reactive-time {
        type uint64;
        description
          "The reactive time after which the address is not blocked.";
      }
      description
        "If the login from an address failed several times in a certain period,
this address will be blocked for a certain time range.";
    }
    description
      "If the network device supports web interface. The configuration status of
 the web server.";
  }
}
```

6.3. Module 'ietf-aaa-security'

```
<CODE BEGINS> file "ietf-aaa-security@2018-10-16.yang"
module ietf-aaa-security {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-aaa-security";
  prefix aaasec;

  import ietf-inet-types {
    prefix inet;
      reference
        "RFC 6991 - Common YANG Data Types.";
  }

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";

  contact
```

```
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";

  description
    "This YANG module defines ietf-aaa-security YANG module, which contains conf
igurations of AAA.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  /*
  * features
  */
  feature tacacs-supported {
    description
      "Whether the device supports TACACS+ based Authentication, Authorization,
and Accounting.";
  }

  /*
  * typedefs
  */
  typedef aaa-authen-mode {
    type enumeration {
      enum "invalid" {
        description
          "Invalid authentication mode.";
      }
      enum "local" {
        description
          "Local authentication mode.";
      }
      enum "tacacs" {
        description
          "TACACS authentication mode. ";
      }
      enum "radius" {
        description
          "RADIUS authentication mode. ";
       }
```

```
      enum "none" {
        description
          "In this mode, users can pass with authentication.";
      }
      enum "radius-proxy" {
        description
          "RADIUS proxy authentication mode.";
      }
    }
    description
      "Diffrent types of authentication modes.";
  }

  typedef radius-authen-type {
    type enumeration {
      enum "pap" {
        description
          "PAP authentication.";
      }
      enum "chap" {
        description
          "CHAP authentication.";
      }
    }
    description
      "Different authentication types of RADIUS authentication.";
  }

  typedef aaa-author-mode {
    type enumeration {
      enum "invalid" {
        description
          "Invalid authorization mode.";
      }
      enum "local" {
        description
          "Local authorization mode.";
      }
      enum "tacacs" {
        description
          "TACACS authorization mode.";
      }
      enum "if-authenticated" {
        description
          "If-authenticated mode: If users pass the authentication and the authe
ntication is not in this mode, it indicates that the user authorization is passe
d. Otherwise, the authorization is not passed.";
      }
      enum "none" {
        description
```

```
            "Users can pass without authorization.";
        }
      }
      description
        "Different types of AAA authorization modes.";
    }

    typedef aaa-cmd-author-mode {
      type enumeration {
        enum "invalid" {
          description
            "Invalid command line authorization mode.";
        }
        enum "local" {
          description
            "Local command line authorization mode.";
        }
        enum "tacacs" {
          description
            "Specifies that the TACACS mode is applied.";
        }
      }
      description
            "Different types of command line authorization modes.";
    }

    typedef aaa-account-mode {
      type enumeration {
        enum "invalid" {
          description
            "invalid accounting mode.";
        }
        enum "radius" {
          description
            "RADIUS accounting mode. ";
        }
        enum "tacacs" {
          description
            "TACACS accounting mode. ";
        }
        enum "none" {
          description
            "In this mode, users do not be accounting.";
        }
      }
      description
        "Different types of accounting modes.";
    }
```

```
  /*
  * lists & containers
  */
  list authentication-scheme {
    key "authen-scheme-name";
    leaf authen-scheme-name {
      type string;
      description
        "The name of the authentication scheme.";
    }
    leaf-list authen-mode {
      type aaa-authen-mode;
      description
        "A list of authentication modes with different preference level. The sec
ond, third, and the following authentication mode is used only when the first au
thentication mode does not respond.";
    }
    leaf authen-type {
      type radius-authen-type;
      description
        "Authentication type of RADIUS: PAP, CHAP.";
    }
    leaf authen-fail-policy {
      type boolean;
      description
        "The policy to be adopted after user authentication fail: force the user
 to be offline, allow user login to a domain with access control.";
    }
    description
      "Authentication scheme list.";
  }

  list authorization-scheme {
    key "author-scheme-name";
    leaf author-scheme-name {
      type string;
      description
        "The name of the authorization scheme.";
    }
    leaf-list auhtor-mode {
      type aaa-author-mode;
      description
        "A list of authorization modes with different preference level. The seco
nd, third, and the following authorization mode is used only when the first auth
orization mode does not respond.";
    }
    leaf-list cmd-auhtor-mode {
      type aaa-cmd-author-mode;
      description
        "A list of command line authorization modes with different preference le
vel. The second, third, and the following command line authorization mode is use
d only when the first command line authorization mode does not respond.";
    }
    description
      "Authorization scheme list.";
```

```
  }

  list accounting-scheme {
    key "account-scheme-name";
    leaf account-scheme-name {
      type string;
      description
        "The name of the accounting scheme.";
    }
    leaf account-mode {
      type aaa-account-mode;
      description
        "Accounting mode.";
    }
    description
      "Accounting scheme list.";
  }

  container radius-security {
    list radius-authen-servers {
      key "address";
      leaf address {
        type inet:host;
        description
          "The ip address of the authentication server.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number of the authentication server.";
      }
      description
        "A list of RADIUS authentication servers";
    }
    list radius-author-servers {
      key "address";
      leaf address {
        type inet:host;
        description
          "The ip address of the authorization server.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number of the authorization server.";
      }
      description
        "A list of RADIUS authorization servers";
```

```
    }
    list radius-account-servers {
      key "address";
      leaf address {
        type inet:host;
        description
          "The ip address of the accounting server.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number of the accounting server.";
      }
      description
        "A list of RADIUS accounting servers";
    }
    description
      "RADIUS authentication servers, authorization servers and accounting serve
rs.";
  }

  container tacacs-security {
        if-feature tacacs-supported;
    list tacacs-authen-servers {
      key "address";
      leaf address {
        type inet:host;
        description
          "The ip address of the authentication server.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number of the authentication server.";
      }
      description
        "A list of TACACS+ and TACACS+ compatible authentication servers";
    }
    list tacacs-author-servers {
      key "address";
      leaf address {
        type inet:host;
        description
          "The ip address of the authorization server.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number of the authorization server.";
```

```
      }
      description
        "A list of TACACS+ and TACACS+ compatible authorization servers";
    }
    list tacacs-account-servers {
      key "address";
      leaf address {
        type inet:host;
        description
          "The ip address of the accounting server.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number of the accounting server.";
      }
      description
        "A list of TACACS+ and TACACS+ compatible accounting servers";
    }
    description
      "TACACS+ and TACACS+ compatible authentication servers, authorization serv
ers, and accounting servers.";
  }
}
<CODE ENDS>
```

6.4. Module 'ietf-admin-access-statistics'

```
<CODE BEGINS> file "ietf-admin-access-statistics@2018-10-16.yang"
module ietf-admin-access-statistics {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-admin-access-statistics";
  prefix stat;

  import ietf-inet-types {
    prefix inet;
      reference
        "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";
```

```
  contact
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";

  description
    "This YANG module defines ietf-admin-access-statistics YANG module, which co
ntains online administrator lists, ip addresses authentication failure or blocke
d ip addresses.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  /*
  * features
  */
  feature display-online-info {
    description
      "If the device supports reporting the details of administrative accounts t
hat are currenlty online.";
  }

  /*
  * typedef
  */
  typedef ip-block-state-type {
    type enumeration {
      enum "authenfail" {
        description
          "Authentication fialed State";
      }
      enum "blocked" {
        description
          "BLOCKED State";
      }
    }
    description
      "The status of an login failed IP address.";
  }

  /*
  * containers
```

```
  */
  container online {
    leaf total-online-users {
      type uint32;
      config false;
      description
        "The number of administrators that are current online.";
    }
    container online-admin-list {
      if-feature display-online-info;
      list online-users {
        key "account-name";
        leaf account-name {
          type string;
          description
            "The account name of the online account.";
        }
        leaf ip-address {
          type inet:ip-address-no-zone;
          config false;
                description
            "The ip address of the online account.";
        }
        leaf mac-address {
          type yang:mac-address;
          config false;
                description
            "The MAC address of the online account.";
        }
        description
          "Online adminstrator list.";
      }
      description
        "If the device supports providing information of online administrators,
a list of account details are provided.";
    }
    description
      "Online administrator statistics and details.";
  }

  container ip-block-list {
    list blocked-ip {
      key "ip-address";
      leaf ip-address {
        type inet:ip-address-no-zone;
        description
          "The blocked IP address.";
      }
      leaf vpn-instance {
```

```
        type string;
        config false;
        description
          "The VPN instance of the blocked IP address.";
          }
      leaf state {
        type ip-block-state-type;
        config false;
        description
          "The status of an login failed IP address.";
          }
      leaf authen-fail-account {
        type uint32;
        config false;
        description
          "The number of the login failed attempts.";
          }
      description
        "The list of blocked IP addresses and related information.";
    }
    description
      "The information of blocked IP addresses and related information.";
  }
}
<CODE ENDS>
```

6.5. Module 'ietf-snmp-security'

```
<CODE BEGINS> file "ietf-snmp-security@2018-10-16.yang"
module ietf-snmp-security {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-snmp-security";
  prefix snmpsec;

  import ietf-snmp {
    prefix snmp;
        reference
      "RFC 7407.";
  }

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
```

```
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";

  description
    "This YANG module defines ietf-snmp-security YANG module.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  feature tsm {
    description
      "Whether the network device supports Transport Security Model for SNMP.";
  }

  /*
  * typedef
  */
  typedef snmp-transport-type {
    type enumeration {
      enum "udp" {
        description
          "SNMP over UDP.";
      }
      enum "ssh" {
        description
          "SNMP over SSH.";
      }
      enum "tls" {
        description
           "SNMP over TLS.";
      }
      enum "dtls" {
        description
          "SNMP over DTLS.";
      }
    }
    description
      "The transport channels on which the SNMP engine listens.";
   }

  typedef snmp-version-type {
    type enumeration {
```

```
      enum "v1" {
        description
          "SNMPv1";
      }
      enum "v2c" {
        description
          "SNMPv2c";
      }
      enum "v3" {
        description
           "SNMPv3";
      }
    }
    description
      "The version of SNMP protocol";
  }

  typedef auth-pro-type {
    type enumeration {
      enum "none" {
        description
          "Do not enable the authentication of messages sent on behalf of the us
er.";
      }
      enum "md5" {
        description
          "HMAC-MD5-96 authentication protocol";
      }
      enum "sha" {
        description
          "HMAC-SHA-96 authentication protocol";
      }
    }
    description
      "An indication of whether messages sent on behalf of this user can be auth
enticated, and if so, the type of authentication protocol which is used: MD5, SH
A.";
    reference
      "RFC 3414";
  }

  typedef priv-pro-type {
    type enumeration {
      enum "none" {
        description
          "Do not enable the encryption of messages sent on behalf of the user."
;
      }
      enum "des" {
        description
          "DES is used to encrypt messages sent on behalf of the user.";
      }
```

```
      enum "aes" {
        description
          "AES is used to encrypt messages sent on behalf of the user.";
      }
    }
    description
      "An indication of whether messages sent on behalf of this user can be prot
ected from disclosure, and if so, the type of privacy protocol which is used: ED
S, AES.";
    reference
      "RFC 3414 & RFC 3826";
  }

  /*
  * grouping
  */
  grouping user-auth-priv {
    list user {
        key "name";
        leaf name {
          type snmp:identifier;
          description
            "The identifier that represents a user.";
        }
                leaf auth-protocol {
          type auth-pro-type;
          description
            "The type of authentication protocol: none, md5, sha.";
        }
                leaf priv-protocol {
          type priv-pro-type;
          description
            "The type of encryption protocol: none, des, aes.";
        }
        description
          "A list of users and their corresponding authProtocol, privProtocol.";
      }
      description
        "A grouping that represents a list of users and their corresponding auth
Protocol, privProtocol.";
      reference
        "RFC 3414";
  }

  leaf snmp-enable {
    type boolean;
    description
      "whether SNMP is used.";
  }

  /*
```

```
  * containers
  */
  container engine {
    leaf enabled {
      type boolean;
      description
        "The status of the SNMP engine: enabled, disabled.";
    }
    list listen {
      key "name";
      leaf name {
        type snmp:identifier;
        description
          "The name of a transport channel on which the SNMP engine listens.";
      }
      leaf transport {
        type snmp-transport-type;
        description
          "The transport protocol that SNMP uses.";
      }
      description
        "A list of transport channels on which the SNMP engine listens.";
    }
    leaf version {
      type snmp-version-type;
      description
        "SNMP version used by the SNMP engine.";
    }
    leaf enable-authen-traps {
      type boolean;
      description
        "Whether the SNMP entity is permitted to generate authenticationFailure
traps.";
      reference
        "RFC 3418: Management Information Base (MIB) for the Simple Network Mana
gement Protocol (SNMP) SNMPv2-MIB.snmpEnableAuthenTraps";
    }
    description
      "The security configurations for SNMP engine.";
  }

  list target {
    key name;
    leaf name {
      type snmp:identifier;
      description
        "The name identifies the target.";
    }
    leaf transport {
      type snmp-transport-type;
```

```
        description
          "The transport protocol used.";
      }
      leaf target-parmas {
        type snmp:identifier;
        description
          "Parameters for the target.";
      }
      description
        "The list of targets.";
      reference
        "RFC 3413 & RFC 7407";
    }

    list target-params {
      key name;
      leaf name {
        type snmp:identifier;
        description
          "The name identifies the target params.";
      }
      choice params {
        case usm {
          uses snmp:usm-target-params;
          description
            "Reuse the grouping defined in ietf-snmp-usm";
        }
        case tsm {
          if-feature snmp:tsm;
          uses snmp:tsm-target-params;
          description
            "Reuse the grouping defined in ietf-snmp-tsm";
        }
        description
          "The parameters specific to each security model.";
      }
      description
        "List of target parameters.";
    }

    container vacm {
      leaf vacm-enable {
        type boolean;
        config false;
        description
          "Whether VACM based security configurations are used.";
      }
      list group {
```

```
      key name;
      leaf name {
        type snmp:group-name;
        description
          "The name of this VACM group.";
      }
      list member {
        key "security-name";
        leaf security-name {
          type snmp:security-name;
          description
            "The securityName of a group member.";
        }
        leaf-list security-model {
          type snmp:security-model;
          min-elements 1;
          description
            "The security models under which this security-name is a member of t
his group.";
        }
        description
          "A member of this VACM group.";
      }
      list access {
        key "context security-model security-level";
        leaf context {
          type snmp:context-name;
          description
            "The context under which the access rights apply.";
        }
        leaf context-match {
          type enumeration {
            enum exact {
              value 1;
              description
                "The context match type: exact.";
            }
            enum prefix {
              value 2;
              description
                "The context match type: prefix";
            }
          }
          description
            "The match type of the context.";
        }
        leaf security-model {
          type snmp:security-model-or-any;
          description
```

```
            "The security model under which the access rights apply.";
        }
        leaf security-level {
          type snmp:security-level;
          description
            "The minimum security level under which the access rights apply.";
        }
        leaf read-view {
          type snmp:view-name;
          description
            "The name of the MIB view of the SNMP context authorizing read acces
s. If this leaf does not exist in a configuration, it maps to a zero-length vacm
AccessReadViewName.";
        }
        leaf wirte-view {
          type snmp:view-name;
          description
            "The name of the MIB view of the SNMP context authorizing write acce
ss. If this leaf does not exist in a configuration, it maps to a zero-length vac
mAccessWriteViewName.";
        }
        leaf notify-view {
          type snmp:view-name;
          description
            "The name of the MIB view of the SNMP context authorizing notify acc
ess. If this leaf does not exist in a configuration, it maps to a zero-length va
cmAccessNotifyViewName.";
        }
        description
          "Definition of access right for groups.";
      }
      description
        "VACM groups";
      reference
        "RFC 3415: View-based Access Control Model (VACM) for the Simple Network
 Management Protocol (SNMP)";
    }
    list view {
      key name;
      leaf name {
        type snmp:view-name;
        description
          "The name of this MIB view.";
      }
      leaf-list include {
        type snmp:wildcard-object-identifier;
        description
          "A family of subtrees included in this MIB view.";
      }
      leaf-list exclude {
        type snmp:wildcard-object-identifier;
        description
          "A family of subtrees excluded in this MIB view.";
      }
      description
```

```
        "Definition of MIB views.";
    }
    description
      "The security configurations for View-based Access Control Model (VACM).";
  }

  container usm {
    leaf usm-enable {
      type boolean;
      config false;
      description
        "Whether USM based security configurations are used.";
    }
    container local {
      uses user-auth-priv;
      description
        "A list of local users and their corresponding authentication and privac
y protocols.";
    }
    container remote {
      uses user-auth-priv;
      description
        "A list of remote users and their corresponding authentication and priva
cy protocols.";
    }
    description
      "Configuration of the User-based Security Model.";
  }

  container tsm {
    if-feature tsm;
    leaf tsm-enable {
      type boolean;
      config false;
      description
        "Whether TSM based security configurations are used.";
    }
    description
      "Configuration of Transport Security Model.";
  }
}
<CODE ENDS>
```

6.6. Module 'ietf-netconf-security'

```
module ietf-netconf-security {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-security";
  prefix netsec;
```

```
  import ietf-admin-access-security {
    prefix accsec;
  }

  import ietf-inet-types {
    prefix inet;
        reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";

  description
    "This YANG module defines ietf-netconf-security YANG module.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  /*
  * features
  */
  feature listen {
    description
      "The 'listen' feature indicates that the NETCONF server supports opening a
 port to accept NETCONF client connections using at least one transport (e.g., S
SH, TLS, etc.).";
  }

  feature ssh-listen {
    description
      "The 'ssh-listen' feature indicates that the NETCONF server supports openi
ng a port to accept NETCONF over SSH client connections.";
    reference
      "RFC 6242: Using the NETCONF Protocol over Secure Shell (SSH)";
  }
```

```
  feature tls-listen {
    description
      "The 'tls-listen' feature indicates that the NETCONF server supports openi
ng a port to accept NETCONF over TLS client connections.";
    reference
      "RFC 7589: Using the NETCONF Protocol over Transport Layer Security (TLS)
with Mutual X.509 Authentication";
  }

  feature call-home {
    description
      "The 'call-home' feature indicates that the NETCONF server supports initia
ting NETCONF call home connections to NETCONF clients using at least one transpo
rt (e.g., SSH, TLS, etc.).";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  feature ssh-call-home {
    description
      "The 'ssh-call-home' feature indicates that the NETCONF server supports in
itiating a NETCONF over SSH call home connection to NETCONF clients.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  feature tls-call-home {
    description
      "The 'tls-call-home' feature indicates that the NETCONF server supports in
itiating a NETCONF over TLS call home connection to NETCONF clients.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  /*
  * leaf & containers
  */

  leaf netconf-enable {
    type boolean;
    description
      "Whether the NETCONF protocol is used.";
  }

  container listen {
    if-feature listen;
    list endpoint {
      key name;
      leaf name {
        type string;
        description
          "The name of the NETCONF listen endpoint.";
      }
      choice transport {
```

```
        case ssh {
          if-feature ssh-listen;
          leaf port {
            type inet:port-number;
            description
              "The local port number to listen on.";
          }
          uses accsec:ssh-server-attribute-grouping;
          description
            "SSH based listening.";
        }
        case tls {
          if-feature tls-listen;
          leaf port {
            type inet:port-number;
            description
              "The local port number to listen on.";
          }
          uses accsec:tls-server-attribute-grouping;
          description
            "TLS based listening.";
        }
        description
          "The transport protocol used.";
      }
      description
        "List of endpoints to listen for NETCONF connections.";
    }
    description
      "Configurations related the listen behavior.";
  }

  container call-home {
    if-feature call-home;
    list netconf-client {
      key name;
      leaf name {
        type string;
        description
          "The name of the remote NETCONF client.";
      }
      container endpoints {
        list endpoint {
          key name;
          leaf name {
            type string;
            description
              "The name for this endpoint.";
```

```
          }
          choice transport {
            case ssh {
              if-feature ssh-call-home;
              leaf port {
                type inet:port-number;
                description
                  "The IP port for this endpoint.";
              }
              uses accsec:ssh-server-attribute-grouping;
              description
                "SSH based call-home.";
            }
            case tls {
              if-feature tls-call-home;
              leaf port {
                type inet:port-number;
                description
                  "The IP port for this endpoint.";
              }
              uses accsec:tls-server-attribute-grouping;
              description
                "TLS based call-home.";
            }
            description
              "The used transport protocol.";
          }
          description
            "A list of endpoints for this NETCONF server to try to connect in se
quence.";
        }
        description
          "List of endpoints";
      }
      description
        "List of NETCONF clients the NETCONF server is to initiate call-home con
nections to in parallel.";
    }
    description
      "Configurations related to call-home behavior.";
  }
}
```

6.7. Module 'ietf-port-management-security'

```
<CODE BEGINS> file "ietf-port-management-security@2018-10-16.yang"
module ietf-port-management-security {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-port-management-security";
  prefix acsec;
```

```
  import ietf-inet-types {
    prefix inet;
        reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF SACM (Security Automation and Continuous Monitoring) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/sacm/
    WG List: sacm@ietf.org

    Editor: Qiushi Lin
            linqiushi@huawei.com;
    Editor: Liang Xia
            frank.xialiang@huawei.com
        Editor: Henk Birkholz
            henk.birkholz@sit.fraunhofer.de";

  description
    "This YANG module defines ietf-port-management-security YANG module.";

  revision 2018-10-16 {
    description "Initial version.";
        reference
      "draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Inf
rastructure Device Management Plane Security Baseline";
  }

  list port-list {
    key port-number;
    leaf port-number {
      type inet:port-number;
      description
        "The port number.";
    }
    leaf port-status {
      type boolean;
      description
        "The status of the port: open or shut-down.";
    }
    description
      "The status of all the ports in the device.";
  }
}
<CODE ENDS>
```

7. Acknowledgements

8. IANA Considerations

This document requires no IANA actions.

9. Security Considerations

Secure transport should be used to retrieve the current status of management plane security baseline.

10. References

10.1. Normative References

[I-D.birkholz-sacm-yang-content]
Birkholz, H. and N. Cam-Winget, "YANG subscribed notifications via SACM Statements", draft-birkholz-sacm-yang-content-01 (work in progress), January 2018.

[I-D.dong-sacm-nid-cp-security-baseline]
Dong, Y. and L. Xia, "The Data Model of Network Infrastructure Device Control Plane Security Baseline", draft-dong-sacm-nid-cp-security-baseline-00 (work in progress), September 2017.

[I-D.dong-sacm-nid-infra-security-baseline]
Dong, Y. and L. Xia, "The Data Model of Network Infrastructure Device Infrastructure Layer Security Baseline", draft-dong-sacm-nid-infra-security-baseline-01 (work in progress), May 2018.

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-06 (work in progress), September 2018.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-07 (work in progress), September 2018.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", draft-ietf-netconf-ssh-client-server-07 (work in progress), September 2018.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-07 (work in progress), September 2018.

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-20 (work in progress), October 2018.

[I-D.ietf-netmod-syslog-model]
Wildes, C. and K. Koushik, "A YANG Data Model for Syslog Configuration", draft-ietf-netmod-syslog-model-26 (work in progress), March 2018.

[I-D.ietf-sacm-information-model]
Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus, M., Haynes, D., and H. Birkholz, "SACM Information Model", draft-ietf-sacm-information-model-10 (work in progress), April 2017.

[I-D.xia-sacm-nid-dp-security-baseline]
Xia, L. and G. Zheng, "The Data Model of Network Infrastructure Device Data Plane Security Baseline", draft-xia-sacm-nid-dp-security-baseline-02 (work in progress), June 2018.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

10.2. Informative References

[I-D.ietf-tls-oldversions-deprecate]
Moriarty, K. and S. Farrell, "Deprecating TLSv1.0 and TLSv1.1", draft-ietf-tls-oldversions-deprecate-00 (work in progress), September 2018.

[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>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A.

The following is the whole structure of the YANG tree diagram for network infrastructure device management plane. The existed RFCs and drafts that related this document are listed at the right side.

Modules	Related RFCs/Drafts
ietf-admin-account-security	None
ietf-admin-access-security	draft-ietf-netconf-keystore,draft-ietf-netconf-ssh-client-server,draft-ietf-netconf-tls-client-server
ietf-aaa-security	RFC7317
ietf-admin-access-statistics	None
ietf-snmp-security	RFC7407
ietf-netconf-security	draft-ietf-netconf-netconf-client-server,draft-ietf-netconf-keystore
ietf-port-management-security	None
ietf-log-security	draft-ietf-netmod-syslog-model
ietf-file-security	draft-ietf-netconf-keystore,draft-ietf-netconf-ssh-client-server,draft-ietf-netconf-tls-client-server

The modules defined in this document and related RFCs/drafts

Draft [I-D.ietf-netconf-tls-client-server] and draft [I-D.ietf-netconf-ssh-client-server] focus on YANG models for TLS-specific configuration and SSH-specific configuration respectively. The transport-level configuration, such as what ports to listen-on or connect-to, is not included. Besides, as these grouping focus on configurations, the configuration of private-key and "certificate-expiration" notification are not needed. Draft [I-D.ietf-netconf-netconf-client-server] defines NETCONF YANG model based on the data models defined in the above two documents.

[RFC7317] defines a YANG data model for system management of device containing a NETCONF sever. It summarizes data modules for NETCONF user authentication, and defined YANG module for client to configure the RADIUS authentication server information. Three methods are defined for user authentication: public key for local users over SSH, password for local users over any secure transport, password for RADIUS users over any secure transport.

[RFC7407] defines a YANG model for SNMP configuration it is not limited security related configurations and status.

Draft [I-D.ietf-netmod-syslog-model] defines a YANG model for Syslog configuration, including TLS based transport security and syslog messages signing.

Authors' Addresses

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

Email: linqiushi@huawei.com

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

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:

- o [HACK99]: TODO: Provide description.
- o [HACK100]: TODO: Provide description.
- o [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:

- o What are the specific components participating in a SACM Domain?
- o 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)?
- o 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?
- o Should workflows be documented in this draft or separate drafts?
- o 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.

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

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

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

- o 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

- o 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.

- o 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:

- o Capability/operation semantics
- o SACM Component implementation identifiers
- o SACM Component versions
- o 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

Network Working Group L. Xia
Internet-Draft G. Zheng
Intended status: Standards Track W. Pan
Expires: April 25, 2019 Huawei
October 22, 2018

The Data Model of Network Infrastructure Device Data Plane Security Baseline
draft-xia-sacm-nid-dp-security-baseline-03

Abstract

This document proposes one part of the security baseline YANG for network infrastructure device (i.e., router, switch, firewall, etc.): data plane security baseline. The companion documents [I-D.ietf-lin-sacm-nid-mp-security-baseline], [I- D.ietf-dong-sacm-nid-infra-security-baseline] cover other parts of the security baseline YANG for network infrastructure device respectively: management plane security baseline, infrastructure layer security baseline.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Objective

Network security is an essential part of the overall network deployment and operation. Due to the following reasons, network infrastructure devices (e.g. switch, router, firewall) are always the objective and exploited by the network attackers, which bring damages to the victim network:

- o The existence of a lot of unsafe access channels: for the history reason, some old and unsafe protocols still run in the network devices, like: SNMP v1/v2, Telnet, etc., and are not mandatory to be replaced by the according safer protocols (SNMP v3, SSH). Attackers easily exploit them for attack (e.g., invalid login, message eavesdropping);

- o The openness nature of TCP/IP network: despite the benefits of network architecture design and connectivity brought by the network openness, a lot of threats exist at the same time. Spoofing address, security weakness for various protocols, traffic flooding, and other kinds of threat are originated from the network openness;

- o The security challenge by the network complexity: network are becoming more complex, with massive nodes, various protocols and flexible topology. Without careful design and strict management, as well as operation automation, the policy consistency of network security management cannot be ensured. It’s common that part of the network infrastructure is subject to attack;

- o The complex functionality of device: the complexity of device itself increases the difficulty of carrying out the security hardening measurements, as well as the skill requirements to the network administrator. As a result, the network administrator may not be capable of or willing to realize all the security measurements, in addition to implementing the other basic functionalities;

- o The capacity and capability mismatching between the data plane and the control plane: there are a large mismatching of the traffic processing capacity and capability between different planes. Without effective control, the large volume of traffic from the data plane will flooding attack the other planes easily.

Therefore, the importance of ensuring the security of the network infrastructure devices is out of question. To secure the network infrastructure devices, one important task is to identify as far as possible the threats and vulnerabilities in the device itself, such as: unnecessary services, insecure configurations, abnormal status, etc., then enforce the corresponding security hardening measurements, such as: update the patch, modify the security configuration, enhance the security mechanism, etc.. We call this task the developing and deploying the security baseline for the network infrastructure, which provides a solid foundation for the overall network security. This document aims to describe the security baseline for the network infrastructure, which is called security baseline in short in this document.

1.2. Security Baseline

Basically, security baseline can be designed and deployed into different layers of the devices:

- o application layer: refers to the application platform security solution and the typical application security mechanisms it provided like: identity authentication, access control, permission management, encryption and decryption, auditing and tracking, privacy protection, to ensure secure application data transmission/exchange, secure storage, secure processing, ensuring the secure operation of the application system. Specific examples may be: web application security, software integrity protection, encryption of sensitive data, privacy protection, and lawful interception interfaces and secure third-party component;

- o network layer: refers to a series of security measures, to protect the network resources and network services running on the device network platform. Network layer security over network product is complicated. Therefore, it is divided into data plane, control plane, management plane to consider:

  * data plane: focus on the security hardening configuration and status to protect the data plane traffic against eavesdropping, tampering, forging and flooding attacking the network;

  * control plane: focus on the control signaling security of the network infrastructure device, to protect their normal exchange against various attacks (i.e., eavesdropping, tampering, forging and flooding attack) and restrict the malicious control signaling, for ensuring the correct network topology and forwarding behavior;

  * management plane: focus on the management information and platform security. More specific, it includes all the security configuration and status involved in the network OAM process;

- o infrastructure layer: refers to all the security design about the device itself and its running OS. As the foundation of the upper layer services, the secure infrastructure layer must be assured. The specific mechanisms include: OS security, key management, cryptography security, certificate management, software integrity.

1.3. Security Baseline Data Model Design

The security baseline varies according to many factors, like: different device types (i.e., router, switch, firewall), the supporting security features of device, the specific security requirements of network operator. It’s impossible to design a complete set for it, so this document and the companion ones are going to propose the most important and universal points of them. More baseline contents can be added in future following the data model scheme specified.

[I-D.ietf-birkholz-sacm-yang-content] defines a method of constructing the YANG data model scheme for the security posture assessment of the network infrastructure device by brokering of YANG push telemetry via SACM statements. The basic steps are:

- use YANG push mechanism[I-D.ietf-netconf-yang-push]to collect the created streams of notifications (telemetry) [I-D.ietf-netconf-subscribed-notifications]providing SACM content on SACM data plane, and the filter expressions used in the context of YANG subscriptions constitute SACM content that is imperative guidance consumed by SACM components on SACM management plane;

- then encapsulate the above YANG push output into a SACM Content Element envelope, which is again encapsulated in a SACM statement envelope;

- lastly, publish the SACM statement into a SACM domain via xmpp-grid publisher.

In this document, we follow the same way as [I-D.ietf-birkholz-sacm-yang-content] to define the YANG output for network infrastructure device security baseline posture based on the SACM information model definition [I-D.ietf-sacm-information-model].

1.4. Summary

The following contents propose part of the security baseline YANG output for network infrastructure device: data plane security baseline. The companion documents [I-D.ietf- dong-sacm-nid-cp-security-baseline], [I-D.ietf-lin-sacm-nid-mp-security-baseline], [I-D.ietf-xia-sacm-nid-app-infr-layers-security-baseline] cover other parts of the security baseline YANG output for network infrastructure device respectively: control plane security baseline, management plane security baseline, application layer and infrastructure layer security baseline.

2. Terminology

2.1. Key Words

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

2.2. Definition of Terms

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

3. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. Data Model Structure

As the network infrastructure device, it makes decision of the forwarding path based on the IP/MAC address and sends the packet in data plane. The NP or ASIC are the main components for the data plane functions.

This section describes the key data plane security baseline of the network infrastructure devices, and defines their specific data models.

4.1. Layer 2 protection

Mac table is the key resource in terms of layer 2 forwarding, also easily attacked by learning massive invalid mac address. The mac limit function is to protect the mac table by limiting the maximum number of learned mac address in appointed interfaces. The mac address is not learned and the packet is discarded when the up-limit is reached, and the alarm is created possibly.

If the broadcast traffic is not suppressed in layer 2 network (i.e., Ethernet), a great amount of network bandwidth is consumed by a great deal of broadcast traffic. The network performance is degraded, even interrupting the communication. In such a case, configuring the broadcast traffic suppression on the device to ensure some bandwidth can be reserved for unicast traffic forwarding when broadcast traffic bursts across the network. It's flexible to configure the device to suppress broadcast, multicast, and unknown unicast traffic on an interface, a specified interface in a VLAN, a sub-interface, and over a virtual switch instance (VSI) pseudo wire (PW).

```
module: ietf-layer2-protection
    +--rw mac-limit
    |  +--rw vlan-mac-limits
    |  |  +--rw vlan-mac-limit* [vlan-id]
    |  |     +--rw vlan-id       mac-vlan-id
    |  |     +--rw maximum       uint32
    |  |     +--rw rate?         uint32
    |  |     +--rw action?       mac-limit-forward
    |  |     +--rw alarm?        mac-enable-status
    |  +--rw bd-mac-limits
    |  |  +--rw bd-mac-limit* [bd-id]
    |  |     +--rw bd-id         uint32
    |  |     +--rw maximum       uint32
    |  |     +--rw rate?         uint32
    |  |     +--rw action?       mac-limit-forward
    |  |     +--rw alarm?        mac-enable-status
    |  +--rw vsi-mac-limits
    |  |  +--rw vsi-mac-limit* [vsi-name]
    |  |     +--rw vsi-name      string
    |  |     +--rw maximum       uint32
    |  |     +--rw rate?         uint32
    |  |     +--rw action?       mac-limit-forward
    |  |     +--rw alarm?        mac-enable-status
    |  +--rw pw-mac-limits
    |  |  +--rw pw-mac-limit* [vsi-name pw-name]
    |  |     +--rw vsi-name      string
    |  |     +--rw pw-name       string
    |  |     +--rw maximum       uint32
    |  |     +--rw rate?         uint32
    |  |     +--rw action?       mac-limit-forward
    |  |     +--rw alarm?        mac-enable-status
    |  +--rw if-mac-limits
    |  |  +--rw if-mac-limit* [if-name]
    |  |     +--rw if-name       string
    |  |     +--rw maximum       uint32
    |  |     +--rw rate?         uint32
    |  |     +--rw action?       mac-limit-forward
    |  |     +--rw alarm?        mac-enable-status
    |  +--rw subif-mac-limits
    |     +--rw subif-mac-limit* [if-name]
```

```
   |        +--rw if-name       string
   |        +--rw maximum       uint32
   |        +--rw rate?         uint32
   |        +--rw action?       mac-limit-forward
   |        +--rw alarm?        mac-enable-status
   +--rw traffic-suppress
      +--rw vlan-suppresses
      |  +--rw vlan-suppress* [vlan-id suppress-type direction]
      |     +--rw vlan-id          mac-vlan-id
      |     +--rw suppress-type    suppress-type
      |     +--rw direction        direction-type
      |     +--rw cir?             uint64
      |     +--rw cbs?             uint64
      +--rw bd-suppresses
      |  +--rw bd-suppress* [bd-id suppress-type direction]
      |     +--rw bd-id            uint32
      |     +--rw suppress-type    suppress-type
      |     +--rw direction        direction-type
      |     +--rw cir?             uint64
      |     +--rw cbs?             uint64
      +--rw vsi-suppresses
      |  +--rw vsi-suppress* [vsi-name suppress-type direction]
      |     +--rw vsi-name         string
      |     +--rw suppress-type    suppress-type
      |     +--rw direction        direction-type
      |     +--rw cir?             uint64
      |     +--rw cbs?             uint64
      +--rw pw-suppresses
      |  +--rw pw-suppress* [vsi-name pw-name suppress-type direction]
      |     +--rw vsi-name         string
      |     +--rw pw-name          string
      |     +--rw suppress-type    suppress-type
      |     +--rw direction        direction-type
      |     +--rw cir?             uint64
      |     +--rw cbs?             uint64
      +--rw if-suppresses
      |  +--rw if-suppress* [if-name suppress-type direction]
      |     +--rw if-name          string
      |     +--rw suppress-type    suppress-type
      |     +--rw direction        direction-type
      |     +--rw percent?         uint64
      |     +--rw packets?         uint64
      |     +--rw cir?             uint64
      |     +--rw cbs?             uint64
      +--rw sub-if-suppresses
      |  +--rw sub-if-suppress* [if-name suppress-type direction]
      |     +--rw if-name          string
      |     +--rw suppress-type    suppress-type
```

```
       |      +--rw direction        direction-type
       |      +--rw cir?             uint64
       |      +--rw cbs?             uint64
       +--rw if-storm-controls
          +--rw if-storm-control* [if-name]
             +--rw if-name                         string
             +--rw action?                         storm-ctrl-action-type
             +--rw trap-enable?                    enable-type
             +--rw log-enable?                     enable-type
             +--rw interval?                       uint64
             +--rw if-packet-control-rules
             |  +--rw if-packet-control-rule* [packet-type]
             |     +--rw packet-type               storm-ctrl-type
             |     +--rw rate-type?                storm-ctrl-rate-type
             |     +--rw min-rate                  uint64
             |     +--rw max-rate                  uint64
             +--ro if-storm-control-infos
                +--ro ifstorm-control-info* [packet-type]
                   +--ro packet-type               storm-ctrl-type
                   +--ro punish-status?            storm-ctrl-action-type
                   +--ro last-punish-time?         string
```

4.2. ARP

ARP security is a set of functions to protect the ARP protocol and networks against malicious attacks so that the network communication keeps stable and important user information is protected, which mainly includes:

ARP anti-spoofing functions: protect devices against spoofing ARP attack packets, improving the security and reliability of network communication.

ARP anti-flooding functions: relieve CPU load and prevent the ARP table overflow, ensuring normal network operation.

```
module: ietf-arp-sec
    +--rw arp-sec
       +--rw arp-packet-validate
       |  +--rw global-validate-type                    arp-validate-type
       |  +--rw if-validate-rule* [if-name]
       |     +--rw if-name                              string
       |     +--rw validate-type                        arp-validate-type
       +--rw arp-gratuitous-control
       |  +--rw global-send-gratuitous-enable           boolean
       |  +--rw global-receive-gratuitous-enable        boolean
       |  +--rw if-gratuitous-rule* [if-name]
       |     +--rw if-name              string
```

```
     |     +--rw send-gratuitous-enable              boolean
     |     +--rw receive-gratuitous-enable           boolean
     +--rw arp-learning-control
     |  +--rw global-strict-learning-enable          boolean
     |  +--rw if-learning-rule* [if-name]
     |     +--rw if-name                             string
     |     +--rw learning-disable                    boolean
     |     +--rw strict-learning-enable              boolean
     +--rw arp-entry-limit
     |  +--rw if-limit-rule* [if-name]
     |  |  +--rw if-name                             string
     |  |  +--rw entry-maximum                       uint32
     |  +--rw if-vlan-limit-rule* [if-name vlan-begin]
     |     +--rw if-name                             string
     |     +--rw vlan-begin                          mac-vlan-id
     |     +--rw vlan-end                            mac-vlan-id
     |     +--rw entry-maximum                       uint32
     +--rw arp-rate-limit
     |  +--rw global-rate-limit                      uint32
     |  +--rw limit-by-source-mac
     |  |  +--rw rate-limit-per-source-mac           uint32
     |  |  +--rw source-mac-rule* [source-mac]
     |  |     +--rw source-mac                       mac-address
     |  |     +--rw rate-limit                       uint32
     |  +--rw limit-by-source-ip
     |  |  +--rw rate-limit-per-source-ip            uint32
     |  |  +--rw source-ip-rule* [source-ip]
     |  |     +--rw source-ip                        inet:ip-address
     |  |     +--rw rate-limit                       uint32
     |  +--rw limit-by-destination-ip
     |  |  +--rw rate-limit-per-destination-ip       uint32
     |  +--rw limit-by-interface
     |  |  +--rw rate-limit-per-interface            uint32
     |  |  +--rw interface-rule* [if-name]
     |  |     +--rw if-name                          string
     |  |     +--rw rate-limit                       uint32
     |  +--rw limit-by-vlan
     |     +--rw vlan-rule* [vlan-id]
     |        +--rw vlan-id                          mac-vlan-id
     |        +--rw rate-limit                       uint32
     +--rw arp-miss-rate-limit
     |  +--rw global-rate-limit                      uint32
     |  +--rw limit-by-source-ip
     |  |  +--rw rate-limit-per-source-ip            uint32
     |  |  +--rw source-ip-rule* [source-ip]
     |  |     +--rw source-ip                        inet:ip-address
     |  |     +--rw rate-limit                       uint32
     |  +--rw limit-by-interface
```

```
       |  |  +--rw rate-limit-per-interface           uint32
       |  |  +--rw interface-rule* [if-name]
       |  |     +--rw if-name                         string
       |  |     +--rw rate-limit                      uint32
       |  +--rw limit-by-vlan
       |     +--rw vlan-rule* [vlan-id]
       |        +--rw vlan-id                         mac-vlan-id
       |        +--rw rate-limit                      uint32
       +--ro sec-dis-arp-chks
          +--ro sec-dis-arp-chk* [slot-id check-type]
             +--ro slot-id                            string
             +--ro check-type                         arp-attack-type
             +--ro total-packets?                     uint64
             +--ro passed-packets?                    uint64
             +--ro dropped-packets?                   uint64
```

4.3. URPF

Unicast Reverse Path Forwarding (URPF) is a technology used to defend against network attacks based on source address spoofing. Generally, upon receiving a packet, a router first obtains the destination IP address of the packet and then searches the forwarding table for a route to the destination address. If the router finds such a route, it forwards the packet; otherwise, it discards the packet. A URPF-enabled router, however, obtains the source IP address of a received packet and searches for a route to the source address. If the router fails to find the route, it considers that the source address is a forged one and discards the packet. In this manner, URPF can effectively protect against malicious attacks that are launched by changing the source addresses of packets.

URPF can be performed in strict or loose mode. The strict mode checks both the existence of source address in the route table and the interface consistency, while loose mode only checks if the source address is in the route table. In some case, the router may have only one default route to the router of the ISP. Therefore, matching the default route entry needs to be supported.

URPF can be performed over interface, defined flow and traffic sent to local CPU.

```
module: ietf-urpf-sec
    +--rw urpf-sec
       +--rw interface-urpf
       |  +--rw interface-rule* [if-name]
       |     +--rw if-name                                   string
       |     +--rw urpf-mode                                 urpf-mode-type
       |     +--rw allow-default                             boolean
       +--rw flow-urpf
       augment /policy:policies/policy:policy-entry +
               /policy:classifier-entry +
               /policy:classifier-action-entry-cfg +
               /policy:action-cfg-params:
       |  +--:(urpf)
       |     +--rw urpf-cfg
       |        +--rw urpf-mode                              urpf-mode-type
       |        +--rw allow-default                          boolean
       +--rw local-urpf
          +--rw slot-rule* [slot-id]
             +--rw slot-id                                   string
             +--rw urpf-mode                                 urpf-mode-type
             +--rw allow-default                             boolean
```

4.4. DHCP Snooping

DHCP, which is widely used on networks, dynamically assigns IP addresses to clients and manages configuration information in a centralized manner. During DHCP packet forwarding, some attacks may occur, such as bogus DHCP server attacks, DHCP exhaustion attacks, denial of service (DoS) attacks, and DHCP flooding attacks.

DHCP snooping is a DHCP security feature that functions in a similar way to a firewall between DHCP clients and servers. A DHCP-snooping-capable device intercepts DHCP packets and uses information carried in the packets to create a DHCP snooping binding table. This table records hosts' MAC addresses, IP addresses, IP address lease time, VLAN, and interface information. The device uses this table to check the validity of received DHCP packets. If a DHCP packet does not match any entry in this table, the device discards the packet.

Besides the binding table, DHCP snooping has other security features such as trusted interface, max DHCP user limit and whitelist to defend against the bogus DHCP server, DHCP flooding and other fine-grained DHCP attacks.

```
module: ietf-dhcp-snooping
    +--rw dhcp-snooping
       +--rw dhcp-snooping-enable
       |  +--rw global-enable                              boolean
```

```
       |  +--rw enable-vlan* [vlan-id]
       |  |  +--rw vlan-id                                 uint16
       |  |  +--rw dhcp-snp-enable                         boolean
       |  +--rw enable-vlan-interface* [vlan-id if-name]
       |  |  +--rw vlan-id                                 uint16
       |  |  +--rw if-name                                 string
       |  |  +--rw dhcp-snp-enable                         boolean
       |  +--rw enable-interface* [if-name]
       |  |  +--rw if-name                                 string
       |  |  +--rw dhcp-snp-enable                         boolean
       |  +--rw enable-bd* [bd-id]
       |     +--rw bd-id                                   uint32
       |     +--rw dhcp-snp-enable                         boolean
       +--rw dhcp-snooping-trust
       |  +--rw trust-vlan* [vlan-id]
       |  |  +--rw vlan-id                                 uint16
       |  |  +--rw dhcp-snp-trust                          boolean
       |  |  +--rw untrust-reply-alarm-enable              boolean
       |  |  +--rw untrust-reply-alarm-threshold           uint32
       |  +--rw trust-vlan-interface* [vlan-id if-name]
       |  |  +--rw vlan-id                                 uint16
       |  |  +--rw if-name                                 string
       |  |  +--rw dhcp-snp-trust                          boolean
       |  |  +--rw untrust-reply-alarm-enable              boolean
       |  |  +--rw untrust-reply-alarm-threshold           uint32
       |  +--rw trust-interface* [if-name]
       |  |  +--rw if-name                                 string
       |  |  +--rw dhcp-snp-trust                          boolean
       |  |  +--rw untrust-reply-alarm-enable              boolean
       |  |  +--rw untrust-reply-alarm-threshold           uint32
       |  +--rw trust-bd* [bd-id]
       |     +--rw bd-id                                   uint32
       |     +--rw dhcp-snp-trust                          boolean
       |     +--rw untrust-reply-alarm-enable              boolean
       |     +--rw untrust-reply-alarm-threshold           uint32
       +--rw dhcp-snooping-packet-check
       |  +--rw check-vlan* [vlan-id]
       |  |  +--rw vlan-id                                 uint16
       |  |  +--rw check-vlan-rule* [check-type]
       |  |     +--rw check-type                           check-type
       |  |     +--rw check-enable                         boolean
       |  |     +--rw alarm-enable                         boolean
       |  |     +--rw alarm-threshold                      uint32
       |  +--rw check-vlan-interface* [vlan-id if-name]
       |  |  +--rw vlan-id                                 uint16
       |  |  +--rw if-name                                 string
       |  |  +--rw check-vlan-if-rule* [check-type]
       |  |     +--rw check-type                           check-type
```

```
     |  |     +--rw check-enable                       boolean
     |  |     +--rw alarm-enable                       boolean
     |  |     +--rw alarm-threshold                    uint32
     |  +--rw check-interface* [if-name]
     |  |  +--rw if-name                               string
     |  |  +--rw check-if-rule* [check-type]
     |  |     +--rw check-type                         check-type
     |  |     +--rw check-enable                       boolean
     |  |     +--rw alarm-enable                       boolean
     |  |     +--rw alarm-threshold                    uint32
     |  +--rw check-bd* [bd-id]
     |     +--rw bd-id                                 uint32
     |     +--rw check-bd-rule* [check-type]
     |        +--rw check-type                         check-type
     |        +--rw check-enable                       boolean
     |        +--rw alarm-enable                       boolean
     |        +--rw alarm-threshold                    uint32
     +--rw dhcp-snooping-max-user-limit
     |  +--rw user-limit-vlan* [vlan-id]
     |  |  +--rw vlan-id                               uint16
     |  |  +--rw max-user-limit                        uint32
     |  |  +--rw alarm-enable                          boolean
     |  |  +--rw alarm-threshold                       uint32
     |  +--rw user-limit-vlan-interface* [vlan-id if-name]
     |  |  +--rw vlan-id                               uint16
     |  |  +--rw if-name                               string
     |  |  +--rw max-user-limit                        uint32
     |  |  +--rw alarm-enable                          boolean
     |  |  +--rw alarm-threshold                       uint32
     |  +--rw user-limit-interface* [if-name]
     |  |  +--rw if-name                               string
     |  |  +--rw max-user-limit                        uint32
     |  |  +--rw alarm-enable                          boolean
     |  |  +--rw alarm-threshold                       uint32
     |  +--rw user-limit-bd* [bd-id]
     |     +--rw bd-id                                 uint32
     |     +--rw max-user-limit                        uint32
     |     +--rw alarm-enable                          boolean
     |     +--rw alarm-threshold                       uint32
     +--rw dhcp-snooping-rate-limit
     |  +--rw global-check-enable                      boolean
     |  +--rw global-rate-limit                        uint32
     |  +--rw global-alarm-enable                      boolean
     |  +--rw global-alarm-threshold                   uint32
     |  +--rw rate-limit-vlan* [vlan-id]
     |  |  +--rw vlan-id                               uint16
     |  |  +--rw check-enable                          boolean
     |  |  +--rw rate-limit                            uint32
```

```
     |  |  +--rw alarm-enable                        boolean
     |  |  +--rw alarm-threshold                     uint32
     |  +--rw rate-limit-vlan-interface* [vlan-id if-name]
     |  |  +--rw vlan-id                             uint16
     |  |  +--rw if-name                             string
     |  |  +--rw check-enable                        boolean
     |  |  +--rw rate-limit                          uint32
     |  |  +--rw alarm-enable                        boolean
     |  |  +--rw alarm-threshold                     uint32
     |  +--rw rate-limit-interface* [if-name]
     |  |  +--rw if-name                             string
     |  |  +--rw check-enable                        boolean
     |  |  +--rw rate-limit                          uint32
     |  |  +--rw alarm-enable                        boolean
     |  |  +--rw alarm-threshold                     uint32
     |  +--rw rate-limit-bd* [bd-id]
     |     +--rw bd-id                               uint32
     |     +--rw check-enable                        boolean
     |     +--rw rate-limit                          uint32
     |     +--rw alarm-enable                        boolean
     |     +--rw alarm-threshold                     uint32
     +--rw dhcp-snooping-static-binding-table
     |  +--rw vlan-static-bind-tbl* [vlan-id ip-address ce-vlan]
     |  |  +--rw vlan-id                             uint16
     |  |  +--rw ip-address                          inet:ip-address
     |  |  +--rw mac-address?                        mac-address
     |  |  +--rw if-name?                            string
     |  |  +--rw ce-vlan                             uint16
     |  +--rw if-static-bind-tbl* [if-name ip-address pe-vlan ce-vlan]
     |  |  +--rw if-name                             string
     |  |  +--rw ip-address                          inet:ip-address
     |  |  +--rw mac-address?                        mac-address
     |  |  +--rw pe-vlan                             uint16
     |  |  +--rw ce-vlan                             uint16
     |  +--rw bd-static-bind-tbl* [bd-id ip-address pe-vlan ce-vlan]
     |     +--rw bd-id                               uint32
     |     +--rw ip-address                          inet:ip-address
     |     +--rw mac-address?                        mac-address
     |     +--rw pe-vlan                             uint16
     |     +--rw ce-vlan                             uint16
     +--rw dhcp-snp-white-lists
     |  +--rw dhcp-snp-white-list* [wht-lst-name]
     |     +--rw wht-lst-name                        string
     |     +--rw apply-flag                          boolean
     |     +--rw dhcp-snp-white-rules
     |        +--rw dhcp-snp-white-rule* [rule-id]
     |           +--rw rule-id                       uint16
     |           +--rw src-ip?                       inet:ip-address
```

```
       |           +--rw src-mask?                   inet:ip-address
       |           +--rw dst-ip?                     inet:ip-address
       |           +--rw dst-mask?                   inet:ip-address
       |           +--rw src-port?                   dhcp-snp-port
       |           +--rw dst-port?                   dhcp-snp-port
       +--ro dhcp-snp-dyn-bind-tbls
       |  +--ro dhcp-snp-dyn-bind-tbl* [ip-address outer-vlan inner-vlan vsi-nam
e vpn-name bridge-domain]
       |     +--ro ip-address                       inet:ip-address
       |     +--ro outer-vlan                       uint16
       |     +--ro inner-vlan                       uint16
       |     +--ro vsi-name                         string
       |     +--ro vpn-name                         string
       |     +--ro bridge-domain                    uint32
       |     +--ro mac-address?                     mac-address
       |     +--ro if-name?                         string
       |     +--ro lease?                           yang:date-and-time
       +--ro dhcp-snp-statistics
          +--ro pkt-cnt-drop-by-global-rate          uint32
          +--ro dhcp-snp-vlan-statistics
          |  +--ro dhcp-snp-vlan-statistic* [vlan-id]
          |     +--ro vlan-id                        uint16
          |     +--ro drop-arp-pkt-cnt?              uint32
          |     +--ro drop-ip-pkt-cnt?               uint32
          |     +--ro pkt-cnt-drop-by-user-bind?     uint32
          |     +--ro pkt-cnt-drop-by-mac-check?     uint32
          |     +--ro pkt-cnt-drop-by-untrust-reply? uint32
          |     +--ro pkt-cnt-drop-by-rate?          uint32
          +--ro dhcp-snp-vlan-if-statistics
          |  +--ro dhcp-snp-vlan-if-statistic* [vlan-id if-name]
          |     +--ro vlan-id                        uint16
          |     +--ro if-name                        string
          |     +--ro drop-arp-pkt-cnt?              uint32
          |     +--ro drop-ip-pkt-cnt?               uint32
          |     +--ro pkt-cnt-drop-by-user-bind?     uint32
          |     +--ro pkt-cnt-drop-by-mac-check?     uint32
          |     +--ro pkt-cnt-drop-by-untrust-reply? uint32
          |     +--ro pkt-cnt-drop-by-rate?          uint32
          +--ro dhcp-snp-if-statistics
          |  +--ro dhcp-snp-if-statistic* [if-name]
          |     +--ro if-name                        string
          |     +--ro drop-arp-pkt-cnt?              uint32
          |     +--ro drop-ip-pkt-cnt?               uint32
          |     +--ro pkt-cnt-drop-by-user-bind?     uint32
          |     +--ro pkt-cnt-drop-by-mac-check?     uint32
          |     +--ro pkt-cnt-drop-by-untrust-reply? uint32
          |     +--ro pkt-cnt-drop-by-rate?          uint32
          +--ro dhcp-snp-bd-statistics
             +--ro dhcp-snp-bd-statistic* [if-name]
```

```
                +--ro bd-id                            uint32
                +--ro drop-arp-pkt-cnt?                uint32
                +--ro drop-ip-pkt-cnt?                 uint32
                +--ro pkt-cnt-drop-by-user-bind?       uint32
                +--ro pkt-cnt-drop-by-mac-check?       uint32
                +--ro pkt-cnt-drop-by-untrust-reply?   uint32
                +--ro pkt-cnt-drop-by-rate?            uint32
```

4.5. CPU Protection

For the network device, there are maybe a large number of packets to be sent to its CPU, or malicious packets attempt to attack the device CPU. If the CPU receives excessive packets, it will be overloaded and support the normal services with very poor performance; In extreme cases, the system fails.

More specifically, services are negatively affected when the CPU is attacked because of the following reasons:

- o Valid protocol packets are not distinguished from invalid protocol packets. The CPU is busy in processing a large number of invalid protocol packets. Consequently, the CPU usage rises sharply and valid packets cannot be processed properly

- o Packets of some protocols are sent to the CPU through the same channel. When excessive packets of a certain type of protocol packet block the channel, the transmission of other protocol packets is affected

- o The bandwidth of a channel is not set appropriately. When an attack occurs, processing of protocol packets on other channels is affected

Accordingly, the following countermeasures can be taken by the network device for CPU protection:

- o Collect and classify protocols related to various services running on equipment

- o Use ACLs to filter the packets. Valid protocol packets are put into the whitelist and a user-defined flow, other packets are put into the blacklist

- o Plan the priorities, channel bandwidth, length of packets, and alarm function of the preceding three lists

- o Disable services that are not deployed on the equipment, and control the total forwarding bandwidth

In this manner, the number of packets sent to the CPU is under control, and the bandwidth is ensured preferentially for services with higher priorities. In addition, CPU overload is prevented and an alarm is generated when an attack occurs.

```
module: ietf-cpu-defend
    +--rw cpu-defend
       +--rw cpu-defend-policies
       |  +--rw cpu-defend-policy* [policy-id]
       |     +--rw policy-id                     uint32
       |     +--rw description?                  string
       |     +--rw white-list-acl-number?        uint32
       |     +--rw black-list-acl-number?        uint32
       |     +--rw user-defined-flows
       |     |  +--rw user-defined-flow* [flow-id]
       |     |     +--rw flow-id                 uint32
       |     |     +--rw acl-number              uint32
       |     +--rw cpu-defend-car-rules
       |        +--rw cpu-defend-car-rule* [rule-type pkt-index flow-id protocol
-type tcp-ip-type]
       |           +--rw rule-type               rule-type
       |           +--rw pkt-index?              uint16
       |           +--rw flow-id?                uint32
       |           +--rw protocol?               protocol-type
       |           +--rw tcp-ip-type?            tcp-ip-type
       |           +--rw car-attr
       |           |  +--rw cir?                 uint32
       |           |  +--rw cbs?                 uint32
       |           |  +--rw pir?                 uint32
       |           |  +--rw pbs?                 uint32
       |           |  +--rw min-pkt-len?         uint32
       |           |  +--rw pkt-rate?            uint32
       |           |  +--rw weight?              uint16
       |           +--rw priority?               priority-type
       |           +--rw drop-alarm
       |              +--rw enable               boolean
       |              +--rw packets-threshold?   uint32
       |              +--rw interval?            uint16
       |              +--rw speed-threshold?     uint32
       +--rw cpu-defend-policy-bindings
       |  +--rw cpu-defend-policy-binding* [slot-id]
       |     +--rw slot-id                       string
       |     +--rw policy-id                     uint32
       +--ro cpu-defend-cars-cfgs
       |  +--ro cpu-defend-cars-cfg* [slot-id pkt-index]
       |     +--ro slot-id                       string
       |     +--ro pkt-index                     uint16
       |     +--ro cir?                          uint32
       |     +--ro cbs?                          uint32
       |
```

```
     |     +--ro min-pkt?                        uint32
     |     +--ro priority?                       priority-type
     |     +--ro protocol                        protocol-type
     +--ro protocol-stats
     |  +--ro protocol-stat* [slot-id protocol]
     |     +--ro slot-id                         string
     |     +--ro protocol                        protocol-type
     |     +--ro default-act                     action-type
     |     +--ro default-cir                     uint32
     |     +--ro default-cbs                     uint32
     +--ro sec-non-car-stats
     |  +--ro sec-non-car-stat* [slot-id policy-type protocol]
     |     +--ro slot-id                         string
     |     +--ro policy-type                     policy-type
     |     +--ro protocol                        protocol-type
     |     +--ro total-packets?                  uint64
     |     +--ro passed-packets?                 uint64
     |     +--ro dropped-packets?                uint64
     +--ro sec-car-stats
     |  +--ro sec-car-stat* [slot-id policy-type policy-index]
     |     +--ro slot-id                         string
     |     +--ro policy-type                     policy-type
     |     +--ro policy-index                    uint32
     |     +--ro app-enable?                     boolean
     |     +--ro app-default-act?                action-type
     |     +--ro proto-enable?                   boolean
     |     +--ro passed-packets?                 uint64
     |     +--ro dropped-packets?                uint64
     |     +--ro cfg-cir?                        uint32
     |     +--ro cfg-cbs?                        uint32
     |     +--ro actual-cir?                     uint32
     |     +--ro actual-cbs?                     uint32
     |     +--ro priority?                       priority-type
     |     +--ro min-pkt-len?                    uint32
     |     +--ro acl-deny-packets?               uint64
     |     +--ro hist-pps?                       uint64
     |     +--ro hist-pps-time?                  yang:date-and-time
     |     +--ro average-drop-rate?              uint64
     |     +--ro drop-begin-time?                yang:date-and-time
     |     +--ro drop-end-time?                  yang:date-and-time
     |     +--ro total-dropped-packets?          uint64
     +--ro total-packet-stats
     |  +--ro total-packet-stat* [slot-id]
     |     +--ro slot-id                         string
     |     +--ro total-packets?                  uint64
     |     +--ro passed-packets?                 uint64
     |     +--ro dropped-packets?                uint64
     +--rw hostcar-policies
```

```
       |  +--rw hostcar-policy* [slot-id host-car-type]
       |     +--rw slot-id                          string
       |     +--rw host-car-type                    host-car-type
       |     +--rw cir?                             uint32
       |     +--rw pir?                             uint32
       |     +--rw cbs?                             uint32
       |     +--rw pbs?                             uint32
       |     +--rw automatic-adjustment
       |        +--rw enable?                       boolean
       |        +--rw drop-threshold?               uint32
       |        +--rw interval?                     uint32
       +--ro host-car-stats
       |  +--ro host-car-stat* [slot-id host-car-type stat-type host-car-id http
-host-car-id vlan-host-car-id]
       |     +--ro slot-id                          string
       |     +--ro host-car-type                    host-car-type
       |     +--ro stat-type                        stat-type
       |     +--ro host-car-id                      uint32
       |     +--ro http-host-car-id                 uint32
       |     +--ro vlan-host-car-id                 uint32
       |     +--ro passed-bytes?                    uint64
       |     +--ro dropped-bytes?                   uint64
       +--ro host-car-cfgs
          +--ro host-car-cfg* [slot-id]
             +--ro slot-id                          string
             +--ro host-car-type?                   host-car-type-enum
             +--ro default-cir?                     uint32
             +--ro default-pir?                     uint32
             +--ro default-cbs?                     uint32
             +--ro default-pbs?                     uint32
             +--ro actual-cir?                      uint32
             +--ro actual-pir?                      uint32
             +--ro actual-cbs?                      uint32
             +--ro actual-pbs?                      uint32
             +--ro droprate-en?                     boolean
             +--ro log-interval?                    uint32
             +--ro log-threshold?                   uint32
```

4.6. TCP/IP Attack Defense

Defense against TCP/IP attacks is applied to the router on the edge of the network or other routers that are easily to be attacked by illegal TCP/IP packets. Defense against TCP/IP attacks can protect the CPU of the router against malformed packets, fragmented packets, TCP SYN packets, and UDP packets, ensuring that normal services can be processed.

```
module: ietf-tcp-ip-attack-defense
    +--rw tcp-ip-attack-defense
       +--rw anti-enable?                               boolean
       +--rw abnormal-enable?                           boolean
       +--rw udp-flood-enable?                          boolean
       +--rw tcp-syn-enable?                            boolean
       +--rw icmp-flood-enable?                         boolean
       +--rw fragment-enable?                           boolean
       +--rw sec-anti-attack-car-cfg
       |  +--rw cir-fragment?                           uint32
       |  +--rw cir-icmp?                               uint32
       |  +--rw cir-tcp?                                uint32
       +--rw sec-anti-attack-stats
          +--ro sec-anti-attack-stat* [attack-type]
             +--ro attack-type                          anti-attack-type
             +--ro total-count?                         uint64
             +--ro drop-count?                          uint64
             +--ro pass-count?                          uint64
```

5. Network Infrastructure Device Security Baseline Yang Module

```
   <CODE BEGINS> file "ietf-mac-limit@2018-06-04.yang"


  module ietf-mac-limit {
  namespace "urn:ietf:params:xml:ns:yang:ietf-mac-limit";
  prefix mac-limit;
  organization
    "IETF SACM Working Group";
  contact
    "Liang Xia: Frank.xialiang@huawei.com;
    Guangying Zheng: Zhengguangying@huawei.com";
  description
    "MAC address limit.";

  revision 2018-06-04 {
    description
      "Init revision";
    reference "xxx.";
  }

  /*
   * Typedefs
   */
  typedef mac-limit-forward {
    type enumeration {
      enum "forward" {
```

```
          description
            "Forward.";
        }
        enum "discard" {
          description
            "Discard.";
        }
      }
      description
        "MAC Limit Forward";
    }
    typedef mac-enable-status {
      type enumeration {
        enum "enable" {
          description
            "Enable.";
        }
        enum "disable" {
          description
            "Disable.";
        }
      }
      description
        "MAC Enable Status";
    }
    typedef mac-vlan-id {
      type uint16 {
        range "1..4094";
      }
      description
        "MAC Vlan Id";
    }
    typedef mac-type {
      type enumeration {
        enum "static" {
          description
            "Static MAC address entry.";
        }
        enum "dynamic" {
          description
            "Dynamic MAC address entry.";
        }
        enum "black-hole" {
          description
            "Blackhole MAC address entry";
        }
        enum "sticky" {
          description
```

```
            "sticky MAC address entry";
        }
        enum "security" {
          description
            "security MAC address entry";
        }
        enum "evn" {
          description
            "EVN MAC address entry.";
        }
        enum "mux" {
          description
            "MUX MAC address entry.";
        }
        enum "snooping" {
          description
            "SNOOPING MAC address entry.";
        }
        enum "tunnel" {
          description
            "TUNNEL MAC address entry.";
        }
        enum "authen" {
          description
            "AUTHEN MAC address entry.";
        }
      }
      description
        "MAC Type";
    }
    typedef suppress-type {
      type enumeration {
        enum "broadcast" {
          description
            "Broadcast.";
        }
        enum "multicast" {
          description
            "Multicast.";
        }
        enum "unknown-unicast" {
          description
            "Unknown unicast.";
        }
        enum "unicast" {
          description
            "Unicast.";
        }
```

```
    }
    description
      "Suppress Type";
  }
  typedef limit-type {
    type enumeration {
      enum "-mac-limit" {
        description
          "Interface MAC rule limit.";
      }
      enum "mac-apply" {
        description
          "Interface MAC rule application.";
      }
    }
    description
      "Limit Type";
  }

  typedef mac-pw-encap-type {
    type enumeration {
      enum "ethernet" {
        description
          "Ethernet.";
      }
      enum "vlan" {
        description
          "VLAN.";
      }
    }
    description
      "MAC PW Encapsulation Type";
  }

  typedef suppress-style {
    type enumeration {
      enum "percent" {
        description
          "Percent.";
      }
      enum "absolute-value" {
        description
          "Absolute value.";
      }
    }
    description
      "Suppress Style";
  }
```

```
  typedef direction-type {
    type enumeration {
      enum "inbound" {
        description
          "Inbound.";
      }
      enum "outbound" {
        description
          "Outbound.";
      }
    }
    description
      "Direction Type";
  }

  typedef storm-ctrl-action-type {
    type enumeration {
      enum "normal" {
        description
          "Normal.";
      }
      enum "error-down" {
        description
          "Error down.";
      }
      enum "block" {
        description
          "Block.";
      }
      enum "suppress" {
        description
          "Suppress";
      }
    }
    description
      "Storm Ctrl Action Type";
  }

  typedef enable-type {
    type enumeration {
      enum "disable" {
        description
          "Disable.";
      }
      enum "enable" {
        description
          "Enable.";
      }
```

```
      }
      description
        "Enable Type";
    }

    typedef storm-ctrl-type {
      type enumeration {
        enum "broadcast" {
          description
            "Broadcast.";
        }
        enum "multicast" {
          description
            "Multicast.";
        }
        enum "unicast" {
          description
            "Unicast.";
        }
        enum "unknown-unicast" {
          description
            "Unknown unicast.";
        }
      }
      description
        "Storm Ctrl Type";
    }

    typedef storm-ctrl-rate-type {
      type enumeration {
        enum "pps" {
          description
            "Packets per second.";
        }
        enum "percent" {
          description
            "Percent.";
        }
        enum "kbps" {
          description
            "Kilo bits per second.";
        }
      }
      description
        "Storm Ctrl Rate Type";
    }
```

```
  container mac {
    description
      "MAC address forwarding. ";
    container mac-limit-rules {
      description
        "Global MAC address learning limit rule.";
      list mac-limit-rule {
        key "rule-name";
        description
          "Global MAC address learning limit.";
        leaf rule-name {
          type string {
            length "1..31";
          }
          description
            "Global MAC address learning limit rule name.";
        }
        leaf maximum {
          type uint32 {
            range "0..131072";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number.";
        }
      }
    }
    container vlan-mac-limits {
```

```
      description
        "VLAN MAC address limit list.";
      list vlan-mac-limit {
        key "vlan-id";
        description
          "VLAN MAC address limit.";
        leaf vlan-id {
          type mac-vlan-id;
          description
            "VLAN ID.";
        }
        leaf maximum {
          type uint32 {
            range "0..130048";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a VLAN.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a VLAN.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number in a VLAN.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number in a VLAN.";
        }
      }
    }
    container vsi-mac-limits {
      description
        "VSI MAC address limit list.";
      list vsi-mac-limit {
        key "vsi-name";
        description
          "VSI MAC address limit.";
```

```
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf maximum {
          type uint32 {
            range "0..524288";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a VSI.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a VSI.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number in a VSI.";
        }
        leaf alarm {
          type mac-enable-status;
          default "disable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number in a VSI.";
        }
        leaf up-threshold {
          type uint8 {
            range "80..100";
          }
          mandatory true;
          description
            "Upper limit for the number of MAC addresses.";
        }
        leaf down-threshold {
          type uint8 {
            range "60..100";
          }
          mandatory true;
```

```
          description
            "Upper limit for the number of MAC addresses.";
        }
      }
    }
    container bd-mac-limits {
      description
        "BD MAC address limit list.";
      list bd-mac-limit {
        key "bd-id";
        description
          "BD MAC address limit.";
        leaf bd-id {
          type uint32 {
            range "1..16777215";
          }
          description
            "Specifies the ID of a bridge domain.";
        }
        leaf maximum {
          type uint32 {
            range "0..130048";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a BD.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a BD.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";

          description
            "Forward or discard the packet.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number.";
        }
```

```
      }
    }
    container pw-mac-limits {
      description
        "PW MAC address limit list.";
      list pw-mac-limit {
        key "vsi-name pw-name";
        description
          "PW MAC address limit.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf pw-name {
          type string {
            length "1..15";
          }
          description
            "PW name.";
        }
        leaf maximum {
          type uint32 {
            range "0..130048";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned in a PW.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval at which MAC addresses are learned in a PW.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number in a PW.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
```

```
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number in a PW.";
        }
      }
    }
    container if-mac-limits {
      description
        "Interface MAC address limit list.";
      list if-mac-limit {
        key "if-name limit-type";
        description
          "Interface MAC address limit.";
        leaf if-name {
          type string;
          description
            "Interface name.";
        }
        leaf limit-type {
          type limit-type;
          description
            "Interface MAC limit type.";
        }
        leaf rule-name {
          type leafref {
            path "/mac/mac-limit-rules/mac-limit-rule/rule-name";
          }
          description
            "Rule name.";
        }
        leaf maximum {
          type uint32 {
            range "0..131072";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned on an interface
.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval (ms) at which MAC addresses are learned on an interface.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
```

```
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number on an interface";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number on an interface.";
        }
      }
    }
    container if-vlan-mac-limits {
      description
        "Interface + VLAN MAC address limit list.";
      list if-vlan-mac-limit {
        key "if-name vlan-begin limit-type";
        config false;
        description
          "Interface + VLAN MAC address limit.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf vlan-begin {
          type mac-vlan-id;
          description
            "Start VLAN ID.";
        }
        leaf vlan-end {
          type mac-vlan-id;
          description
            "End VLAN ID.";
        }
        leaf limit-type {
          type limit-type;
          description
            "Interface MAC limit type.";
        }
        leaf rule-name {
          type leafref {
            path "/mac/mac-limit-rules/mac-limit-rule/rule-name";
          }
          description
            "Rule name.";
        }
        leaf maximum {
          type uint32 {
```

```
            range "0..131072";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned on an interface
.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          mandatory true;
          description
            "Interval (ms) at which MAC addresses are learned on an interface.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward the packet.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number.";
        }
      }
    }
    container subif-mac-limits {
      description
        "Sub-interface MAC address limit list.";
      list subif-mac-limit {
        key "if-name limit-type";
        description
          "Sub-interface MAC address limit.";
        leaf if-name {
          type string;
          description
            "-name of a sub-interface. ";
        }
        leaf limit-type {
          type limit-type;
          description
            "Sub-interface MAC limit type.";
        }
        leaf vsi-name {
          type string {
            length "1..36";
```

```
          }
          config false;
          mandatory true;
          description
            "VSI name , EVPN name or bridge domain ID.";
        }
        leaf rule-name {
          type string {
            length "1..31";
          }
          mandatory true;
          description
            "Rule name.";
        }
        leaf maximum {
          type uint32 {
            range "0..131072";
          }
          mandatory true;
          description
            "Maximum number of MAC addresses that can be learned on a sub-interf
ace.";
        }
        leaf rate {
          type uint16 {
            range "0..1000";
          }
          default "0";
          description
            "Interval (ms) at which MAC addresses are learned on a sub-interface
.";
        }
        leaf action {
          type mac-limit-forward;
          default "discard";
          description
            "Discard or forward after the number of learned MAC addresses reache
s the maximum number on a sub-interface.";
        }
        leaf alarm {
          type mac-enable-status;
          default "enable";
          description
            "Whether an alarm is generated after the number of learned MAC addre
sses reaches the maximum number on a sub-interface.";
        }
      }
    }
    container vsi-storm-supps {
      description
        "VSI Suppression List.";
      list vsi-storm-supp {
```

```
        key "vsi-name suppress-type";
        description
          "VSI Suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Traffic suppression type.";
        }
        leaf cir {
          type uint64 {
            range "0..4294967295";
          }
          default "0";
          description
            "CIR value.";
        }
        leaf cbs {
          type uint64 {
            range "0..4294967295";
          }
          description
            "CBS value.";
        }
      }
    }
    container vlan-storm-supps {
      description
        "VLAN Suppression List.";
      list vlan-storm-supp {
        key "vlan-id suppress-type";
        description
          "VLAN Suppression.";
        leaf vlan-id {
          type mac-vlan-id;
          description
            "VLAN ID.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Traffic suppression type.";
```

```
        }
        leaf cir {
          type uint64 {
            range "64..4294967295";
          }
          default "64";
          description
            "CIR value.";
        }
        leaf cbs {
          type uint64 {
            range "10000..4294967295";
          }
          description
            "CBS value.";
        }
      }
    }
    container sub-if-suppresss {
      description
        "Sub-interface traffic suppression list.";
      list sub-if-suppress {
        key "if-name suppress-type direction";
        description
          "Sub-Interface traffic suppression.";
        leaf if-name {
          type string;
          description
            "Sub-interface name.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Suppression type.";
        }
        leaf direction {
          type direction-type;
          description
            "Suppression direction.";
        }
        leaf cir {
          type uint64 {
            range "0..4294967295";
          }
          default "0";
          description
            "CIR value.";
        }
```

```
        leaf cbs {
          type uint64 {
            range "0..4294967295";
          }
          description
            "CBS value.";
        }
      }
    }
    container pw-suppresss {
      description
        "PW traffic suppress list.";
      list pw-suppress {
        key "vsi-name pw-name suppress-type";
        description
          "PW traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf pw-name {
          type string {
            length "1..15";
          }
          description
            "PW name.";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Traffic suppression type.";
        }
        leaf cir {
          type uint64 {
            range "100..4294967295";
          }
          default "100";
          description
            "CIR value.";
        }
        leaf cbs {
          type uint64 {
            range "100..4294967295";
          }
          description
```

```
            "CBS value.";
        }
      }
    }

    container vsi-in-suppressions {
      description
        "VSI inbound traffic suppression list.";
      list vsi-in-suppression {
        key "vsi-name";
        description
          "VSI inbound traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf inbound-supp {
          type mac-enable-status;
          default "enable";
          description
            "Inbound suppression.";
        }
      }
    }
    container vsi-out-suppressions {
      description
        "VSI outbound traffic suppression list.";
      list vsi-out-suppression {
        key "vsi-name";
        description
          "VSI outbound traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          description
            "VSI name.";
        }
        leaf out-bound-supp {
          type mac-enable-status;
          default "enable";
          description
            "Outbound suppression.";
        }
      }
```

```
    }
    container vsi-suppresss {
      description
        "VSI traffic suppression list.";
      list vsi-suppress {
        key "sub-if-name";
        description
          "VSI traffic suppression.";
        leaf vsi-name {
          type string {
            length "1..31";
          }
          mandatory true;
          description
            "VSI name.";
        }

        leaf sub-if-name {
          type string;
          description
            "Sub-interface name.";
        }
        leaf is-enable {
          type boolean;
          default "true";
          description
            "Enable status.";
        }
        leaf suppress-type {
          type suppress-style;
          default "percent";
          description
            "Traffic suppression type.";
        }
        leaf broadcast {
          type uint32 {
            range "0..200000000";
          }
          default "64";
          description
            "Broadcast suppression (kbit/s)";
        }
        leaf broadcast-percent {
          type uint32 {
            range "0..100";
          }
          default "1";
          description
```

```
            "Broadcast suppression.";
        }
        leaf unicast {
          type uint32 {
            range "0..200000000";
          }
          default "64";
          description
            "Unknown unicast suppression (kbit/s).";
        }
        leaf unicast-percent {
          type uint32 {
            range "0..100";
          }
          default "1";
          description
            "Unknown unicast suppression.";

        }
        leaf multicast {
          type uint32 {
            range "0..200000000";
          }
          default "64";
          description
            "Multicast suppression (kbit/s).";
        }
        leaf multicast-percent {
          type uint32 {
            range "0..100";
          }
          default "1";
          description
            "Multicast suppression.";
        }
      }
    }
    container vsi-total-numbers {
      description
        "List of MAC address total numbers in a VSI.";
      list vsi-total-number {
        key "vsi-name slot-id mac-type";
        config false;
        description
          "Total number of MAC addresses in a VSI.";
        leaf vsi-name {
          type string {
            length "1..31";
```

```
          }
          description
            "VSI name.";
        }
        leaf slot-id {
          type string {
            length "1..24";
          }
          description
            "Slot ID.";
        }
        leaf mac-type {
          type mac-type;
          description
            "MAC address type.";
        }
        leaf number {
          type uint32;
          mandatory true;
          description
            "Number of MAC addresses.";
        }
      }
    }
    container if-storm-supps {
      description
        "Interface traffic suppression list.";
      list if-storm-supp {
        key "if-name suppress-type";
        description
          "Interface traffic suppression.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf suppress-type {
          type suppress-type;
          description
            "Suppression type.";
        }
        leaf percent {
          type uint64 {
            range "0..99";
          }
          description
            "Percent.";
        }
```

```
        leaf packets {
          type uint64 {
            range "0..148810000";
          }
          description
            "Packets per second.";
        }
        leaf cir {
          type uint64 {
            range "0..100000000";
          }
          description
            "CIR(Kbit/s).";
        }
        leaf cbs {
          type uint64 {
            range "10000..4294967295";
          }
          description
            "CBS(Bytes).";
        }
      }
    }
    container if-storm-blocks {
      description
        "Interface traffic block list.";
      list if-storm-block {
        key "if-name block-type direction";
        description
          "Interface traffic suppression.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf block-type {
          type suppress-type;
          description
            "Block type.";
        }
        leaf direction {
          type direction-type;
          description
            "Direction.";
        }
      }
    }
    container if-storm-contrls {
```

```
      description
        "Interface storm control list.";
      list if-storm-contrl {
        key "if-name";
        description
          "Interface storm control.";
        leaf if-name {
          type string;
          description
            "-name of an interface. ";
        }
        leaf action {
          type storm-ctrl-action-type;
          default "normal";
          description
            "Action type.";
        }
        leaf trap-enable {

          type enable-type;
          default "disable";
          description
            "Trap state.";
        }
        leaf log-enable {
          type enable-type;
          default "disable";
          description
            "Log state.";
        }
        leaf interval {
          type uint64 {
            range "1..180";
          }
          default "5";
          description
            "Detect interval.";
        }
        container if-packet-contrl-attributes {
          description
            "Storm control rate list.";
          list if-packet-contrl-attribute {
            key "packet-type";
            description
              "Storm control rate.";
            leaf packet-type {
              type storm-ctrl-type;
              description
```

```
                "Packet type.";
            }
            leaf rate-type {
              type storm-ctrl-rate-type;
              default "pps";
              description
                "Storm control rate type.";
            }
            leaf min-rate {
              type uint32 {
                range "1..148810000";
              }
              mandatory true;
              description
                "Storm control min rate.";
            }
            leaf max-rate {
              type uint64 {
                range "1..148810000";
              }
              mandatory true;
              description
                "Storm control max rate.";
            }
          }
        }
        container ifstorm-contrl-infos {
          description
            "Storm control info list.";
          list ifstorm-contrl-info {
            key "packet-type";
            config false;
            description
              "Storm control info";
            leaf packet-type {
              type storm-ctrl-type;
              description
                "Packet type.";
            }
            leaf punish-status {
              type storm-ctrl-action-type;
              description
                "Storm control status.";
            }
            leaf last-punish-time {
              type string {
                length "1..50";
              }
```

```
              description
                "Last punish time.";
            }
          }
        }
      }
    }
  }
}
```

<CODE ENDS>

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

To be added.

8. Acknowledgements

9. References

9.1. 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>.

9.2. Informative References

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-17 (work in progress), September 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-19 (work in progress), September 2018.

[I-D.ietf-sacm-information-model]
Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus, M., Haynes, D., and H. Birkholz, "SACM Information Model", draft-ietf-sacm-information-model-10 (work in progress), April 2017.

Authors' Addresses

Liang Xia
Huawei

Email: frank.xialiang@huawei.com

Guangying Zheng
Huawei

Email: zhengguangying@huawei.com

Wei Pan
Huawei

Email: william.panwei@huawei.com