

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 26, 2018

Y. Dong
L. Xia
Huawei
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.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 26, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 1.1. Infrastructure layer security baseline | 3 |
| 2. Terminology | 4 |
| 2.1. Key Words | 4 |
| 2.2. Definition of Terms | 4 |
| 3. Tree Diagrams | 4 |
| 4. Data Model Structure | 5 |
| 4.1. Integrity measurement | 5 |
| 4.2. Cryptography security | 6 |
| 4.2.1. Symmetrical cryptography | 7 |
| 4.2.2. Asymmetrical cryptography | 8 |
| 4.2.3. Hash function | 10 |
| 4.2.4. Message authentication code | 10 |
| 4.2.5. Key derivation function | 11 |
| 4.3. Key management | 11 |
| 4.3.1. Key generation | 12 |
| 4.3.2. Key distribution | 13 |
| 4.3.3. Key store | 13 |
| 4.3.4. Key update | 13 |
| 4.3.5. Key backup | 14 |
| 4.3.6. Key destroy | 14 |
| 4.4. Cert management | 14 |
| 4.4.1. Cert management | 15 |
| 4.4.2. CRL management | 16 |
| 5. Infrastructure Layer YANG Module | 17 |
| 6. IANA Considerations | 26 |
| 7. Security Considerations | 26 |
| 8. Acknowledgements | 26 |
| 9. References | 26 |
| 9.1. Normative References | 26 |
| 9.2. Informative References | 26 |
| Authors' Addresses | 27 |

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-envelop)
        +---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 describes 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 temporary-key {  
          leaf key-length {  
            type unit16;  
            description  
              "The length of the temporary 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 {
    container temporary-key {
        leaf key-length {
            type unit16;
            description
                "The length of the temporary 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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

H. Birkholz
Fraunhofer SIT
J. Fitzgerald-McKay
National Security Agency
C. Schmidt
The MITRE Corporation
D. Waltermire
NIST
7 March 2022

Concise Software Identification Tags
draft-ietf-sacm-coswid-21

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

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

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

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

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | | |
|--------|---|----|
| 1. | Introduction | 3 |
| 1.1. | The SWID and CoSWID Tag Lifecycle | 4 |
| 1.2. | Concise SWID Format | 8 |
| 1.3. | Requirements Notation | 8 |
| 2. | Concise SWID Data Definition | 8 |
| 2.1. | Character Encoding | 10 |
| 2.2. | Concise SWID Extensions | 10 |
| 2.3. | The concise-swid-tag Map | 13 |
| 2.4. | concise-swid-tag Co-Constraints | 18 |
| 2.5. | The global-attributes Group | 18 |
| 2.6. | The entity-entry Map | 19 |
| 2.7. | The link-entry Map | 21 |
| 2.8. | The software-meta-entry Map | 25 |
| 2.9. | The Resource Collection Definition | 28 |
| 2.9.1. | The hash-entry Array | 28 |
| 2.9.2. | The resource-collection Group | 28 |
| 2.9.3. | The payload-entry Map | 32 |
| 2.9.4. | The evidence-entry Map | 32 |
| 2.10. | Full CDDL Specification | 33 |
| 3. | Determining the Type of CoSWID | 39 |
| 4. | CoSWID Indexed Label Values | 40 |
| 4.1. | Version Scheme | 40 |
| 4.2. | Entity Role Values | 42 |
| 4.3. | Link Ownership Values | 44 |
| 4.4. | Link Rel Values | 44 |
| 4.5. | Link Use Values | 46 |
| 5. | URI Schemes | 47 |
| 5.1. | "swid" URI Scheme | 47 |
| 5.2. | "swidpath" URI Scheme | 48 |
| 6. | IANA Considerations | 49 |
| 6.1. | CoSWID Items Registry | 49 |
| 6.2. | Software Tag Values Registries | 52 |
| 6.2.1. | Registration Procedures | 52 |
| 6.2.2. | Private Use of Index and Name Values | 52 |
| 6.2.3. | Expert Review Criteria | 53 |
| 6.2.4. | Software Tag Version Scheme Values Registry | 53 |
| 6.2.5. | Software Tag Entity Role Values Registry | 55 |

| | | |
|--------|--|----|
| 6.2.6. | Software Tag Link Ownership Values Registry | 56 |
| 6.2.7. | Software Tag Link Relationship Values Registry | 57 |
| 6.2.8. | Software Tag Link Use Values Registry | 60 |
| 6.3. | swid+cbor Media Type Registration | 61 |
| 6.4. | CoAP Content-Format Registration | 62 |
| 6.5. | CBOR Tag Registration | 62 |
| 6.6. | URI Scheme Registrations | 62 |
| 6.6.1. | URI-scheme swid | 63 |
| 6.6.2. | URI-scheme swidpath | 63 |
| 6.7. | CoSWID Model for use in SWIMA Registration | 64 |
| 7. | Signed CoSWID Tags | 64 |
| 8. | CBOR-Tagged CoSWID Tags | 67 |
| 9. | Security Considerations | 67 |
| 10. | Privacy Consideration | 71 |
| 11. | Change Log | 72 |
| 12. | References | 77 |
| 12.1. | Normative References | 77 |
| 12.2. | Informative References | 80 |
| | Acknowledgments | 81 |
| | Contributors | 81 |
| | Authors' Addresses | 82 |

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) [RFC8949].

Size comparisons between XML SWID and CoSWID mainly depend on domain-specific applications and the complexity of attributes used in instances. While the values stored in CoSWID are often unchanged and therefore not reduced in size compared to an XML SWID, the scaffolding that the CoSWID encoding represents is significantly smaller by taking up 10 percent or less in size. This effect is visible in representation sizes, which in early experiments benefited from a 50 percent to 85 percent reduction in generic usage scenarios. Additional size reduction is enabled with respect to the memory footprint of XML parsing/validation.

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 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 CoSWID 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.

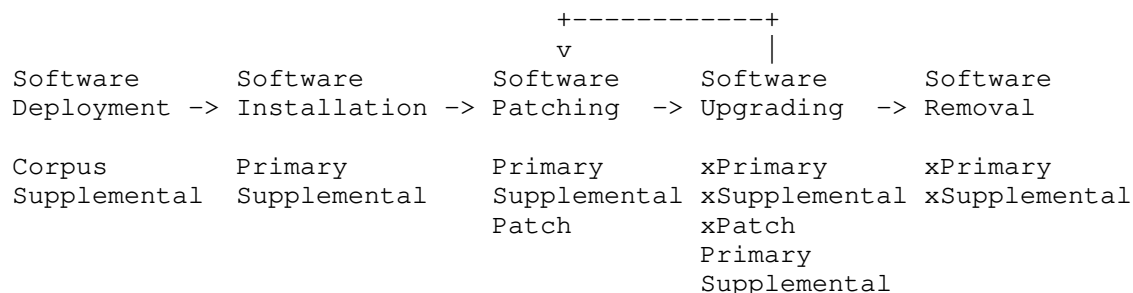


Figure 1: Use of Tag Types in the Software Lifecycle

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. XPath expressions [W3C.REC-xpath20-20101214] need to use SWID names, see Section 5.2.

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.

Through use of CDDL-based integer labels, CoSWID allows for future expansion in subsequent revisions of this specification and through extensions (see Section 2.2). New constructs can be associated with a new integer index. A deprecated construct can be replaced by a new construct with a new integer index. An implementation can use these integer indexes to identify the construct to parse. The CoSWID Items registry, defined in Section 6.1, is used to ensure that new constructs are assigned a unique index value. This approach avoids the need to have an explicit CoSWID version.

In a number of places, the value encoding admits both integer values and text strings. The integer values are defined in a registry specific to the kind of value; the text values are not intended for interchange and exclusively meant for private use as defined in Section 6.2.2. Encoders SHOULD NOT use string values based on the names registered in the registry, as these values are less concise than their index value equivalent; a decoder MUST however be prepared to accept text strings that are not specified in this document (and ignore the construct if that string is unknown). In the rest of the document, we call this an "integer label with text escape".

The root of the CDDL specification provided by this document is the rule `coswid` (as defined in Section 8):

```
start = coswid
```

In CBOR, an array is encoded using bytes that identify the array, and the array's length or stop point (see [RFC8949]). To make items that support 1 or more values, the following CDDL notation 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.

Usage of this construct can be simplified using

```
one-or-more<T> = T / [ 2* T ]
```

simplifying the above example to

```
_name_ = (_label_ => one-or-more<_data_>)
```

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 Section 3.1 of [RFC8949]). 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 6.2 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:

- * 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 2.4, Paragraph 2.

- * 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 |
| resource-collection | \$\$resource-collection-extension | Section 2.9.2 |
| 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 6.1 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 6.2 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 => one-or-more<software-meta-entry>,  
  entity => one-or-more<entity-entry>,  
  ? link => one-or-more<link-entry>,  
  ? payload-or-evidence,  
  * $$coswid-extension,  
  global-attributes,  
}
```

```
payload-or-evidence //= ( payload => payload-entry )  
payload-or-evidence //= ( evidence => 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 /= int / 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 2.4, Paragraph 2.
- * **tag-id (index 0):** A 16-byte binary string, or a textual identifier, uniquely referencing a software component. The tag identifier **MUST** be globally unique. Failure to ensure global uniqueness can create ambiguity in tag use since the tag-id serves as the global key for matching and lookups. 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 the identifier is structured, but examples include a 16-byte GUID (e.g., class 4 UUID) [RFC4122], or a DNS domain name followed by a "/" and a text string, where the domain name serves to ensure uniqueness across organizations. A textual tag-id **MUST NOT** contain a sequence of two underscores ("__", see Section 6.7).
- * **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 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.
- * **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 an 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".

- * `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.
- * `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".
- * `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. This item maps to `'/SoftwareIdentity/@name'` in [SWID].
- * `software-version` (index 13): A textual value representing the specific release or development version of the software component. This item maps to `'/SoftwareIdentity/@version'` in [SWID].
- * `version-scheme` (index 14): An integer or textual value representing the versioning scheme used for the software-version item, as an integer label with text escape (Section 2, for the "Version Scheme" registry Section 4.1. . 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 6.2.2). Integer values in the range 0 to 65535 correspond to registered entries in the IANA "Software Tag Version Scheme Values" registry (see Section 6.2.4).
- * `media` (index 10): This text value is a hint to the tag consumer to understand what target platform this tag applies to. This 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. This item maps to `'/SoftwareIdentity/Meta'` in [SWID].
- * `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. This item is mutually exclusive to evidence, as payload can only be provided by an external entity. Described in Section 2.9.3.
- * `evidence` (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. This item is mutually exclusive to payload, as evidence is always generated on the target device ad-hoc. 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. If any of these constraints is not met, a signed tag cannot be used anymore as a signed statement.

- * 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 2.7) with both the rel item value of "patches" and an href item specifying an association with the software that was patched. Without at least one link item the target of the patch cannot be identified and the patch tag cannot be applied without external context.
- * 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. Without at least one link item the target of supplement tag cannot be identified and the patch tag cannot be applied without external context.
- * 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 => text,  
    * any-attribute,  
)  
  
any-attribute = (  
    label => one-or-more<text> / one-or-more<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 => one-or-more<$role>,  
    ? thumbprint => hash-entry,  
    * $$entity-extension,  
    global-attributes,  
}
```

```
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 /= int / 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 2.4, Paragraph 2.
- * `entity-name` (index 31): The textual name of the organizational entity claiming the roles specified by the role item for the CoSWID tag. This item maps to `'/SoftwareIdentity/Entity/@name'` in [SWID].
- * `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; it is not intended to be dereferenced. The scope will usually be the scope of an organization.
- * `role` (index 33): An integer or textual value (integer label with text escape, see Section 2) 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 6.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Software Tag Entity Role Values" registry (see Section 6.2.5).

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 a hash (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. 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,  
    * $$link-extension,  
    global-attributes,  
}
```

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

```
$ownership /= shared  
$ownership /= private  
$ownership /= abandon  
$ownership /= int / text  
abandon=1  
private=2  
shared=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 /= int / text
optional=1
required=2
recommended=3
```

The following describes each member of this map.

- * `global-attributes`: The `global-attributes` group described in Section 2.4, Paragraph 2.
- * `artifact` (index 37): To be used with `rel="installation-media"`, this item's value provides the absolute filesystem 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 relative reference relative to the base URI of the CoSWID tag, i.e., the URI under which the CoSWID tag was provided. 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-f7ddddlade4c"` references the tag with the tag-id value `"2df9de35-0aff-4a86-ace6-f7ddddlade4c"`.

- a URI with "swidpath:" as the scheme, which refers to another software tag via an XPATH query [W3C.REC-xpath20-20101214] that matches items in that tag (Section 5.2). This scheme is provided for compatibility with [SWID]. This specification does not define how to resolve an XPATH query in the context of CBOR, see Section 5.2.
- * 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]). As highlighted in media defined in Section 2.3, 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.
- * ownership (index 39): An integer or textual value (integer label with text escape, see Section 2, for the "Software Tag Link Ownership Values" registry Section 4.3) 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 6.2.2). Integer values in the range 0 to 255 correspond to registered entries in the "Software Tag Link Ownership Values" registry.

- * `rel` (index 40): An integer or textual value that (integer label with text escape, see Section 2, for the "Software Tag Link Relationship Values" registry Section 4.3) 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 6.2.2). Integer values in the range 0 to 65535 correspond to registered entries in the IANA "Software Tag Link Relationship Values" registry (see Section 6.2.7. If a string value is used it MUST be either a private use name as defined in Section 6.2.2 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 correspond to registered entries in the "Software Tag Link Relationship Values" registry.
- * `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. (This is a `_hint_`: There is no obligation for the server hosting the target of the URI to use the indicated media type when the URI is dereferenced.) Media types are identified by referencing a "Name" from the IANA "Media Types" registry: <http://www.iana.org/assignments/media-types/media-types.xhtml>. This item maps to `'/SoftwareIdentity/Link/@type'` in [SWID].
- * `use` (index 42): An integer or textual value (integer label with text escape, see Section 2, for the "Software Tag Link Relationship Values" registry Section 4.3) 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 6.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Link Use Values" registry (see Section 6.2.8. If a string value is used it MUST be a private use name as defined in Section 6.2.2. String values correspond to registered entries in the "Software Tag Link Use Values" registry.

- * `$$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 / bstr .size 16,  
  ? persistent-id => text,  
  ? product => text,  
  ? product-family => text,  
  ? revision => text,  
  ? summary => text,  
  ? unspsc-code => text,  
  ? unspsc-version => text,  
  * $$software-meta-extension,  
  global-attributes,  
}
```

```
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 2.4, Paragraph 2.

- * `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 (byte-by-byte) 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. Since CoSWID tags are not intended to contain confidential information, tag authors are advised not to record unprotected, private software license keys in this field.

- * 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" [IANA.named-information] 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` is an integer-based hash algorithm identifier whose value MUST refer to an ID in the IANA "Named Information Hash Algorithm Registry" [IANA.named-information] with a Status of "current" (at the time the generator software was built or later); other hash algorithms MUST NOT be used. If the `hash-alg-id` is not known, then the integer value "0" MUST be used. This allows for conversion from ISO SWID tags [SWID], which do not allow an algorithm to be identified for this field.

The `hash-value` MUST represent the raw hash value as a byte string (as opposed to, e.g., base64 encoded) generated from the representation of the resource using the hash algorithm indicated by `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 => one-or-more<directory-entry>,  
                        ? file => one-or-more<file-entry>,  
                        )  
  
resource-collection = (  
    path-elements-group,  
    ? process => one-or-more<process-entry>,  
    ? resource => one-or-more<resource-entry>,  
    * $$resource-collection-extension,  
)  
  
filesystem-item = (  
    ? key => bool,  
    ? location => text,  
    fs-name => text,  
    ? root => text,  
)  
  
file-entry = {  
    filesystem-item,  
    ? size => uint,  
    ? file-version => text,  
    ? hash => hash-entry,  
    * $$file-extension,  
    global-attributes,  
}  
  
directory-entry = {  
    filesystem-item,  
    ? path-elements => { path-elements-group },  
    * $$directory-extension,  
    global-attributes,  
}  
  
process-entry = {  
    process-name => text,  
    ? pid => integer,  
    * $$process-extension,  
    global-attributes,  
}  
  
resource-entry = {  
    type => text,  
    * $$resource-extension,  
    global-attributes,  
}  
  
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 2.4, Paragraph 2.
- * 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 (if available). This item maps to `'/SoftwareIdentity/(Payload|Evidence)/File/@version'` in [SWID].
- * hash (index 7): A hash of the file as described in Section 2.9.1.
- * 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 (as indicated in the location value in the evidence entry map) 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]. This item maps to `'/SoftwareIdentity/(Payload|Evidence)/(File|Directory)/@name'` in [SWID].
- * root (index 25): A host-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. This is a construction within the CDDL definition of CoSWID to support shared syntax and does not appear in [SWID].
- * 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]. This item maps to `'/SoftwareIdentity/(Payload|Evidence)/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 human-readable 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,  
    * $$payload-extension,  
    global-attributes,  
}
```

The following describes each child item of this group.

- * `global-attributes`: The `global-attributes` group described in Section 2.4, Paragraph 2.
- * `resource-collection`: The `resource-collection` group described in Section 2.9.2.
- * `$$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 => integer-time,  
  ? device-id => text,  
  ? location => text,  
  * $$evidence-extension,  
  global-attributes,  
}
```

```
date = 35  
device-id = 36
```

The following describes each child item of this group.

- * `global-attributes`: The `global-attributes` group described in Section 2.4, Paragraph 2.
- * `resource-collection`: The `resource-collection` group described in Section 2.9.2.
- * `date` (index 35): The date and time the information was collected pertaining to the evidence item.
- * `device-id` (index 36): The endpoint's string identifier from which the evidence was collected.
- * `location` (index 23): The absolute filepath of the location of the CoSWID tag generated as evidence. (Location values in `filesystem-items` in the payload can be expressed relative to this location.)
- * `$$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 => one-or-more<software-meta-entry>,
entity => one-or-more<entity-entry>,
? link => one-or-more<link-entry>,
? payload-or-evidence,
* $$coswid-extension,
global-attributes,
}

payload-or-evidence //= ( payload => payload-entry )
payload-or-evidence //= ( evidence => evidence-entry )

any-uri = uri
label = text / int

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

any-attribute = (
  label => one-or-more<text> / one-or-more<int>
)

one-or-more<T> = T / [ 2* T ]

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 => one-or-more<$role>,
  ? thumbprint => hash-entry,
  * $$entity-extension,
  global-attributes,
}
```

```
$role /= tag-creator
$role /= software-creator
$role /= aggregator
$role /= distributor
$role /= licenser
$role /= maintainer
$role /= int / text

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

$ownership /= shared
$ownership /= private
$ownership /= abandon
$ownership /= int / 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 /= int / 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 / bstr .size 16,
? persistent-id => text,
? product => text,
? product-family => text,
? revision => text,
? summary => text,
? unspsc-code => text,
? unspsc-version => text,
* $$software-meta-extension,
global-attributes,
}

path-elements-group = ( ? directory => one-or-more<directory-entry>,
                        ? file => one-or-more<file-entry>,
                        )

resource-collection = (
  path-elements-group,
  ? process => one-or-more<process-entry>,
  ? resource => one-or-more<resource-entry>,
  * $$resource-collection-extension,
)

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

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

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

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

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

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

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

integer-time = #6.1(int)

; "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
abandon=1
private=2
shared=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

This section defines a number of kinds of indexed label values that are maintained in a registry each (Section 6). These values are represented as positive integers. In each registry, the value 0 is marked as Reserved.

4.1. Version Scheme

The following table contains a set of values for use in the concise-swid-tag group's version-scheme item. Version Scheme Name strings 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 decimal integers (e.g., 1.2.3, 1.2.3.4.5.6.7, 1.4.5, 1.21) |
| 2 | multipartnumeric+suffix | Numbers separated by dots, where the numbers are interpreted as decimal integers with an additional textual suffix (e.g., 1.2.3a) |
| 3 | alphanumeric | Strictly a string, no interpretation as number |
| 4 | decimal | A single decimal floating point number |
| 16384 | semver | A semantic version as defined by [SWID]. Also see the [SEMVER] specification for more information |

Table 3: Version Scheme Values

multipartnumeric and the numbers part of multipartnumeric+suffix are interpreted as a sequence of numbers and are sorted in lexicographical order by these numbers (i.e., not by the digits in the numbers) and then the textual suffix (for multipartnumeric+suffix). Alphanumeric strings are sorted lexicographically as character strings. Decimal version numbers are interpreted as a single floating point number (e.g., 1.25 is less than 1.3).

The values above are registered in the IANA "Software Tag Version Scheme Values" registry defined in Section 6.2.4. Additional entries will likely be registered over time in this registry.

A CoSWID producer that is aware of the version scheme that has been used to select the version value, SHOULD include the optional version-scheme item to avoid semantic ambiguity. If the CoSWID producer does not have this information, it SHOULD omit the version-scheme item. The following heuristics can be used by a CoSWID consumer, based on the version schemes' partially overlapping value spaces:

- * "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 assumed.
- * "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 assumed.
- * "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 assumed instead of "alphanumeric".

Note that these heuristics are imperfect and can guess wrong, which is the reason the version-scheme item SHOULD be included by the producer.

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 6.2.5. Additional values will likely be registered over time.

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 6.2.6. Additional values will likely be registered over time.

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 6.2.7. Additional values will likely be registered over time.

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 6.2.8. Additional values will likely be registered over time.

5. URI Schemes

This specification defines the following URI schemes for use in CoSWID and to provide interoperability with schemes used in [SWID].

Note: These URI schemes are used in [SWID] without an IANA registration. The present specification ensures that these URI schemes are properly defined going forward.

// RFC Ed.: throughout this section, please replace RFC-AAAA with the
// RFC number of this specification and remove this note.

5.1. "swid" URI Scheme

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 2.7. Since this scheme is used both 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. In Section 6.6.1, the scheme "swid" is registered as a 'permanent' scheme for that purpose.

URIs specifying the "swid" scheme are 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 URIs with this scheme.

For URIs that use the "swid" scheme, the scheme specific part MUST consist of a referenced software tag's tag-id. This tag-id MUST be URI encoded according to [RFC3986] Section 2.1.

The following expression is a valid example:

```
swid:2df9de35-0aff-4a86-ace6-f7dddddade4c
```

5.2. "swidpath" URI Scheme

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 2.7. The scheme named "swidpath" is used for this purpose in [SWID], but not registered. To enable usage without fear of conflicts with current or future actual schemes, the present document registers it as a 'permanent' scheme for that purpose (see Section 6.6.2).

URIs specifying the "swidpath" scheme are used to filter tags out of a base collection, so that matching tags are included in the identified tag collection. The XPath expression [W3C.REC-xpath20-20101214] references the data that must be found in a given software tag out of base collection for that tag to be considered a matching tag. Tags to be evaluated (the base collection) include all tags in the context of where the "swidpath URI" 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 a URI with this scheme.

For URIs that use the "swidpath" scheme, the following requirements apply:

- * The scheme specific part MUST be an XPath expression as defined by [W3C.REC-xpath20-20101214]. The included XPath expression will be URI encoded according to [RFC3986] Section 2.1.
- * This XPath is evaluated over SWID tags, or COSWID tags transformed into SWID tags, found on a system. A given tag MUST be considered a match if the XPath evaluation result value has an effective boolean value of "true" according to [W3C.REC-xpath20-20101214] Section 2.4.3.

6. 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.

6.1. 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 [BCP26] as follows:

| Range | Registration Procedures |
|------------------|-------------------------------------|
| 0-32767 | Standards Action with Expert Review |
| 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 |
| 30 | Unassigned | |

| | | |
|----|---------------------------|----------|
| 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 Initial Registrations

6.2. 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. While neither the CoSWID nor SWID specification is subordinate to the other and will evolve as their respective standards group chooses, there is value in supporting alignment between the two standards. Shared use of common code points, as spelled out in these registries, will facilitate this alignment, hence the intent for shared use of these registries and the decision to use "swid" (rather than "coswid") in registry names.

6.2.1. Registration Procedures

The following registries allow for the registration of index values and names. New registrations will be permitted through either a Standards Action with Expert Review policy or a Specification Required policy [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.

6.2.2. 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:

domainprefix/name

Where both "domainprefix" and "name" MUST each be either an NR-LDH label or a U-label as defined by [RFC5890], and "name" also MUST be a unique name within the namespace defined by the "domainprefix". Use of a prefix in this way allows for a name to be used in the private use range. This is consistent with the guidance in [BCP178].

6.2.3. Expert Review Criteria

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

- * 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 MUST 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.
- * 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.
- * 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.

6.2.4. 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 6.2.1 with the following associated ranges:

| Range | Registration Procedures |
|-------------|-------------------------------------|
| 0-16383 | Standards Action with Expert Review |
| 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 | See Section 4.1 |
| 16385-65535 | Unassigned | |

Table 11: CoSWID Version Scheme Initial Registrations

Registrations MUST conform to the expert review criteria defined in Section 6.2.3.

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]. See also Section 4.1.

6.2.5. 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 6.2.1 with the following associated ranges:

| Range | Registration Procedures |
|---------|-------------------------------------|
| 0-127 | Standards Action with Expert Review |
| 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 criteria defined in Section 6.2.3.

6.2.6. 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 6.2.1 with the following associated ranges:

| | |
|---------|-------------------------------------|
| Range | Registration Procedures |
| 0-127 | Standards Action with Expert Review |
| 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 Initial Registrations

Registrations MUST conform to the expert review criteria defined in Section 6.2.3.

6.2.7. 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 6.2.1 with the following associated ranges:

| Range | Registration Procedures |
|-------------|-------------------------------------|
| 0-32767 | Standards Action with Expert Review |
| 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 criteria defined in Section 6.2.3.

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.

6.2.8. 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 6.2.1 with the following associated ranges:

| Range | Registration Procedures |
|---------|-------------------------------------|
| 0-127 | Standards Action with Expert Review |
| 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 criteria defined in Section 6.2.3.

6.3. swid+cbor Media Type Registration

IANA is requested to add the following to the IANA "Media Types" registry [IANA.media-types].

Type name: application

Subtype name: swid+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: Binary (encoded as CBOR [RFC8949]). See RFC-AAAA for details.

Security considerations: See Section 9 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: The syntax and semantics of fragment identifiers specified for "application/swid+cbor" are as specified for "application/cbor". (At publication of RFC-AAAA, there is no fragment identification syntax defined for "application/cbor".)

Additional information:

Magic number(s): if tagged, first five bytes in hex: da 53 57 49 44 (see Section 8 in RFC-AAAA)

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: IESG <iesg@ietf.org>

Intended usage: COMMON

Restrictions on usage: None

Author: Henk Birkholz <henk.birkholz@sit.fraunhofer.de>

Change controller: IESG

6.4. 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] [IANA.core-parameters]:

| Media type | Encoding | ID | Reference |
|-----------------------|----------|------|-----------|
| application/swid+cbor | - | TBD1 | RFC-AAAA |

Table 20: CoAP Content-Format IDs

6.5. CBOR Tag Registration

IANA is requested to allocate a tag in the "CBOR Tags" registry [IANA.cbor-tags], preferably with the specific value requested:

| Tag | Data Item | Semantics |
|------------|-----------|---|
| 1398229316 | map | Concise Software Identifier (CoSWID) [RFC-AAAA] |

Table 21: CoSWID CBOR Tag

6.6. 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.

URI schemes are registered within the "Uniform Resource Identifier (URI) Schemes" registry maintained at [IANA.uri-schemes].

6.6.1. URI-scheme swid

IANA is requested to register the URI scheme "swid". This registration request complies with [RFC7595].

Scheme name:
swid

Status:
Permanent

Applications/protocols that use this scheme name:
Applications that require Software-IDs (SWIDs) or Concise Software-IDs (CoSWIDs); see Section 5.1 of RFC-AAAA.

Contact:
IETF Chair <chair@ietf.org>

Change controller:
IESG <iesg@ietf.org>

Reference:
Section 5.1 in RFC-AAAA

6.6.2. URI-scheme swidpath

IANA is requested to register the URI scheme "swidpath". This registration request complies with [RFC7595].

Scheme name:
swidpath

Status:
Permanent

Applications/protocols that use this scheme name:
Applications that require Software-IDs (SWIDs) or Concise Software-IDs (CoSWIDs); see Section 5.2 of RFC-AAAA.

Contact:
IETF Chair <chair@ietf.org>

Change controller:
IESG <iesg@ietf.org>

Reference:
Section 5.2 in RFC-AAAA

6.7. 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] [IANA.pa-tnc-parameters] to support CoSWID use in SWIMA as follows:

Pen: 0

Integer: TBD2

Name: Concise Software Identifier (CoSWID)

Reference: RFC-AAAA

Deriving Software Identifiers:

A Software Identifier generated from a CoSWID tag is expressed as a concatenation of the form in [RFC5234] as follows:

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.

7. Signed CoSWID Tags

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.

Signing CoSWID tags follows the procedures defined in CBOR Object Signing and Encryption [I-D.ietf-cose-rfc8152bis-struct]. A CoSWID tag MUST be wrapped in a COSE Signature structure, either COSE_Sign1 or COSE_Sign. In the first case, a Single Signer Data Object (COSE_Sign1) contains a single signature and MUST be signed by the tag creator. The following CDDL specification defines a restrictive subset of COSE header parameters that MUST be used in the protected header in this case.

```
<CODE BEGINS>
COSE-Sign1-coswid<payload> = [
    protected: bstr .cbor protected-signed-coswid-header,
    unprotected: unprotected-signed-coswid-header,
    payload: bstr .cbor payload,
    signature: bstr,
]

cose-label = int / tstr
cose-values = any

protected-signed-coswid-header = {
    1 => int,                ; algorithm identifier
    3 => "application/swid+cbor",
    * cose-label => cose-values,
}

unprotected-signed-coswid-header = {
    * cose-label => cose-values,
}
<CODE ENDS>
```

The COSE_Sign structure allows for more than one signature, one of which MUST be issued by the tag creator, to be applied to a CoSWID tag and MAY be used. The corresponding usage scenarios are domain-specific and require well-specified application guidance.

```
<CODE BEGINS>
COSE-Sign-coswid<payload> = [
    protected: bstr .cbor protected-signed-coswid-header1,
    unprotected: unprotected-signed-coswid-header,
    payload: bstr .cbor payload,
    signature: [ * COSE_Signature ],
]

protected-signed-coswid-header1 = {
    3 => "application/swid+cbor",
    * cose-label => cose-values,
}

protected-signature-coswid-header = {
    1 => int,                                ; algorithm identifier
    * cose-label => cose-values,
}

unprotected-sign-coswid-header = {
    * cose-label => cose-values,
}

COSE_Signature = [
    protected: bstr .cbor protected-signature-coswid-header,
    unprotected: unprotected-sign-coswid-header,
    signature : bstr
]
<CODE ENDS>
```

Additionally, the COSE Header counter signature MAY be used as an attribute in the unprotected header map of the COSE envelope of a CoSWID [I-D.ietf-cose-countersign]. 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).

A CoSWID MUST be signed, using the above mechanism, to protect the integrity of the CoSWID tag. See the security considerations (in Section 9) for more information on why a signed CoSWID is valuable in most cases.

8. CBOR-Tagged CoSWID Tags

This specification allows for tagged and untagged CBOR data items that are CoSWID tags. Consecutively, the CBOR tag for CoSWID tags defined in Table 21 SHOULD be used in conjunction with CBOR data items that are a CoSWID tags. Other CBOR tags MUST NOT be used with a CBOR data item that is a CoSWID tag. If tagged, both signed and unsigned CoSWID tags MUST use the CoSWID CBOR tag. In case a signed CoSWID is tagged, a CoSWID CBOR tag MUST be appended before the COSE envelope whether it is a COSE_Untagged_Message or a COSE_Tagged_Message. In case an unsigned CoSWID is tagged, a CoSWID CBOR tag MUST be appended before the CBOR data item that is the CoSWID tag.

```
<CODE BEGINS>
coswid = unsigned-coswid / signed-coswid
unsigned-coswid = concise-swid-tag / tagged-coswid<concise-swid-tag>
signed-coswid1 = signed-coswid-for<unsigned-coswid>
signed-coswid = signed-coswid1 / tagged-coswid<signed-coswid1>

tagged-coswid<T> = #6.1398229316(T)

signed-coswid-for<payload> = #6.18(COSE-Sign1-coswid<payload>)
    / #6.98(COSE-Sign-coswid<payload>)
<CODE ENDS>
```

This specification allows for a tagged CoSWID tag to reside in a COSE envelope that is also tagged with a CoSWID CBOR tag. In cases where a tag creator is not a signer (e.g., hand-offs between entities in a trusted portion of a supply-chain), retaining CBOR tags attached to unsigned CoSWID tags can be of great use. Nevertheless, redundant use of tags SHOULD be avoided when possible.

9. Security Considerations

The following security considerations for use of CoSWID tags focus on:

- * ensuring the integrity and authenticity of a CoSWID tag
- * the application of CoSWID tags to address security challenges related to unmanaged or unpatched software
- * reducing the potential for unintended disclosure of a device's software load

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 supplier of the software component, who is expected to be an expert in their own software. Thus, authoritative CoSWID tags can represent authoritative information about the software component. The degree to which this information can be trusted depends on the tag's chain of custody and the ability to verify a signature provided by the supplier if present in the CoSWID tag. The provisioning and validation of CoSWID tags are handled by local policy and is outside the scope of this document.

A signed CoSWID tag (see Section 7) whose signature has been validated can be relied upon to be unchanged since it was signed. By contrast, the data contained in unsigned tags can be altered by any user or process with write-access to the tag. To support signature validation, there is the need to associate the right key with the software provider or party originating the signature in a secure way. This operation is application specific and needs to be addressed by the application or a user of the application; a specific approach for which is out-of-scope for this document.

When an authoritative tag is signed, the originator of the signature can be verified. 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 public key enabling the validation of a tag signature can realize the assessment of trustworthiness of an authoritative tag. Verifying that the software provider is the signer is a different matter. This requires an association between the signature and the tag's entity item associated corresponding to the software provider. No mechanism is defined in this draft to make this association; therefore, this association will need to be handled by local policy. As always, the validity of a signature does not imply veracity of the signed statements: anyone can sign assertions such that the software is from a specific software-creator or that a specific persistent-id applies; policy needs to be applied to evaluate these statements and to determine their suitability for a specific use.

Loss of control of signing credentials used to sign CoSWID tags would create doubt about the authenticity and integrity of any CoSWID tags signed using the compromised keys. In such cases, the legitimate tag signer (namely, the software provider for an authoritative CoSWID tag) can employ uncompromised signing credentials to create a new signature on the original tag. The tag version number would not be incremented since the tag itself was not modified. Consumers of CoSWID tags would need to validate the tag using the new credentials and would also need to make use of revocation information available

for the compromised credentials to avoid validating tags signed with them. The process for doing this is beyond the scope of this specification.

The CoSWID format allows the use of hash values without an accompanying hash algorithm identifier. This exposes the tags to some risk of cross-algorithm attacks. We believe that this can become a practical problem only if some implementations allow the use of insecure hash algorithms. Since it may not become known immediately when an algorithm becomes insecure, this leads to a strong recommendation to only include support for hash algorithms that are generally considered secure, and not just marginally so.

CoSWID tags are intended to contain public information about software components and, as such, the contents of a CoSWID tag (as opposed to the set of tags that apply to the endpoint, see below) does not need to be protected against unintended disclosure on an endpoint. Converse, generators of CoSWID tags need to ensure that only public information is disclosed. Entitlement Keys are an example for information where particular care is required; tag authors are advised not to record unprotected, private software license keys in this field.

CoSWID tags are intended to be easily discoverable by authorized applications and users on an endpoint in order to make it easy to determine the tagged software load. Access to the collection of an endpoint's CoSWID tags needs to be appropriately controlled to authorized applications and users using an appropriate access control mechanism.

Since the tag-id of a CoSWID tag can be used as a global index value, failure to ensure the tag-id's uniqueness can cause collisions or ambiguity in CoSWID tags that are retrieved or processed using this identifier. CoSWID is designed to not require a registry of identifiers. As a result, CoSWID requires the tag creator to employ a method of generating a unique tag identifier. Specific methods of generating a unique identifier are beyond the scope of this specification. A collision in tag-ids may result in false positives/negatives in software integrity checks or mis-identification of installed software, undermining CoSWID use cases such as vulnerability identification, software inventory, etc. If such a collision is detected, then the tag consumer may want to contact the maintainer of the CoSWID to have them issue a correction addressing the collision; however, this also discloses to the maintainer that the consumer has the other tag with the given tag-id in their database. More generally speaking, a tag consumer needs to be robust against such collisions lest the collision become a viable attack vector.

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 CoSWID 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 CoSWID 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 a preliminary understanding of that endpoint's software inventory.

As CoSWID tags do not expire, inhibiting new CoSWID tags from reaching an intended consumer would render that consumer stuck with outdated information, potentially leaving associated vulnerabilities or weaknesses unmitigated. Therefore, a CoSWID tag consumer should actively check for updated tag-versions via more than one means.

This specification makes use of relative paths (e.g., filesystem paths) in several places. A signed CoSWID tag cannot make use of these to derive information that is considered to be covered under the signature. Typically, relative file system paths will be used to identify targets for an installation, not sources of tag information.

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 authorized applications and users on 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, loop detection, and signature verification are ways that implementations can address this concern.

More generally speaking, the security considerations of [RFC8949], [I-D.ietf-cose-rfc8152bis-struct], and [I-D.ietf-cose-countersign] apply.

10. Privacy Consideration

As noted in Section 9, collected information about an endpoint's software load, such as what might be represented by an endpoint's CoSWID tag collection, could be used to identify vulnerable software for attack. Collections of endpoint software information also can have privacy implications for users. The set of application a user installs can give clues to personal matters such as political affiliation, banking and investments, gender, sexual orientation, medical concerns, etc. While the collection of CoSWID tags on an endpoint wouldn't increase the privacy risk (since a party able to view those tags could also view the applications themselves), if

those CoSWID tags are gathered and stored in a repository somewhere, visibility into the repository now also gives visibility into a user's application collection. For this reason, repositories of collected CoSWID tags not only need to be protected against collection by malicious parties, but even authorized parties will need to be vetted and made aware of privacy responsibilities associated with having access to this information. Likewise, users should be made aware that their software inventories are being collected from endpoints. Furthermore, when collected and stored by authorized parties or systems, the inventory data needs to be protected as both security and privacy sensitive information.

11. Change Log

This section is to be removed before publishing as an RFC.

[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
- * Renamed socket name in software-meta to be consistent in naming
- * Aligned excerpt examples in I-D text with full CDDL
- * Fixed titles where title was referring to group instead of map
- * Added missing date in SEMVER
- * Fixed root cardinality for file and directory, etc.
- * Transformed path-elements-entry from map to group for re-usability
- * Scrubbed IANA Section
- * Removed redundant supplemental rule
- * Aligned discrepancy with ISO spec.

- * Addressed comments on typos.
- * Fixed kramdown nits and BCP reference.
- * Addressed comments from WGLC reviewers.

Changes in version 12:

- * Addressed a bunch of minor editorial issues based on WGLC feedback.
- * Added text about the use of UTF-8 in CoSWID.
- * Adjusted tag-id to allow for a UUID to be provided as a bstr.
- * Cleaned up descriptions of index ranges throughout the document, removing discussion of 8 bit, 16 bit, etc.
- * Adjusted discussion of private use ranges to use negative integer values and to be more clear throughout the document.
- * Added discussion around resolving overlapping value spaces for version schemes.
- * Added a set of expert review criteria for new IANA registries created by this document.
- * Added new registrations for the "swid" and "swidpath" URI schemes, and for using CoSWID with SWIMA.

Changes from version 03 to version 11:

- * Reduced representation complexity of the media-entry type and removed the Section describing the older data structure.
- * Added more signature schemes from COSE
- * Included a minimal required set of normative language
- * Reordering of attribute name to integer label by priority according to semantics.
- * Added an IANA registry for CoSWID items supporting future extension.
- * Cleaned up IANA registrations, fixing some inconsistencies in the table labels.

- * Added additional CDDL sockets for resource collection entries providing for additional extension points to address future SWID/CoSWID extensions.
- * Updated Section on extension points to address new CDDL sockets and to reference the new IANA registry for items.
- * Removed unused references and added new references to address placeholder comments.
- * Added table with semantics for the link ownership item.
- * Clarified language, made term use more consistent, fixed references, and replacing lowercase RFC2119 keywords.

Changes from version 02 to version 03:

- * Updated core CDDL including the CDDL design pattern according to RFC 8428.

Changes from version 01 to version 02:

- * Enforced a more strict separation between the core CoSWID definition and additional usage by moving content to corresponding appendices.
- * Removed artifacts inherited from the reference schema provided by ISO (e.g., NMTOKEN(S))
- * Simplified the core data definition by removing group and type choices where possible
- * Minor reordering of map members
- * Added a first extension point to address requested flexibility for extensions beyond the any-element

Changes from version 00 to version 01:

- * Ambiguity between evidence and payload eliminated by introducing explicit members (while still
- * allowing for "empty" SWID tags)
- * Added a relatively restrictive COSE envelope using cose_sign1 to define signed CoSWID (single signer only, at the moment)

- * 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:

- * Removed redundant any-attributes originating from the ISO-19770-2:2015 XML schema definition
- * Fixed broken multi-map members
- * Introduced a more restrictive item (any-element-map) to represent custom maps, increased restriction on types for the any-attribute, accordingly
- * Fixed X.1520 reference
- * Minor type changes of some attributes (e.g., NMTOKENS)
- * Added semantic differentiation of various name types (e.g. fs-name)

Changes from version 06 to version 07:

- * Added type choices/enumerations based on textual definitions in 19770-2:2015
- * Added value registry request
- * 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.
- * Added a Section describing the CoSWID model in detail.
- * Created IANA registries for entity-role and version-scheme

Changes from version 02 to version 03:

- * Updated CDDL to allow for a choice between a payload or evidence
- * Re-index label values in the CDDL.
- * Added item definitions
- * Updated references for COSE, CBOR Web Token, and CDDL.

Changes from version 01 to version 02:

- * Added extensions for Firmware and CoSWID use as Reference Integrity Measurements (CoSWID RIM)
- * Changes meta handling in CDDL from use of an explicit use of items to a more flexible unconstrained collection of items.
- * Added Sections discussing use of COSE Signatures and CBOR Web Tokens

Changes from version 00 to version 01:

- * Added CWT usage for absolute SWID paths on a device
- * Fixed cardinality of type-choices including arrays
- * Included first iteration of firmware resource-collection

12. References

12.1. 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>>.
- [I-D.ietf-cose-countersign] Schaad, J. and R. Housley, "CBOR Object Signing and Encryption (COSE): Countersignatures", Work in Progress, Internet-Draft, draft-ietf-cose-countersign-05, 23 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-cose-countersign-05.txt>>.
- [I-D.ietf-cose-rfc8152bis-struct] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>>.

- [IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags",
<<http://www.iana.org/assignments/cbor-tags>>.
- [IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE)
Parameters",
<<http://www.iana.org/assignments/core-parameters>>.
- [IANA.media-types]
IANA, "Media Types",
<<http://www.iana.org/assignments/media-types>>.
- [IANA.named-information]
IANA, "Named Information",
<<http://www.iana.org/assignments/named-information>>.
- [IANA.pa-tnc-parameters]
IANA, "Posture Attribute (PA) Protocol Compatible with
Trusted Network Connect (TNC) Parameters",
<<http://www.iana.org/assignments/pa-tnc-parameters>>.
- [IANA.uri-schemes]
IANA, "Uniform Resource Identifier (URI) Schemes",
<<http://www.iana.org/assignments/uri-schemes>>.
- [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>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
Specifications: ABNF", STD 68, RFC 5234,
DOI 10.17487/RFC5234, January 2008,
<<https://www.rfc-editor.org/info/rfc5234>>.

- [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>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [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>>.
- [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>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.
- [SAM] "Information technology - Software asset management - Part 5: Overview and vocabulary", ISO/IEC 19770-5:2015, 15 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, 1 October 2015.
- [UNSPSC] "United Nations Standard Products and Services Code", 26 October 2020, <<https://www.unspsc.org/>>.
- [W3C.REC-css3-mediaqueries-20120619]
Rivoal, F., "Media Queries", World Wide Web Consortium Recommendation REC-css3-mediaqueries-20120619, 19 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, 28 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, 14 December 2010, <<https://www.w3.org/TR/2010/REC-xpath20-20101214>>.

12.2. Informative References

- [CamelCase]
"UpperCamelCase", 29 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", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-15, 8 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-rats-architecture-15.txt>>.
- [KebabCase]
"KebabCase", 18 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>>.
- [RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <<https://www.rfc-editor.org/info/rfc7595>>.
- [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.A., 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>>.
- [X.1520] "Recommendation ITU-T X.1520 (2014), Common vulnerabilities and exposures", 20 April 2011.

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 for the careful reviews provided by the IESG reviewers. Special thanks go to Benjamin Kaduk.

Contributors

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

Carsten Bormann contributed to the CDDL specifications and the IANA considerations.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@sit.fraunhofer.de

Jessica Fitzgerald-McKay
National Security Agency
9800 Savage Road
Ft. Meade, Maryland
United States of America
Email: jmfitz2@cyber.nsa.gov

Charles Schmidt
The MITRE Corporation
202 Burlington Road
Bedford, Massachusetts 01730
United States of America
Email: cmschmidt@mitre.org

David Waltermire
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland 20877
United States of America
Email: david.waltermire@nist.gov

SACM
Internet-Draft
Intended status: Best Current Practice
Expires: December 23, 2019

D. Haynes
The MITRE Corporation
J. Fitzgerald-McKay
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

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

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

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

This Internet-Draft will expire on December 23, 2019.

Copyright Notice

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

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

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 4 |
| 2. Terminology | 4 |
| 3. Endpoint Posture Collection Profile | 5 |
| 3.1. Components | 6 |
| 3.1.1. Endpoint | 7 |
| 3.1.1.1. Posture Collection Engine | 7 |
| 3.1.2. Posture Manager | 8 |
| 3.1.2.1. Posture Collection Manager | 8 |
| 3.1.3. Repository | 8 |
| 3.1.4. Evaluator | 9 |
| 3.1.5. Orchestrator | 9 |
| 3.1.6. Administrative Interface and API | 9 |
| 3.2. Transactions | 9 |
| 3.2.1. Provisioning | 10 |
| 3.2.2. Discovery and Validation | 10 |
| 3.2.3. Event Driven Collection | 10 |
| 3.2.4. Querying the Endpoint | 10 |
| 3.2.5. Data Storage | 10 |
| 3.2.6. Data Sharing | 11 |
| 4. IETF NEA EPCP Implementation for Traditional Endpoints | 11 |
| 4.1. Endpoint Provisioning | 13 |
| 4.2. Endpoint | 13 |
| 4.2.1. Posture Collector | 14 |
| 4.2.2. Posture Broker Client | 14 |
| 4.2.3. Posture Transport Client | 14 |
| 4.3. Posture Manager | 14 |
| 4.3.1. Posture Validator | 14 |
| 4.3.2. Posture Broker Server | 14 |
| 4.3.3. Posture Transport Server | 14 |
| 4.4. Repository | 15 |
| 4.5. IETF SACM SWAM Extension to the IETF NEA EPCP Implementation | 15 |
| 4.5.1. Endpoint Pre-Provisioning | 15 |
| 4.5.2. SWID Tags | 15 |
| 4.5.3. SWID Posture Collectors and Posture Validators | 15 |
| 4.5.3.1. The SWID Posture Collector | 16 |
| 4.5.3.2. The SWID Posture Validator | 16 |
| 4.5.4. Repository | 17 |
| 5. IETF NETCONF EPCP Implementation for Network Device Endpoints | 17 |
| 5.1. Endpoint Provisioning | 18 |

| | | |
|-------------|--|----|
| 5.2. | Posture Manager Provisioning | 18 |
| 5.3. | Endpoint | 18 |
| 5.3.1. | Datastore | 18 |
| 5.4. | Posture Manager | 19 |
| 5.5. | Repository | 19 |
| 6. | Future Work | 19 |
| 7. | Acknowledgements | 20 |
| 8. | IANA Considerations | 22 |
| 9. | Security Considerations | 22 |
| 9.1. | Threat Model | 22 |
| 9.1.1. | Endpoint Attacks | 23 |
| 9.1.2. | Network Attacks | 23 |
| 9.1.3. | Posture Manager Attacks | 23 |
| 9.1.4. | Repository Attacks | 24 |
| 9.2. | Countermeasures | 25 |
| 9.2.1. | Countermeasures for Endpoint Attacks | 25 |
| 9.2.2. | Countermeasures for Network Attacks | 25 |
| 9.2.3. | Countermeasures for Posture Manager Attacks | 26 |
| 9.2.4. | Countermeasures for Repository Attacks | 26 |
| 10. | Privacy Considerations | 27 |
| 11. | References | 27 |
| 11.1. | Informative References | 27 |
| 11.2. | Normative References | 28 |
| Appendix A. | Rationale for an EPCP Solution | 30 |
| A.1. | Preventative Posture Assessments | 30 |
| A.2. | All Network-Connected Endpoints are Endpoints | 31 |
| A.3. | All Endpoints on the Network Must be Uniquely Identified | 31 |
| A.4. | Standardized Data Models | 31 |
| A.5. | Posture Information Must Be Stored | 32 |
| A.6. | Posture Information Can Be Shared | 32 |
| A.7. | Enterprise Asset Posture Information Belongs to the Enterprise | 32 |
| Appendix B. | EPCP Supported Use Cases and Non-Supported Use Cases | 33 |
| B.1. | Supported Use Cases | 33 |
| B.1.1. | Hardware Asset Management | 33 |
| B.1.2. | Software Asset Management | 33 |
| B.1.3. | Vulnerability Management | 34 |
| B.1.4. | Threat Detection and Analysis | 34 |
| B.2. | Non-Supported Use Cases | 34 |
| Appendix C. | Endpoint Posture Collection Profile Examples | 35 |
| C.1. | Continuous Posture Assessment of an Endpoint | 35 |
| C.1.1. | Change on Endpoint Triggers Posture Assessment | 35 |
| C.2. | Administrator Searches for Vulnerable Endpoints | 38 |
| Appendix D. | Change Log | 39 |
| D.1. | -04 to -05 | 39 |
| D.2. | -03 to -04 | 40 |
| D.3. | -02 to -03 | 40 |
| D.4. | -01 to -02 | 41 |

Haynes, et al. Expires December 23, 2019 [Page 3]

| | | | |
|--------------------|------------|-----------|----|
| D.5. | -00 to -01 | | 41 |
| D.6. | -01 to -02 | | 41 |
| D.7. | -02 to -00 | | 41 |
| D.8. | -00 to -01 | | 41 |
| Authors' Addresses | | | 41 |

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 be 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.

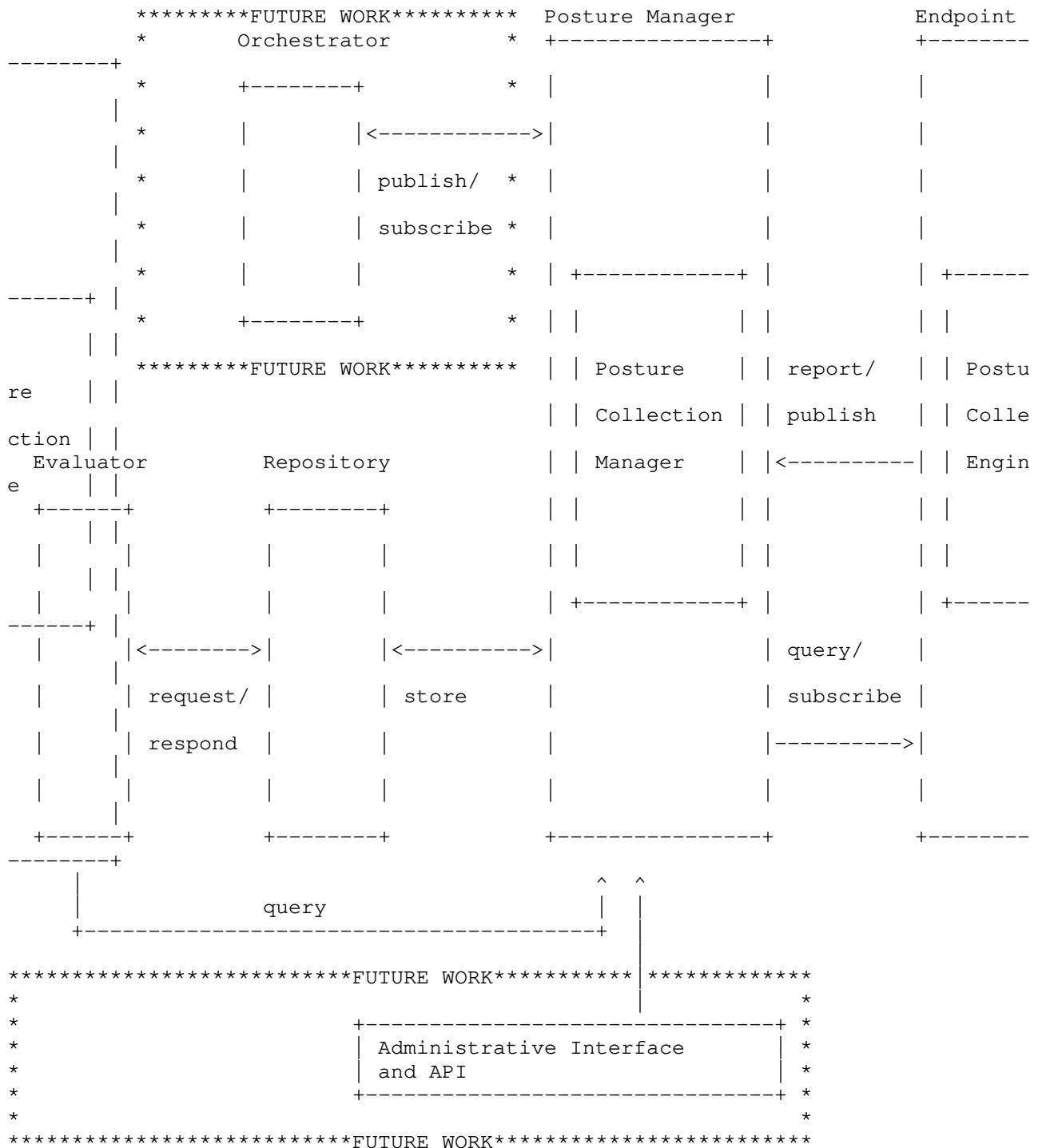


Figure 1: EPCP Components

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

Haynes, et al. Expires December 23, 2019 [Page 8]

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.

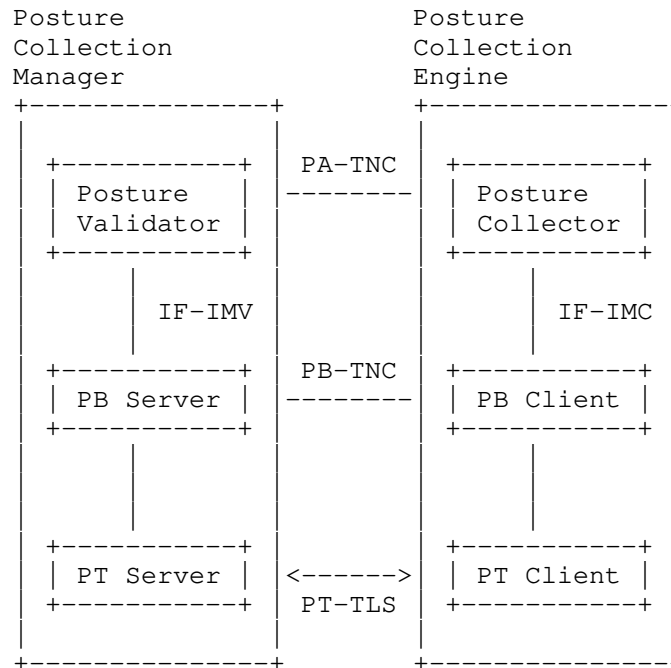


Figure 2: NEA Components

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:

- o the software installer; or
- o 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.

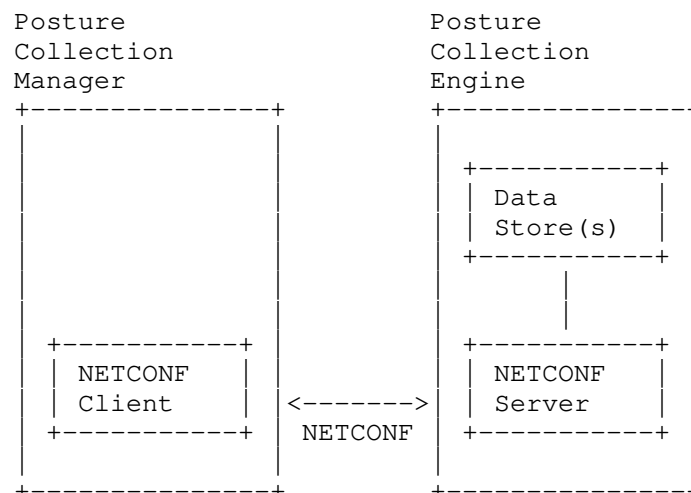


Figure 3: NETCONF Components

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.
- o Add support endpoint types beyond workstations, servers, and network infrastructure devices.
 - o Examine the integration of [I-D.ietf-mile-xmpp-grid].
 - o 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.
 - o 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).
 - o Retention of posture information on the target endpoint.
 - o Define an orchestrator component as well as publish/subscribe interface for it.
 - o Define an evaluator component as well as an interface for it.
 - o 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.

- o Compromise of endpoint: A compromised endpoint may report false information to confuse or even provide maliciously crafted information with a goal of infecting others.
- o 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.
- o 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:

- o Eavesdropping, modification, injection, replay, deletion;
- o Traffic analysis; and
- o 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.

- o **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.
- o **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.
- o **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.

- o **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.
- o **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.
- o **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.
- o **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

Haynes, et al. Expires December 23, 2019 [Page 26]

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:

- o what endpoints are connected to the network at any given time; and
- o 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:

- o vulnerability management: knowing what software is installed supports the ability to determine which endpoints contain vulnerable software and need to be patched.
- o 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:

- o 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.
- o 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

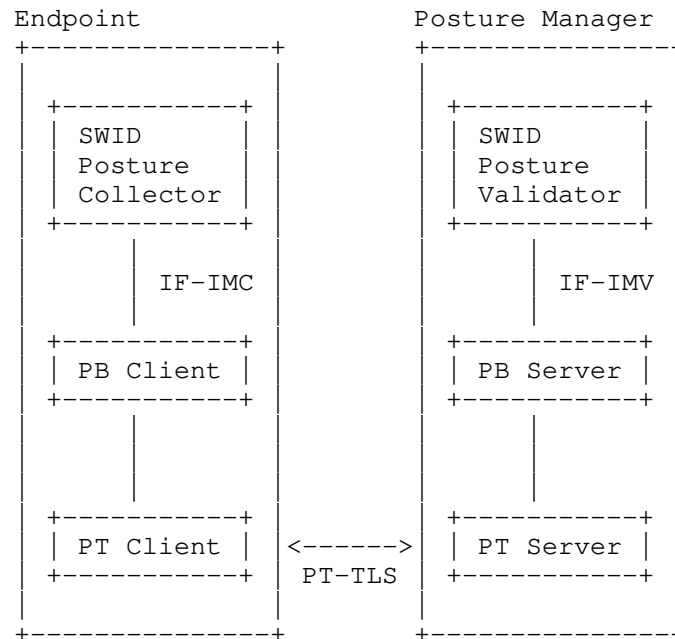


Figure 4: Continuous Posture Assessment of an Endpoint

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.

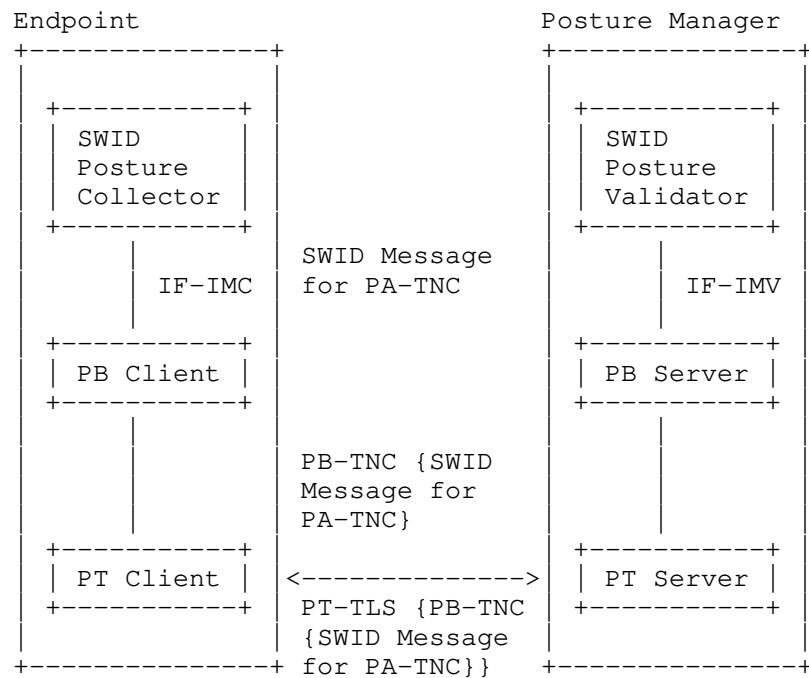


Figure 5: Compliance Protocol Encapsulation

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).

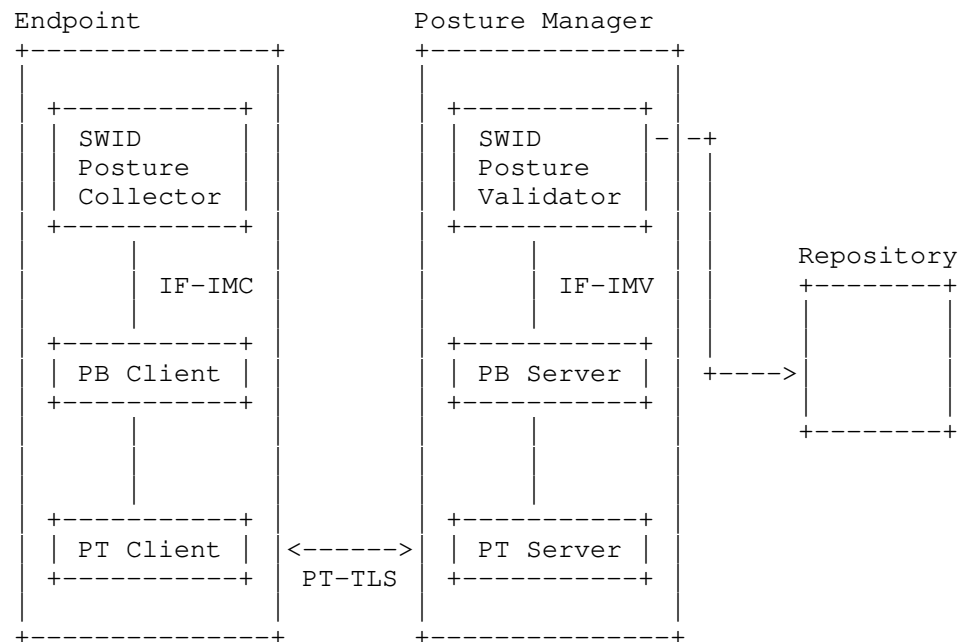


Figure 6: Storing SWIDs in the Repository

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.

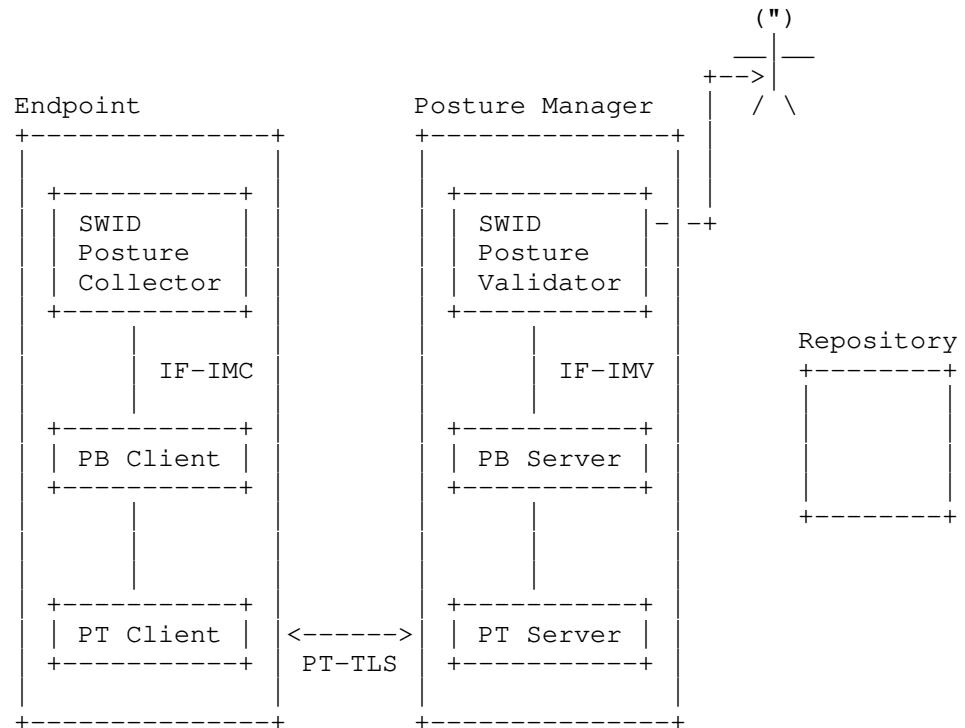


Figure 7: Server Alerts Network Admin

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.

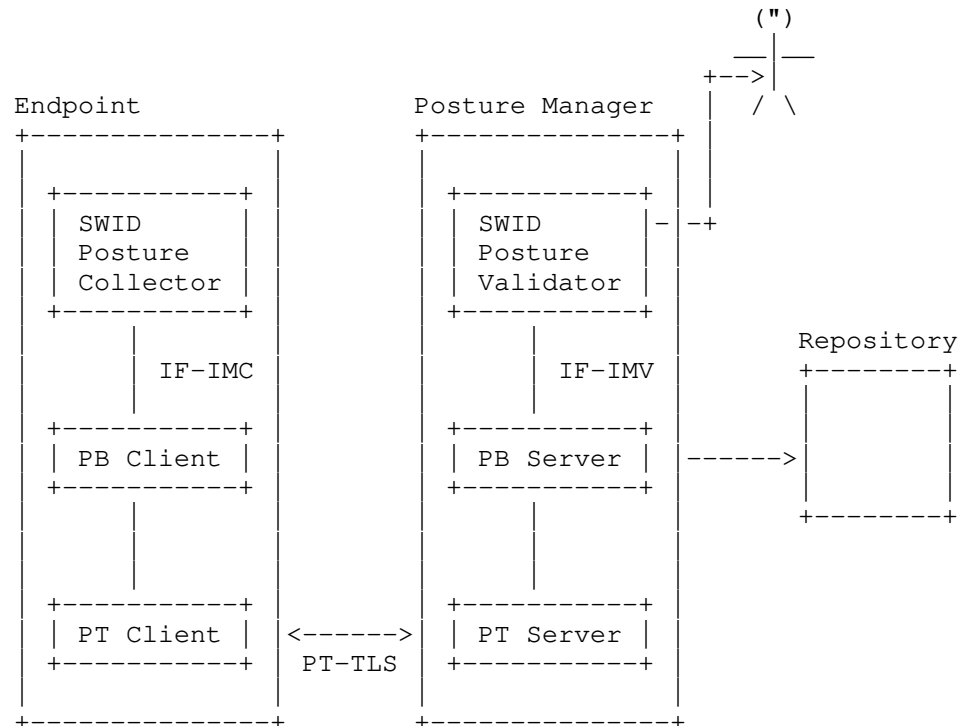


Figure 8: Admin Searches for Vulnerable Endpoints

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 Working Group
Internet-Draft

Intended status: Informational National Institute of Standards and Techno
Expires: September 22, 2018

D. Waltermire
S. Banghart
March 21, 2018

Definition of the ROLIE Software Descriptor Extension
draft-ietf-sacm-rolie-softwaredescriptor-02

Abstract

This document extends the Resource-Oriented Lightweight Information Exchange (ROLIE) core to add the information type category and related requirements needed to support Software Record and Software Inventory use cases. The 'software-descriptor' information type is defined as a ROLIE extension. Additional supporting requirements are also defined that describe the use of specific formats and link relations pertaining to the new information type.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 22, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. Information-type Extensions | 3 |
| 3.1. The "software-descriptor" information type | 4 |
| 4. rolie:property Extensions | 5 |
| 4.1. urn:ietf:params:rolie:property:swd:id | 5 |
| 4.2. urn:ietf:params:rolie:property:swd:swname | 5 |
| 5. Use of the rolie:format element | 5 |
| 5.1. The ISO SWID 2015 format | 5 |
| 5.1.1. Description | 5 |
| 5.1.2. Requirements | 6 |
| 5.2. The Concise SWID format | 6 |
| 5.2.1. Description | 7 |
| 5.2.2. Requirements | 7 |
| 6. atom:link Extensions | 8 |
| 7. IANA Considerations | 8 |
| 7.1. Media Type Registrations | 8 |
| 7.1.1. ISO SWID | 8 |
| 7.2. software-descriptor information-type | 9 |
| 7.3. swd:id property | 10 |
| 7.4. swd:swname property | 10 |
| 8. Security Considerations | 10 |
| 9. Privacy Considerations | 11 |
| 10. Normative References | 11 |
| Appendix A. Schema | 12 |
| Appendix B. Examples of Use | 12 |
| Authors' Addresses | 12 |

1. Introduction

This document defines an extension to the Resource-Oriented Lightweight Information Exchange (ROLIE) [RFC8322] protocol to support the publication of software descriptor information. Software descriptor information is information that characterizes:

an installable software package, or

information about static software components that may be installed by a software package or patch.

Software descriptor information includes identifying, versioning, software creation and publication, and file artifact information. Software descriptor information provides data about what might be

installed, but doesn't describe where or how a specific software installation is installed, configured, or executed.

Some possible use cases for Software descriptor information include:

- o Software providers can publish software descriptor information so that software researchers and users of software can understand the collection of software produced by a 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 value added information (e.g., software configuration checklist repositories) using identification and characterization information derived from software descriptor information.
- o End user organizations can consume sources of software descriptor information, and other 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, thru mechanisms such as RIM or other integrity measurements.

This document supports these use cases by describing the content requirements for Collections and Entries of software descriptor information that are to be published to or retrieved from a ROLIE repository. This document also discusses requirements around the use of atom:link and rolie:format.

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].

Definitions for some of the common computer security-related terminology used in this document can be found in Section 2 of [RFC5070].

3. Information-type Extensions

This document defines the following information type[s]:

3.1. The "software-descriptor" information type

The "software-descriptor" information type represents any 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: This software identification and characterization information can be provided by a large variety of data, but always describes software in a pre-installed state.

provides software installer metadata: This represents 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
- o Vendor and source information: Includes where the software was developed or distributed from, 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 OSES, intended purpose or audience, etc.

Note again that this list is not exhaustive, any information that in is the abstract realm of an incident should be classified under this information-type.

This information type does not include descriptions of running software, or state and configuration information that is associated with a software installation.

4. rolie:property Extensions

This document registers new valid rolie:property names as follows:

4.1. urn:ietf:params:rolie:property:swd:id

This property provides an exposure point for an identification field from the associated software descriptor. The value of this property SHOULD be uniquely identifying information generated from the software descriptor linked to by the entry's atom:content element. swd:id property values SHOULD have a one-to-one mapping to individual pieces of SWD content.

4.2. urn:ietf:params:rolie:property:swd:sname

This property provides an exposure point for the plain text name of the software being described. Due to the great variance in naming schemes, this property should be considered informative.

5. Use of the rolie:format element

This section defines usage guidance and additional requirements on the rolie:format element 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.

5.1. The ISO SWID 2015 format

5.1.1. 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].

5.1.2. Requirements

For an Entry to be considered as a "SWID Tag Entry", it MUST fulfill the following conditions:

- o 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.
- o The document linked to by the "href" attribute of the atom:content element is a SWID Tag as per ISO/IEC 19770-2:2015.

A "SWID Tag Entry" MUST conform to the following requirements:

- o The value of the "type" attribute of the atom:content element MUST be "application/swid2015+xml".
- o There MUST be one rolie:property with the "name" attribute equal to "urn:ietf:params:rolie:property:swd:id" and the "value" attribute exactly equal to the "<tagid>" element in the attached SWID Tag. This allows for ROLIE consumers to more easily search for SWID tags without needing to download the tag itself.
- o 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 field aids ROLIE consumers in search and filtering Entries.

5.2. The Concise SWID format

5.2.1. Description

The Concise SWID (COSWID) format is an alternative representation of the SWID Tag format using a Concise Binary Object Representation (CBOR) encoding. This provides the format with a reduced size that is more suitable for constrained devices. It provides the same features and attributes as are specified in ISO 19770-2:2015, plus:

- o a straight forward method to sign and encrypt using COSE, and
- o 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].

5.2.2. Requirements

For an Entry to be considered as a "COSWID Tag Entry", it MUST fulfill the following conditions:

- o 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.
- o The document linked to by the "href" attribute of the atom:content element is a COSWID Tag as per [I-D.ietf-sacm-coswid]

A "COSWID Tag Entry" MUST conform to the following requirements:

- o The value of the "type" attribute of the atom:content element MUST be "application/coswid+cbor".
- o There MUST be one rolie:property with the "name" attribute equal to "urn:ietf:params:rolie:property:swd:id" and the "value" attribute exactly equal to the "tag-id" element in the attached COSWID Tag. This allows for ROLIE consumers to more easily search for COSWID tags without needing to download the tag itself.
- o 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 "swid-name" element in the attached COSWID Tag. As above, this field aids ROLIE consumers in searching and filtering Entries.

6. 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. |
| patches | Links to a software descriptor resource that defines the software being patched by 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. |
| installs | Links to a software descriptor resource that defines the software that is installed by this software. |
| installationrecord | Links to a software descriptor resource that defines the software package that installs this software. |

Table 1: Link Relations for Resource-Oriented Lightweight Indicator Exchange

7. IANA Considerations

7.1. Media Type Registrations

7.1.1. ISO SWID

This document registers a MIME Type for the SWID Tag format. The registration is as follows

MIME media type name: application

MIME subtype name: swid2015+xml

Mandatory parameters: None.

Optional parameters: "charset": This parameter has semantics identical to the charset parameter of the "application/xml" media type as specified in [RFC3023].

Encoding considerations: Identical to those of "application/xml" as described in [RFC3023], Section 3.2.

Security considerations: As defined in this specification, and in [RFC8322]. In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [RFC3023], Section 10.

Interoperability considerations: There are no known interoperability issues.

Published specification: This specification.

Applications that use this media type: No known applications currently use this media type.

Additional information:

Magic number(s): As specified for "application/xml" in [RFC3023], Section 3.2.

File extension: .swidtag

Fragment identifiers: As specified for "application/xml" in [RFC3023], Section 5.

Base URI: As specified in [RFC3023], Section 6.

Macintosh File Type code: TEXT

Person and email address to contact for further information: Stephen Banghart <stephen.banghart@nist.gov>

Intended usage: COMMON

Author/Change controller: IESG

7.2. 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 3.1

7.3. swd:id 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:id

Extension IRI: urn:ietf:params:rolie:property:swd:id

Reference: This document, Section 4.1

Subregistry: None

7.4. swd:sname 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:sname

Extension IRI: urn:ietf:params:rolie:property:swd:sname

Reference: This document, Section 4.2

Subregistry: None

8. Security Considerations

Use of this extension implies dealing with the security implications of both ROLIE and of software descriptors in general. As with any SWD information, care should be taken to verify the trustworthiness and veracity of the descriptor information to the fullest extent possible.

Ideally, software descriptors should have been signed by the software manufacturer, or signed by whichever agent processed the source code. SWD documents from these sources are more likely to be accurate than those generated by scraping installed software.

These "authoritative" sources of SWD 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 SWD version information for low fault tolerance comparisons and searches, care should be taken that the correct version scheme is being utilized.

9. Privacy Considerations

This extension does not introduce any privacy considerations above or beyond that of the core ROLIE document. Any implementations using this extension should understand the privacy considerations of ROLIE and the Atom Publishing Protocol.

10. Normative References

[I-D.ietf-sacm-coswid]

Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identifiers", draft-ietf-sacm-coswid-05 (work in progress), March 2018.

[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.

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 SWD 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/provider/SWD/123456"/>
  <category
    scheme="urn:ietf:params:rolie:category:information-type"
    term="software-descriptor"/>
  <rolie:format ns="urn:example:COSWID"/>
  <content type="application/xml"
    src="http://www.example.org/provider/SWD/123456/data"/>
</entry>
```

Authors' Addresses

David Waltermire
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Email: david.waltermire@nist.gov

Stephen Banghart
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Email: stephen.banghart@nist.gov

SACM Working Group
Internet-Draft
Intended status: Informational
Expires: June 17, 2019

H. Birkholz
Fraunhofer SIT
J. Lu
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

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

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

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

This Internet-Draft will expire on June 17, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

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

1. Introduction

Our goal with this document is to improve our agreement on the terminology used in documents produced by the IETF Working Group for Security Automation and Continuous Monitoring. Agreeing on terminology should help reach consensus on which problems we're trying to solve, and propose solutions and decide which ones to use.

2. Terms and Definitions

This section describes terms that have been defined by other RFC's and defines new ones. The predefined terms will reference the RFC and where appropriate will be annotated with the specific context by which the term is used in SACM. Note that explanatory or informational augmentation to definitions are segregated from the definitions themselves. The definition for the term immediately follows the term on the same line, whereas expository 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 be 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 schedule 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 summary 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 quantity 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:

- o 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:

- o Added Vulnerability, Vulnerability Management, xposure, Misconfiguration, and Software flaw.

Changes from version 02 to version 03:

- o 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:

- o TODO

Changes from version 04 to version 05:

- o TODO

Changes from version 05 to version 06:

- o Updated author information.
- o Combined "Pre-defined Terms" with "New Terms and Definitions".
- o Removed "Requirements language".
- o Removed unused reference to use case draft; resulted in removal of normative references.
- o Removed introductory text from Section 1 indicating that this document is intended to be temporary.
- o Added placeholders for missing change log entries.

Changes from version 06 to version 07:

- o Added Contributors section.
- o Updated author list.
- o Changed title from "Terminology for Security Assessment" to "Secure Automation and Continuous Monitoring (SACM) Terminology".
- o Changed abbrev from "SACM-Terms" to "SACM Terminology".
- o Added appendix The Attic to stash terms for future updates.
- o Added Authentication, Authorization, Data Confidentiality, Data Integrity, Data Origin, Data Provenance, SACM Component, SACM Component Discovery, Target Endpoint Discovery.
- o Major updates to Building Block, Function, SACM Role, Target Endpoint.
- o Minor updates to Broker, Capability, Collection Task, Evaluation Task, Posture.
- o Relabeled Role to SACM Role, Endpoint Target to Target Endpoint, Endpoint Discovery to Endpoint Identification.
- o Moved Asset Targeting, Client, Endpoint Identification to The Attic.
- o Endpoint Attributes added as a TODO.
- o Changed the structure of the Change Log.

Changes from version 07 to version 08:

- o Added Assertion, Collection Result, Collector, Excluded Endpoint, Internal Collector, Network Address, Network Interface, SACM Domain, Statement, Target Endpoint Identifier, Target Endpoint Label, Timestamp.
- o 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.
- o Minor updates to Asset, Building Block, Data Origin, Data Source, Data Provenance, Endpoint, Management Plane, Posture, Posture Attribute, SACM Component, SACM Component Discovery, Target Endpoint Discovery.

- o Relabeled Function to SACM Function.

Changes from version 08 to version 09:

- o Updated author list.
- o Added Data Plane, Endpoint Characterization, Endpoint Classification, Guidance, Interaction Model, Software Component, Software Instance, Software Package, Statement, Target Endpoint Profile, SACM Task.
- o Removed Building Block.
- o Major updates to Control Plane, Endpoint Attribute, Expected Endpoint State, Information Model, Management Plane.
- o Minor updates to Attribute, Capabilities, SACM Function, SACM Component, Collection Task.
- o Moved Asset Characterization to The Attic.

Changes from version 09 to version 10:

- o Added Configuration Drift, Data in Motion, Data at Rest, Endpoint Management Capability, Hardware Component, Hardware Inventory, Hardware Type, SACM Interface, Target Endpoint Characterization Record, Target Endpoint Characterization Task, Target Endpoint Classification Task, Target Endpoint Discovery Task, Vulnerability Description Information, Vulnerability Detection Data, Vulnerability Management Capability, Vulnerability Assessment
- o Added references to i2nsf definitions in Capability, SACM Component, SACM Interface, SACM Role.
- o Added i2nsf Terminology I-D Reference.
- o Major Updates to Endpoint, SACM Task, Target Endpoint Identifier.
- o Minor Updates to Guidance, SACM Component Discovery, Target Endpoint Label, Target Endpoint Profile.
- o Relabeled SACM Task
- o Removed Target Endpoint Discovery

Changes from version 10 to version 11:

- o Added Content Element, Content Metadata, Endpoint Label, Information Element, Metadata, SACM Component Label, Workflow.
- o Major Updates to Assessment, Capability, Collector, Endpoint Management Capabilities, Guidance, Vulnerability Assessment Capabilities, Vulnerability Detection Data, Vulnerability Assessment Capabilities.
- o Minor updates to Collection Result, Control Plane, Data in Motion, Data at Rest, Data Origin, Network Interface, Statement, Target Endpoint Label.
- o Relabeled Endpoint Management Capability, Vulnerability Management Capability, Vulnerability Assessment.

Changes from version 11 to version 12:

- o Added Configuration, Endpoint Characteristic, Event, SACM Content, State, Subject.
- o Major Updates to Assertion, Data in Motion, Data Provenance, Data Source, Interaction Model.
- o Minor Updates to Attribute, Control Plane, Data Origin, Data Provenance, Expected Endpoint State, Guidance, Target Endpoint Classification Task, Vulnerability Detection Data.

Changes from version 12 to version 13:

- o Added Virtual Component.
- o Major Updates to Capability, Collection Task, Hardware Component, Hardware Type, Security Automation, Subject, Target Endpoint, Target Endpoint Profile.
- o Minor Updates to Assertion, Data Plane, Endpoint Characteristics.

Changes from version 13 to version 14:

- o Handled a plethora of issues listed in GitHub.
- o Pruned some commonly understood terms.
- o Narrowing term labels per their definitions.
- o In some cases, excised expositional text.

- o Where expository text was left intact, it has been separated from the actual definition of a term.

Changes from version 14 to version 16:

- o moved obsolete definitions into the Appendix (attic).

7. Contributors

David Waltermire
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20877
USA

Email: david.waltermire@nist.gov

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

Email: adam.w.montville@gmail.com

David Harrington
Effective Software
50 Harding Rd
Portsmouth, NH 03801
USA

Email: ietfdbh@comcast.net

Brian Ford
Lancope
3650 Brookside Parkway, Suite 500
Alpharetta, GA 30022
USA

Email: bford@lancope.com

Merike Kaeo
Double Shot Security
3518 Fremont Avenue North, Suite 363
Seattle, WA 98103
USA

Email: merike@doubleshotsecurity.com

8. References

8.1. Normative References

- [RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5792, DOI 10.17487/RFC5792, March 2010, <<https://www.rfc-editor.org/info/rfc5792>>.
- [RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <<https://www.rfc-editor.org/info/rfc6933>>.

8.2. Informative References

- [I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-06 (work in progress), July 2018.
- [I-D.ietf-netmod-entity]
Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-08 (work in progress), January 2018.
- [I-D.ietf-sacm-vuln-scenario]
Coffin, C., Cheikes, B., Schmidt, C., Haynes, D., Fitzgerald-McKay, J., and D. Waltermire, "SACM Vulnerability Assessment Scenario", draft-ietf-sacm-vuln-scenario-02 (work in progress), September 2016.
- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <<https://www.rfc-editor.org/info/rfc3444>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.
- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.
- [RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <<https://www.rfc-editor.org/info/rfc6192>>.

[X.1252] "ITU-T X.1252 (04/2010)", n.d..

Appendix A. The Attic

The following terms are stashed for now and will be updated later:

Asset: Is a system resource, as defined in [RFC4949], that may be composed of other assets.

Examples of Assets include: Endpoints, Software, Guidance, or X.509 public key certificates. An asset is not necessarily owned by an organization.

Asset Management: The IT process by which assets are provisioned, updated, maintained and deprecated.

Asset Characterization: Asset characterization is the process of defining attributes that describe properties of an identified asset.

Asset Targeting: Asset targeting is the use of asset identification and categorization information to drive human-directed, automated decision making for data collection and analysis in support of endpoint posture assessment.

Client: An architectural component receiving services from another architectural component.

Endpoint Identification (TBD per list; was "Endpoint Discovery"):
The process by which an endpoint can be identified.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Jarrett Lu
Oracle Corporation
4180 Network Circle
Santa Clara, CA 95054
USA

Email: jarrett.lu@oracle.com

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95138
USA

Email: john.sc.strassner@huawei.com

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

Email: ncamwing@cisco.com

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

Email: adam.w.montville@gmail.com

Security Automation and Continuous Monitoring (SACM)
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

Q. Lin
L. Xia
Huawei
H. Birkholz
Fraunhofer SIT
October 22, 2018

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.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 2. Requirements Language | 3 |
| 3. Terminology | 3 |
| 4. Tree Diagrams | 4 |
| 5. Data Model Structure | 4 |
| 5.1. Administration Security | 5 |
| 5.1.1. Administrative Account Security | 5 |
| 5.1.2. Administrator Access Security | 6 |
| 5.1.3. AAA | 9 |
| 5.1.4. Administrator Access Statistics | 10 |
| 5.2. System Management Security | 11 |
| 5.2.1. SNMP Management Security | 11 |
| 5.2.2. NETCONF Management Security | 13 |
| 5.3. Port Management Security | 13 |
| 5.4. Log Security | 14 |
| 5.5. File Security | 14 |
| 6. Network Infrastructure Device Security Baseline Yang Module . | 15 |
| 6.1. Module 'ietf-admin-account-security' | 15 |
| 6.2. Module 'ietf-admin-access-security' | 18 |
| 6.3. Module 'ietf-aaa-security' | 28 |
| 6.4. Module 'ietf-admin-access-statistics' | 35 |
| 6.5. Module 'ietf-snmp-security' | 38 |
| 6.6. Module 'ietf-netconf-security' | 46 |
| 6.7. Module 'ietf-port-management-security' | 50 |
| 7. Acknowledgements | 52 |
| 8. IANA Considerations | 52 |
| 9. Security Considerations | 52 |
| 10. References | 52 |
| 10.1. Normative References | 52 |
| 10.2. Informative References | 53 |
| Appendix A. | 54 |
| Authors' Addresses | 56 |

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 prefaced 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

- o System management security
- o Port management security
- o Log security
- o 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: Qiusi Lin
            linqiusi@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
    rrastructure 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 administrative 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 logging 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, then 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 configurations of AAA.";

revision 2018-10-16 {
description "Initial version.";
reference
"draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Infrastructure 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 authentication is not in this mode, it indicates that the user authorization is passed. 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 second, third, and the following authentication mode is used only when the first authentication 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 second, third, and the following authorization mode is used only when the first authorization 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 level. The second, third, and the following command line authorization mode is used 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 contains online administrator lists, ip addresses authentication failure or blocked ip addresses.";

revision 2018-10-16 {

description "Initial version.";
reference

"draft-lin-sacm-nid-mp-security-baseline-04: The Data Model of Network Infrastructure Device Management Plane Security Baseline";
}

/*

* features

*/

feature display-online-info {
description

"If the device supports reporting the details of administrative accounts that are currently online.";
}

/*

* typedef

*/

typedef ip-block-state-type {

type enumeration {

enum "authenfail" {

description

"Authentication failed 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 administrator 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 user.";
        }
        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 authenticated, and if so, the type of authentication protocol which is used: MD5, SHA.";
    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 protected 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 access. If this leaf does not exist in a configuration, it maps to a zero-length vacmAccessReadViewName.";
    }
    leaf write-view {
        type snmp:view-name;
        description
            "The name of the MIB view of the SNMP context authorizing write access. If this leaf does not exist in a configuration, it maps to a zero-length vacmAccessWriteViewName.";
    }
    leaf notify-view {
        type snmp:view-name;
        description
            "The name of the MIB view of the SNMP context authorizing notify access. If this leaf does not exist in a configuration, it maps to a zero-length vacmAccessNotifyViewName.";
    }
    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 privacy protocols.";
    }
    container remote {
        uses user-auth-priv;
        description
            "A list of remote users and their corresponding authentication and privacy 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
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2018

A. Montville
B. Munyan
CIS
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

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

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

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

This Internet-Draft will expire on September 29, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 1.1. Open Questions | 3 |
| 1.2. Requirements notation | 3 |
| 2. Terms and Definitions | 4 |
| 3. Architectural Discovery | 4 |
| 3.1. SACM Roles | 5 |
| 3.2. Exploring An XMPP-based Solution | 5 |
| 4. Components, Capabilities, Interfaces, and Workflows | 7 |
| 4.1. Components | 7 |
| 4.2. Capabilities | 8 |
| 4.3. Interfaces | 9 |
| 4.4. (Candidate) Workflows | 9 |
| 4.4.1. Vulnerability Management | 9 |
| 4.4.2. Configuration Management | 9 |
| 4.4.3. IT Asset Management | 9 |
| 5. Privacy Considerations | 10 |
| 6. Security Considerations | 10 |
| 7. IANA Considerations | 10 |
| 8. References | 10 |
| 8.1. Normative References | 10 |
| 8.2. Informative References | 10 |
| Appendix A. Mapping to RFC8248 | 12 |
| Appendix B. Example Components | 15 |
| B.1. Policy Services | 15 |
| B.2. Software Inventory | 16 |
| B.3. Datastream Collection | 17 |
| B.4. Network Configuration Collection | 17 |
| Authors' Addresses | 18 |

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.

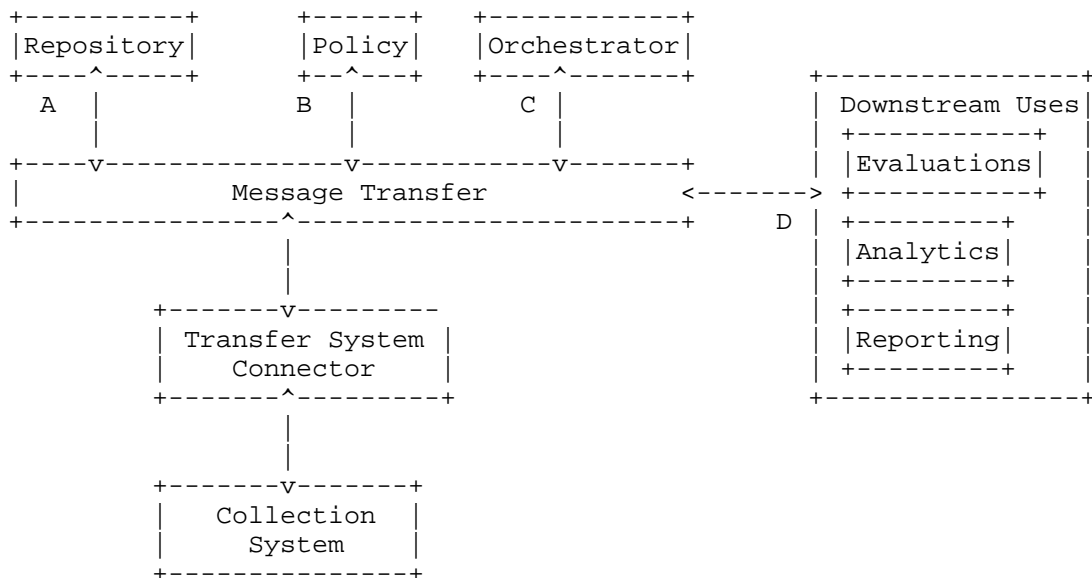


Figure 1: Notional Architecture

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 allow 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.

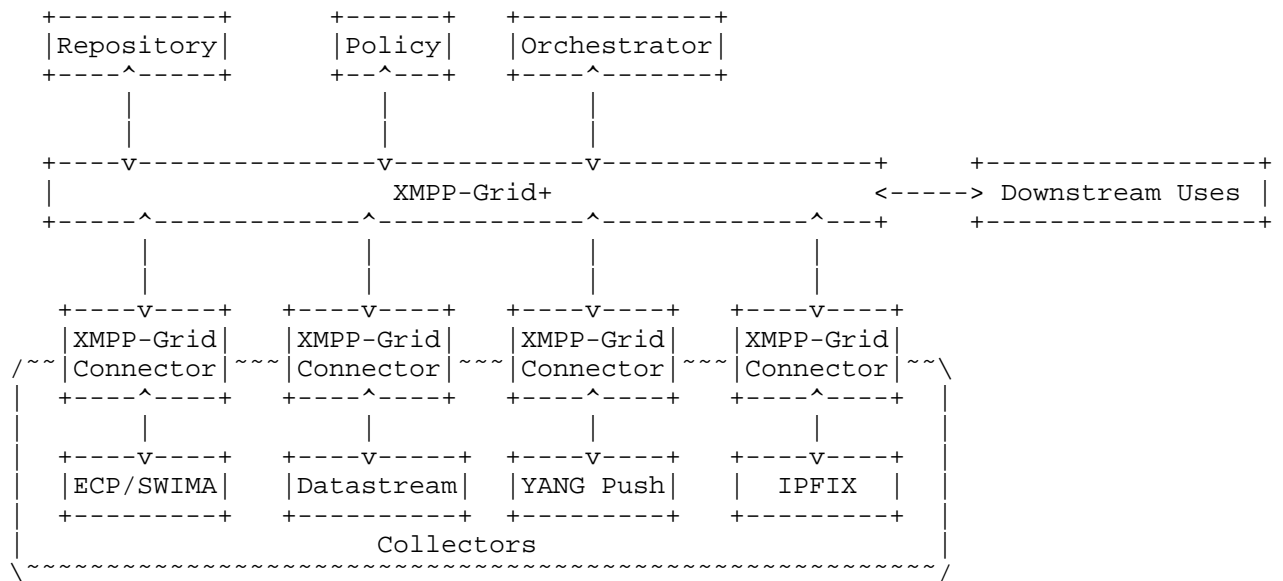


Figure 2: Detailed Architecture

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:

- o Agent-based devices
- o Non-agent based devices
- o Virtual/Cloud environments (public/private) including containers
- o Mobile devices
- o 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:

- o N/A: The requirement is not applicable to this architectural exploration
- o Architecture: This architecture (possibly assuming some components) should meet the requirement
- o XMPP: The set of XMPP Core specifications and the collection of applicable extensions, deployment, and operational considerations.
- o XMPP-Core: The requirement is satisfied by a core XMPP feature
- o XEP-nnnn: The requirement is satisfied by a numbered XMPP extension (see [XMPPEXT])
- o Operational: The requirement is an operational concern or can be addressed by an operational deployment
- o 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.

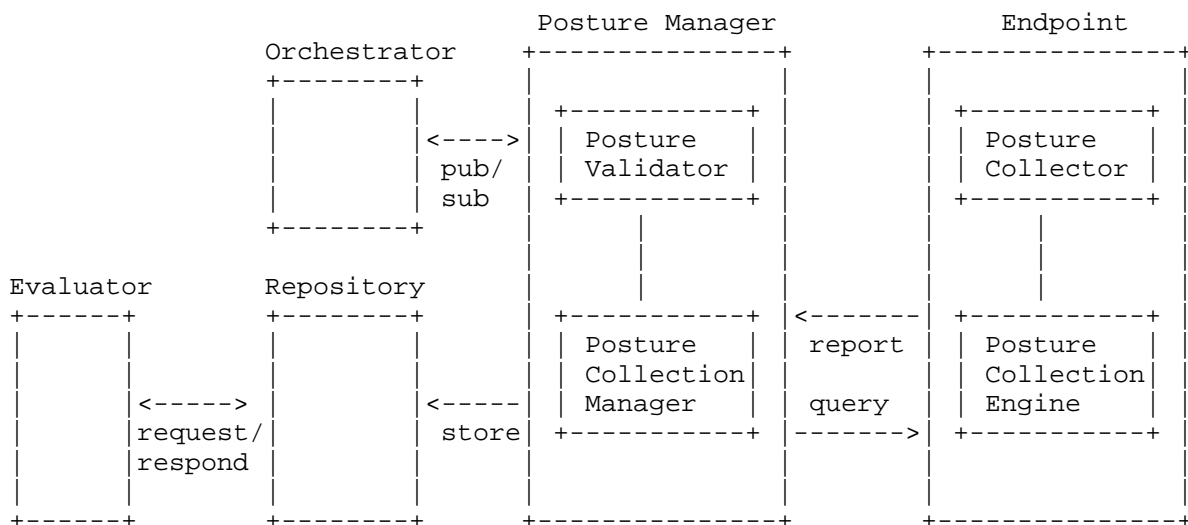


Figure 3: ECP Collection Architecture

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
Internet-Draft
Intended status: Informational
Expires: January 14, 2021

S. Banghart
NIST
B. Munyan
A. Montville
Center for Internet Security
G. Alford
Red Hat, Inc.
July 13, 2020

Definition of the ROLIE configuration checklist Extension
draft-mandm-sacm-rolie-configuration-checklist-02

Abstract

This document extends the Resource-Oriented Lightweight Information Exchange (ROLIE) core to add the information type categories and related requirements needed to support security configuration checklist use cases. Additional categories, properties, and requirements based on content type enables a higher level of interoperability between ROLIE implementations, and richer metadata for ROLIE consumers. Additionally, this document discusses requirements and usage of other ROLIE elements in order to best syndicate security configuration checklists.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. The 'configuration-checklist' information-type | 3 |
| 4. rolie:property Extensions | 4 |
| 5. Handling Existing Checklist Formats | 7 |
| 6. atom:link Extensions | 8 |
| 7. IANA Considerations | 8 |
| 7.1. configuration-checklist information-type | 8 |
| 7.2. checklist:contributor property | 9 |
| 8. Security Considerations | 9 |
| 9. Privacy Considerations | 10 |
| 10. References | 10 |
| 10.1. Normative References | 10 |
| 10.2. Informative References | 10 |
| Appendix A. Examples | 10 |
| Authors' Addresses | 11 |

1. Introduction

Default configurations for endpoints (operating systems, applications, etc.) are normally geared towards ease-of-use or ease-of-deployment, not security. As such, many enterprises operate according to guidance provided to them by a control framework ([CIS_Critical_Controls], [PCI_DSS], [NIST_800-53] etc.), which often prescribe that an enterprise define a standard, security-minded configuration for each technology they operate. Such standard configurations are often referred to as configuration checklists. This document defines an extension to the Resource-Oriented Lightweight Information Exchange (ROLIE) protocol [I-D.ietf-mile-rolie] to support the publication of configuration checklist information. Configuration checklists contain a set of configuration recommendations for a given endpoint. A configuration recommendation prescribes expected values pertaining to one or more discrete endpoint attributes.

2. Terminology

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.

The previous key words are used in this document to define the requirements for implementations of this specification. As a result, the key words in this document are not used for recommendations or requirements for the use of ROLIE.

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 section TODO of ROLIE.

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 its containing Feed as per [RFC8322].

Other terminology used in this document is defined below:

Configuration Item Generally synonymous with endpoint attribute.

Configuration Checklist A configuration checklist is an organized collection of rules about a particular kind of system or platform.

Configuration Recommendation A configuration recommendation is an expression of the desired posture of one or more configuration items. A configuration recommendation generally includes the description of the recommendation, a rationale statement, and the expected state of collected posture information.

TODO: There needs to be a "normative" reference to the SCAP 1.2/3 specifications and schema definitions

3. The 'configuration-checklist' information-type

This document registers a new information type for use in ROLIE repositories. The "configuration-checklist" information type represents a body of information describing a set of configuration recommendations. A configuration recommendation is, minimally, a single configuration item paired with a recommended value or range of

values. Depending on the source, a configuration recommendation may carry with it additional information (i.e. description, references, rationale, etc.). Provided below is a non-exhaustive list of information that may be considered as components of a configuration checklist.

- o A "Data Stream"
- o A "Benchmark"
- o A "Profile"
- o A "Value"
- o A "Rule" or "Group" of Rules
 - * Description
 - * Rationale
 - * Remediation Instructions
 - * Information, described in the dialect of a supported "check system", indicating the method(s) used to audit the checklist configuration item.
- o Applicable Platform Information
- o Information regarding a set of patches to be evaluated

4. rolie:property Extensions

A breadth of metadata may be included with a configuration checklist as identifying information. A publishing organization may wish to recognize or attribute checklist authors or contributors, or maintain a revision/version history over time. Other metadata that may be included could indicate the various categories of products to which the checklist applies, such as Operating System, Network Device, or Application Server.

This document registers several new rolie:property elements to express this metadata in a more efficient and automatable form.

- o contributor (0..n)
 - * name: urn:ietf:params:rolie:property:checklist:contributor

- * value: Indicates those individuals noted as recognized contributors to the configuration checklist and/or the recommendations contained within. The value MUST be either a plaintext name of a entity, or a link to an <author> element that describes an entity.
- o checklist version
 - * name: urn:ietf:params:rolie:property:checklist:version
 - * value: Indicates the version/revision number of the configuration checklist. Implementations MAY choose to incorporate a semantic versioning scheme illustrating "major.minor.point" releases, such as "3.1.1".
- o title
 - * name: urn:ietf:params:rolie:property:checklist:title
 - * value: Indicates the document title of the configuration checklist, such as "CIS Benchmark for Microsoft Windows Server 2019".
- o overview
 - * name: urn:ietf:params:rolie:property:checklist:overview
 - * value: This property allows for a textual overview and/or introduction to the configuration checklist including, but not limited to, overview of the technology under assessment, limitations or caveats, or assumptions to be made when evaluating the checklist.
- o product name (0..n)
 - * name: urn:ietf:params:rolie:property:checklist:product-name
 - * value: This property allows for further refinement and identification of the configuration checklist using the name of the product or products to which the checklist applies, such as Microsoft Windows Server 2019, Red Hat Enterprise Linux, IBM WebSphere Application Server, Google Chrome, etc.
- o product category (0..n)
 - * name: urn:ietf:params:rolie:property:checklist:product-category

* value: This property allows for further refinement and identification of the configuration checklist using the technology category. Examples of product category values may be (but aren't limited to):

- + Antivirus Software
- + Application Server
- + Auditing
- + Authentication
- + Automation/Productivity Application Suite
- + Client and Server Encryption
- + Configuration Management Software
- + Database Management System
- + Desktop Application
- + Desktop Client
- + DHCP Server
- + Directory Service
- + DNS Server
- + Email Server
- + Encryption Software
- + Enterprise Application
- + File Encryption
- + Firewall
- + Firmware
- + Handheld Device
- + Identity Management
- + Intrusion Detection System

- + KVM
- + Mail Server
- + Malware
- + Mobile Solution
- + Monitoring
- + Multi-Functional Peripheral
- + Network Router
- + Network Switch
- + Office Suite
- + Operating System
- + Peripheral Device
- + Security Server
- + Server
- + Virtual Machine
- + Virtualization Software
- + Web Browser
- + Web Server
- + Wireless Email
- + Wireless Network

5. Handling Existing Checklist Formats

Today, checklists are distributed in a myriad of different formats, using a variety of organization schemes. This standard attempts to be as flexible as possible in its approach, in order to be usable by as many checklist distributors as possible.

Using the NIST National Checklist Program as a foundation, checklists consist of a primary set of content and a list of supporting content. These pieces of content come in a number of machine readable and

human readable formats, and it is out of scope of this standard to describe guidance for all them. Instead, a best effort should be made to use the available properties, elements, and attributes to describe the content. Moreover, the content is often a compressed file that consists of a package of other content. Likewise, describing this nested structure is out of scope for this standard. Each organization should use a description scheme that best matches their use and business cases, and this description scheme should be documented as thoroughly as possible for all users.

When existing identifiers, titles, authors, and dates are provided in machine-readable forms inside a ROLIE Entry, automated processes can find and acquire checklist content with more ease than the current state-of-the-art methodology. Fully solving the checklist automation problem will require a more significant effort touching on all parts of the checklist ecosystem.

6. atom:link Extensions

| Name | Description |
|-----------------|---|
| ancestor | Links to a configuration checklist supersceded by that described in this entry |
| target-platform | Links to a software descriptor resource defining the software subject to this configuration checklist entry |
| supporting | Links to a supporting document for the main content. The "title" attribute SHOULD be used to provide a human readable title for this document. Where possible, the "type" attribute MAY be used to describe the type of the supporting document. If the type is a simple IANA Media Type, the media type text should be used, otherwise, a short human readable description should be used. |

7. IANA Considerations

7.1. configuration-checklist 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: configuration-checklist

index: TBD

reference: This document, Section 3

7.2. checklist:contributor 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:checklist:contributor

Extension IRI:

urn:ietf:params:rolie:property:checklist:contributor

Reference: This document, Section 4

Subregistry: None

8. Security Considerations

Use of this extension requires understanding and managing the security considerations of the core ROLIE specification. Beyond that, there must be considerations made for the common use cases and data types that would be shared with this extension in particular.

Checklist information, while typically shared publicly, can have potential security impact if compromised. In these cases, the utmost care should be taken to secure the REST endpoint. Ensure that only authenticated users are allowed request access to any part of the ROLIE repository. Authentication schemes such as OAUTH or basic HTTP Auth provides a significant barrier to compromise. When providing checklist information as a paid service, security is valuable as a means to protect valuable data from being stolen or taken for free. In these cases, the above strategies still apply, but providers may want to make the Feed visible to non-authenticated users, with meaningful error messages sent to users that have not yet paid for the service.

Typical RESTful security measures applied commonly on the web would be effective to secure this ROLIE extension. As a flexible and relatively simple RESTful service, ROLIE server implementations have great flexibility and freedom in securing their repository.

9. Privacy Considerations

This extension poses no additional privacy considerations above and beyond those stated in the core ROLIE specification.

10. References

10.1. Normative References

- [I-D.ietf-mile-rolie]
Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange", draft-ietf-mile-rolie-07 (work in progress), May 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>>.
- [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>>.
- [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>>.

10.2. Informative References

- [CIS_Critical_Controls]
"CIS Critical Security Controls", August 2016, <<https://www.cisecurity.org/critical-controls/>>.
- [NIST_800-53]
Hanson, R., "NIST 800-53", September 2007, <<http://deusty.blogspot.com/2007/09/stunt-out-of-band-channels.html>>.
- [PCI_DSS] "PCI Data Security Standard", April 2016, <https://www.pcisecuritystandards.org/document_library?category=pcidss&document=pci_dss>.

Appendix A. Examples

This section provides some brief examples of a Checklist Information Type ROLIE Entry.


```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="https://www.w3.org/2005/Atom"
  xmlns:rolie="urn:ietf:params:xml:ns:rolie-1.0" xml:lang="en-US">
  <id>c8db0a93-4dcb-426e-997f-ba43c100b863</id>
  <title>NIST National Checklist for Red Hat Virtualization Host 4.x</title>
  <published>2020-06-29T18:13:51.0Z</published>
  <updated>2020-06-29T18:13:51.0Z</updated>
  <category scheme="urn:ietf:params:rolie:category:information-type" term="checklist"/>
  <summary>SCAP content for evaluation of Red Hat Virtualization Host 4.x systems. The Red Hat content embeds multiple pre-established compliance profiles.</summary>
  <rolie:format ns="scap13namespace"/>
  <content type="application/zip" src="https://nvd.nist.gov/ncp/checklist/908/download/5615"/>
  <link rel="supporting" title="OpenControl-formatted NIST 800-53 responses for Red Hat Virtualization Host 4.x" href="https://github.com/ComplianceAsCode/redhat/tree/master/virtualization-host" type="Machine-Readable Format"/>
  <link rel="supporting" title="[DRAFT] DISA STIG for Red Hat Virtualization Host (RHVH)" href="https://galaxy.ansible.com/RedHatOfficial/rhv4_rhvh_stig" type="Ansible Playbook"/>
  <link rel="supporting" title="VPP - Protection Profile for Virtualization v. 1.0 for Red Hat Virtualization Hypervisor (RHVH)" href="https://galaxy.ansible.com/RedHatOfficial/rhv4_rhvh_vpp" type="Ansible Playbook"/>
  <rolie:property name="urn:ietf:params:rolie:property:content-id" value="908"/>
  <rolie:property name="urn:ietf:params:rolie:property:checklist:checklist-version" value="content v0.1.48"/>
  <rolie:property name="urn:ietf:params:rolie:property:content-published-date" value="2020-01-14T00:00:00+00:00"/>
  <rolie:property name="urn:ietf:params:rolie:property:content-updated-date" value="2019-06-14T00:00:00+00:00"/>
  <rolie:property name="urn:ietf:params:rolie:property:checklist:product-category" value="Virtual Machine"/>
</entry>
```

Authors' Addresses

Stephen Banghart
NIST
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Email: stephen.banghart@nist.gov

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

Email: bill.munyan.ietf@gmail.com

Adam Montville
Center for Internet Security

31 Tech Valley Drive
East Greenbush, NY 12061
USA

Email: adam.w.montville@gmail.com

Banghart, et al.

Expires January 14, 2021

[Page 11]

Gabriel Alford
Red Hat, Inc.
100 East Davie Street
Raleigh, North Carolina 27601
USA

Email: galford@redhat.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

L. Xia
G. Zheng
W. Pan
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

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

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

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

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 2 |
| 1.1. Objective | 2 |
| 1.2. Security Baseline | 3 |
| 1.3. Security Baseline Data Model Design | 4 |
| 1.4. Summary | 5 |
| 2. Terminology | 5 |
| 2.1. Key Words | 5 |
| 2.2. Definition of Terms | 6 |
| 3. Tree Diagrams | 6 |
| 4. Data Model Structure | 6 |
| 4.1. Layer 2 protection | 6 |
| 4.2. ARP | 9 |
| 4.3. URPF | 11 |
| 4.4. DHCP Snooping | 12 |
| 4.5. CPU Protection | 17 |
| 4.6. TCP/IP Attack Defense | 20 |
| 5. Network Infrastructure Device Security Baseline Yang Module . | 21 |
| 6. IANA Considerations | 46 |
| 7. Security Considerations | 46 |
| 8. Acknowledgements | 46 |
| 9. References | 46 |
| 9.1. Normative References | 46 |
| 9.2. Informative References | 46 |
| Authors' Addresses | 47 |

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:

- o 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;
- o then encapsulate the above YANG push output into a SACM Content Element envelope, which is again encapsulated in a SACM statement envelope;
- o 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-name
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 reaches the maximum number.";
      }
      leaf alarm {
        type mac-enable-status;
        default "enable";
        description
          "Whether an alarm is generated after the number of learned MAC addresses 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 reaches 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 addresses 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 reaches 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 addresses 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 addresses 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 reaches 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 addresses 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 reaches 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 addresses 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-suppress {
  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..2000000000";
        }
        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..2000000000";
        }
        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-ctrls {
```

```
description
  "Interface storm control list.";
list if-storm-ctrl {
  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-ctrl-attributes {
    description
      "Storm control rate list.";
    list if-packet-ctrl-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-ctrl-infos {
    description
        "Storm control info list.";
    list ifstorm-ctrl-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