Network Working Group A. Fuchs
Internet-Draft H. Birkholz
Intended status: Informational Fraunhofer SIT
Expires: April 26, 2019 I. McDonald
High North Inc
C. Bormann
Universitaet Bremen TZI
October 23, 2018

Time-Based Uni-Directional Attestation
draft-birkholz-i2nsf-tuda-04

Abstract

This memo documents the method and bindings used to conduct time-based uni-directional attestation between distinguishable endpoints over the network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Remote attestation (RA) describes the attempt to determine and appraise properties, such as integrity and trustworthiness, of an endpoint -- the Attestor -- over a network to another endpoint -- the Verifier -- without direct access. Typically, this kind of appraisal is based on integrity measurements of software components right before they are loaded as software instances on the Attestor. In general, attestation procedures are utilizing a hardware root of trust (RoT). The TUDA protocol family uses hash values of all started software components that are stored (extended into) a Trust-Anchor (the RoT) implemented as a Hardware Security Module (e.g. a Trusted Platform Module or similar) and are reported via a signature over those measurements.

This draft introduces the concept of including the exchange of evidence -- created via a hardware RoT containing a shielded secret that is inaccessible to the user -- in order to increase the confidence in a communication peer that is supposed to be a Trusted System [RFC4949]. In consequence, this document introduces the term forward authenticity.

Forward Authenticity (FA): A property of secure communication protocols, in which later compromise of the long-term keys of a data origin does not compromise past authentication of data from that origin. FA is achieved by timely recording of assessments of the authenticity from entities (via "audit logs" during "audit sessions") that are authorized for this purpose, in a time frame much shorter than that expected for the compromise of the long-term keys.

Forward Authenticity enables new level of guarantee and can be included in the basically every protocol, such as ssh, router advertisements, link layer neighbor discover, or even ICMP echo.

1.1. Remote Attestation

In essence, remote attestation (RA) is composed of three activities. The following definitions are derived from the definitions presented in [PRIRA] and [TCGGLOSS].

Attestation: The creation of one ore more claims about the properties of an Attestor, such that the claims can be used as evidence.

Conveyance: The transfer of evidence from the Attestor to the Verifier via an interconnect.

Verification: The appraisal of evidence by evaluating it against declarative guidance.

With TUDA, the claims that compose the evidence are signatures over trustworthy integrity measurements created by leveraging a hardware RoT. The evidence is appraised via corresponding signatures over reference integrity measurements (RIM, represented, for example via [I-D.ietf-sacm-coswid]).

Protocols that facilitate Trust-Anchor based signatures in order to provide RATS are usually bi-directional challenge/response protocols, such as the Platform Trust Service protocol [PTS] or CAVES [PRIRA], where one entity sends a challenge that is included inside the response to prove the recentness -- the freshness (see fresh in [RFC4949]) -- of the attestation information. The corresponding interaction model tightly couples the three activities of creating, transferring and appraising evidence.

The Time-Based Uni-directional Attestation family of protocols -- TUDA -- described in this document can decouple the three activities RATS are composed of. As a result, TUDA provides additional capabilities, such as:

- remote attestation for Attestors that might not always be able to reach the Internet by enabling the verification of past states,

- secure audit logs by combining the evidence created via TUDA with integrity measurement logs that represent a detailed record of corresponding past states,

- an uni-directional interaction model that can traverse "diode-like" network security functions (NSF) or can be leveraged in RESTful architectures (e.g. CoAP [RFC7252]), analogously.

1.2. Evidence Creation

TUDA is a family of protocols that bundles results from specific attestation activities. The attestation activities of TUDA are based on a hardware Root of Trust that provides the following capabilities:

- Platform Configuration Registers (PCR) that store measurements consecutively (corresponding terminology: "to extend a PCR") and represent the chain of measurements as a single measurement value ("PCR value"),

- Restricted Signing Keys (RSK) that can only be accessed, if a specific signature about measurements can be provided as authentication, and

- a dedicated source of (relative) time, e.g. a tick counter.

1.3. Evidence Appraisal

To appraise the evidence created by an Attestor, the Verifier requires corresponding Reference Integrity Measurements (RIM). Typically, a set of RIM are bundled in a RIM-Manifest (RIMM). The scope of a manifest encompasses, e.g., a platform, a device, a computing context, or a virtualised function. In order to be comparable, the hashing algorithms used by the Attestor to create the integrity measurements have to match the hashing algorithms used to create the corresponding RIM that are used by the Verifier to appraise the integrity evidence.

1.4. Activities and Actions

Depending on the platform (i.e. one or more computing contexts including a dedicated hardware RoT), a generic RA activity results in platform-specific actions that have to be conducted. In consequence, there are multiple specific operations and data models (defining the input and output of operations). Hence, specific actions are are not covered by this document. Instead, the requirements on operations and the information elements that are the input and output to these operations are illustrated using pseudo code in Appendix C and D.

1.5. Attestation and Verification

Both the attestation and the verification activity of TUDA also require a trusted Time Stamp Authority (TSA) as an additional third party next to the Attestor and the Verifier. The protocol uses a Time Stamp Authority based on [RFC3161]. The combination of the local source of time provided by the hardware RoT (located on the Attestor) and the Time Stamp Tokens provided by the TSA (to both the Attestor and the Verifier) enable the attestation and verification of an appropriate freshness of the evidence conveyed by the Attestor -- without requiring a challenge/response interaction model that uses a nonce to ensure the freshness.

Typically, the verification activity requires declarative guidance (representing desired or compliant endpoint characteristics in the form of RIM, see above) to appraise the individual integrity measurements the conveyed evidence is composed on. The acquisition or representation (data models) of declarative guidance as well as the corresponding evaluation methods are out of the scope of this document.

1.6. Information Elements and Conveyance

TUDA defines a set of information elements (IE) that are created and stored on the Attestor and are intended to be transferred to the Verifier in order to enable appraisal. Each TUDA IE:

- o is encoded in the Concise Binary Object Representation (CBOR [RFC7049]) to minimize the volume of data in motion. In this document, the composition of the CBOR data items that represent IE is described using the Concise Data Definition Language, CDDL [I-D.ietf-cbor-cddl]

- o that requires a certain freshness is only created/updated when out-dated, which reduces the overall resources required from the Attestor, including the utilization of the hardware root of trust. The IE that have to be created are determined by their age or by specific state changes on the Attestor (e.g. state changes due to a reboot-cycle)

- o is only transferred when required, which reduces the amount of data in motion necessary to conduct remote attestation significantly. Only IE that have changed since their last conveyance have to be transferred

- o that requires a certain freshness can be reused for multiple remote attestation procedures in the limits of its corresponding freshness-window, further reducing the load imposed on the Attestor and its corresponding hardware RoT.

1.7. TUDA Objectives

The Time-Based Uni-directional Attestation family of protocols is designed to:

- o increase the confidence in authentication and authorization procedures,

- o address the requirements of constrained-node networks,

- o support interaction models that do not maintain connection-state over time, such as REST architectures [REST],

- o be able to leverage existing management interfaces, such as SNMP [RFC3411]. RESTCONF [RFC8040] or CoMI [I-D.ietf-core-comi] -- and corresponding bindings,

- o support broadcast and multicast schemes (e.g. [IEEE1609]),

- o be able to cope with temporary loss of connectivity, and to

- o provide trustworthy audit logs of past endpoint states.

1.8. Hardware Dependencies

The binding of the attestation scheme used by TUDA to generate the TUDA IE is specific to the methods provided by the hardware RoT used (see above). In this document,expositional text and pseudo-code that is provided as a reference to instantiate the TUDA IE is based on TPM 1.2 and TPM 2.0 operations. The corresponding TPM commands are specified in [TPM12] and [TPM2]. The references to TPM commands and corresponding pseudo-code only serve as guidance to enable a better understanding of the attestation scheme and is intended to encourage the use of any appropriate hardware RoT or equivalent set of functions available to a CPU or Trusted Execution Environment [TEE].

1.9. 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. TUDA Core Concept

There are significant differences between conventional bi-directional attestation and TUDA regarding both the information elements conveyed between Attestor and Verifier and the time-frame, in which an attestation can be considered to be fresh (and therefore trustworthy).

In general, remote attestation using a bi-directional communication scheme includes sending a nonce-challenge within a signed attestation token. Using the TPM 1.2 as an example, a corresponding nonce-challenge would be included within the signature created by the TPM_Quote command in order to prove the freshness of the attestation response, see e.g. [PTS].

In contrast, the TUDA protocol uses the combined output of TPM_CertifyInfo and TPM_TickStampBlob. The former provides a proof about the platform's state by creating evidence that a certain key is bound to that state. The latter provides proof that the platform was in the specified state by using the bound key in a time operation. This combination enables a time-based attestation scheme. The approach is based on the concepts introduced in [SCALE] and [SFKE2008].

Each TUDA IE has an individual time-frame, in which it is considered to be fresh (and therefore trustworthy). In consequence, each TUDA IE that composes data in motion is based on different methods of creation.

The freshness properties of a challenge-response based protocol define the point-of-time of attestation between:

o the time of transmission of the nonce, and

o the reception of the corresponding response.

Given the time-based attestation scheme, the freshness property of TUDA is equivalent to that of bi-directional challenge response attestation, if the point-in-time of attestation lies between:

o the transmission of a TUDA time-synchronization token, and

o the typical round-trip time between the Verifier and the Attestor.

The accuracy of this time-frame is defined by two factors:

o the time-synchronization between the Attestor and the TSA. The time between the two tickstamps acquired via the hardware RoT define the scope of the maximum drift ("left" and "right" in respect to the timeline) to the TSA timestamp, and

o the drift of clocks included in the hardware RoT.

Since the conveyance of TUDA evidence does not rely upon a Verifier provided value (i.e. the nonce), the security guarantees of the protocol only incorporate the TSA and the hardware RoT. In consequence, TUDA evidence can even serve as proof of integrity in audit logs with precise point-in-time guarantees, in contrast to classical attestations.

Appendix A contains guidance on how to utilize a REST architecture.

Appendix B contains guidance on how to create an SNMP binding and a corresponding TUDA-MIB.

Appendix C contains a corresponding YANG module that supports both RESTCONF and CoMI.

Appendix D.2 contains a realization of TUDA using TPM 1.2 primitives.

Appendix D.3 contains a realization of TUDA using TPM 2.0 primitives.

3. Terminology

This document introduces roles, information elements and types required to conduct TUDA and uses terminology (e.g. specific certificate names) typically seen in the context of attestation or hardware security modules.

3.1. Universal Terms

Attestation Identity Key (AIK): a special purpose signature (therefore asymmetric) key that supports identity related operations. The private portion of the key pair is maintained confidential to the entity via appropriate measures (that have an impact on the scope of confidence). The public portion of the key pair may be included in AIK credentials that provide a claim about the entity.

Claim: A piece of information asserted about a subject [RFC4949]. A claim is represented as a name/value pair consisting of a Claim Name and a Claim Value [RFC7519].

In the context of SACM, a claim is also specialized as an attribute/value pair that is intended to be related to a statement [I-D.ietf-sacm-terminology].

Endpoint Attestation: the creation of evidence on the Attestor that provides proof of a set of the endpoints's integrity measurements. This is done by digitally signing a set of PCRs using an AIK shielded by the hardware RoT.

Endpoint Characteristics: the context, composition, configuration, state, and behavior of an endpoint.

Evidence: a trustworthy set of claims about an endpoint's characteristics.

Identity: a set of claims that is intended to be related to an entity.

Integrity Measurements: Metrics of endpoint characteristics (i.e. composition, configuration and state) that affect the confidence in the trustworthiness of an endpoint. Digests of integrity measurements can be stored in shielded locations (i.e. PCR of a TPM).

Reference Integrity Measurements: Signed measurements about the characteristics of an endpoint's characteristics that are provided by a vendor and are intended to be used as declarative guidance [I-D.ietf-sacm-terminology] (e.g. a signed CoSWID).

Trustworthy: the qualities of an endpoint that guarantee a specific behavior and/or endpoint characteristics defined by declarative guidance. Analogously, trustworthiness is the quality of being trustworthy with respect to declarative guidance. Trustworthiness is not an absolute property but defined with respect to an entity, corresponding declarative guidance, and has a scope of confidence.

Trustworthy Endpoint: an endpoint that guarantees trustworthy behavior and/or composition (with respect to certain declarative guidance and a scope of confidence).

Trustworthy Statement: evidence that is trustworthy conveyed by an endpoint that is not necessarily trustworthy.

3.2. Roles

Attestor: the endpoint that is the subject of the attestation to another endpoint.

Verifier: the endpoint that consumes the attestation of another endpoint to conduct a verification.

TSA: a Time Stamp Authority [RFC3161]

3.2.1. General Types

Byte: the now customary synonym for octet

Cert: an X.509 certificate represented as a byte-string

3.2.2. RoT specific terms

PCR: a Platform Configuration Register that is part of a hardware root of trust and is used to securely store and report measurements about security posture

PCR-Hash: a hash value of the security posture measurements stored in a TPM PCR (e.g. regarding running software instances) represented as a byte-string

3.3. Certificates

TSA-CA: the Certificate Authority that provides the certificate for the TSA represented as a Cert

AIK-CA: the Certificate Authority that provides the certificate for the attestation identity key of the TPM. This is the client platform credential for this protocol. It is a placeholder for a specific CA and AIK-Cert is a placeholder for the corresponding certificate, depending on what protocol was used. The specific protocols are out of scope for this document, see also [AIK-Enrollment] and [IEEE802.1AR].

4. Time-Based Uni-Directional Attestation

A Time-Based Uni-Directional Attestation (TUDA) consists of the following seven information elements. They are used to gain assurance of the Attestor's platform configuration at a certain point in time:

TSA Certificate: The certificate of the Time Stamp Authority that is used in a subsequent synchronization protocol token. This certificate is signed by the TSA-CA.

AIK Certificate: A certificate about the Attestation Identity Key (AIK) used. This may or may not also be an [IEEE802.1AR] IDevID or LDevID, depending on their setting of the corresponding identity property. ([AIK-Credential], [AIK-Enrollment]; see Appendix D.2.1.)

Synchronization Token: The reference for attestations are the relative timestanps provided by the hardware RoT. In order to put attestations into relation with a Real Time Clock (RTC), it is necessary to provide a cryptographic synchronization between these trusted relative timestamps and the regular RTC that is a hardware component of the Attestor. To do so, a synchronization protocol is run with a Time Stamp Authority (TSA).

Restriction Info: The attestation relies on the capability of the hardware RoT to operate on restricted keys. Whenever the PCR values for the machine to be attested change, a new restricted key is created that can only be operated as long as the PCRs remain in their current state.

In order to prove to the Verifier that this restricted temporary key actually has these properties and also to provide the PCR value that it is restricted, the corresponding signing capabilities of the hardware RoT are used. It creates a signed certificate using the AIK about the newly created restricted key.

Measurement Log: Similarly to regular attestations, the Verifier needs a way to reconstruct the PCRs' values in order to estimate the trustworthiness of the device. As such, a list of those elements that were extended into the PCRs is reported. Note though that for certain environments, this step may be optional if a list of valid PCR configurations (in the form of RIM available to the Verifier) exists and no measurement log is required.

Implicit Attestation: The actual attestation is then based upon a signed timestamp provided by the hardware RoT using the restricted temporary key that was certified in the steps above. The signed timestamp provides evidence that at this point in time (with respect to the relative time of the hardware RoT) a certain configuration existed (namely the PCR values associated with the restricted key). Together with the synchronization token this timestamp represented in relative time can then be related to the real-time clock.

Concise SWID tags: As an option to better assess the trustworthiness of an Attestor, a Verifier can request the reference hashes (RIM, which are often referred to as golden measurements) of all started software components to compare them with the entries in the measurement log. References hashes regarding installed (and therefore running) software can be provided by the manufacturer via SWID tags. SWID tags are provided by the Attestor using the Concise SWID representation [I-D.ietf-sacm-coswid] and bundled into a CBOR array (a RIM Manifest). Ideally, the reference hashes include a signature created by the manufacturer of the software to prove their integrity.

These information elements could be sent en bloc, but it is recommended to retrieve them separately to save bandwidth, since these elements have different update cycles. In most cases, retransmitting all seven information elements would result in unnecessary redundancy.

Furthermore, in some scenarios it might be feasible not to store all elements on the Attestor endpoint, but instead they could be retrieved from another location or be pre-deployed to the Verifier. It is also feasible to only store public keys on the Verifier and skip the whole certificate provisioning completely in order to save bandwidth and computation time for certificate verification.

4.1. TUDA Information Elements Update Cycles

An endpoint can be in various states and have various information associated with it during its life cycle. For TUDA, a subset of the states (which can include associated information) that an endpoint and its hardware root of trust can be in, is important to the attestation process. States can be:

- persistent, even after a hard reboot. This includes certificates that are associated with the endpoint itself or with services it relies on.

- volatile to a degree, because they change at the beginning of each boot cycle. This includes the capability of a hardware RoT to provide relative time which provides the basis for the synchronization token and implicit attestation--and which can reset after an endpoint is powered off.

- very volatile, because they change during an uptime cycle (the period of time an endpoint is powered on, starting with its boot). This includes the content of PCRs of a hardware RoT and thereby also the PCR-restricted signing keys used for attestation.

Depending on this "lifetime of state", data has to be transported over the wire, or not. E.g. information that does not change due to a reboot typically has to be transported only once between the Attestor and the Verifier.

There are three kinds of events that require a renewed attestation:

- The Attestor completes a boot-cycle

- A relevant PCR changes

- Too much time has passed since the last attestation statement

The third event listed above is variable per application use case and also depends on the precision of the clock included in the hardware RoT. For usage scenarios, in which the device would periodically push information to be used in an audit-log, a time-frame of approximately one update per minute should be sufficient in most cases. For those usage scenarios, where Verifiers request (pull) a fresh attestation statement, an implementation could use the hardware RoT continuously to always present the most freshly created results. To save some utilization of the hardware RoT for other purposes, however, a time-frame of once per ten seconds is recommended, which would typically leave about 80% of utilization for other applications.

```
   Attestor                                                 Verifier
      |                                                          |
    Boot                                                         |
      |                                                          |
    Create Sync-Token                                            |
      |                                                          |
    Create Restricted Key                                        |
    Certify Restricted Key                                       |
      |                                                          |
      | AIK-Cert ----------------------------------------------> |
      | Sync-Token --------------------------------------------> |
      | Certify-Info ------------------------------------------> |
      | Measurement Log ---------------------------------------> |
      | Attestation -------------------------------------------> |
      |                                           Verify Attestation
      |                                                          |
      |       <Time Passed>                                      |
      |                                                          |
      | Attestation -------------------------------------------> |
      |                                           Verify Attestation
      |                                                          |
      |       <Time Passed>                                      |
      |                                                          |
    PCR-Change                                                   |
      |                                                          |
    Create Restricted Key                                        |
    Certify Restricted Key                                       |
      |                                                          |
      | Certify-Info ------------------------------------------> |
      | Measurement Log ---------------------------------------> |
      | Attestation -------------------------------------------> |
      |                                           Verify Attestation
      |                                                          |
    Boot                                                         |
      |                                                          |
```

```
    Create Sync-Token                                            |
      |                                                          |
    Create Restricted Key                                        |
    Certify Restricted Key                                       |
      |                                                          |
      | Sync-Token --------------------------------------------> |
      | Certify-Info ------------------------------------------> |
      | Measurement Log ---------------------------------------> |
      | Attestation -------------------------------------------> |
      |                                         Verify Attestation
      |                                                          |
      |       <Time Passed>                                      |
      |                                                          |
      | Attestation -------------------------------------------> |
      |                                         Verify Attestation
      |                                                          |
```

Figure 1: Example sequence of events

5. Sync Base Protocol

The uni-directional approach of TUDA requires evidence on how the TPM time represented in ticks (relative time since boot of the TPM) relates to the standard time provided by the TSA. The Sync Base Protocol (SBP) creates evidence that binds the TPM tick time to the TSA timestamp. The binding information is used by and conveyed via the Sync Token (TUDA IE). There are three actions required to create the content of a Sync Token:

- o At a given point in time (called "left"), a signed tickstamp counter value is acquired from the hardware RoT. The hash of counter and signature is used as a nonce in the request directed at the TSA.

- o The corresponding response includes a data-structure incorporating the trusted timestamp token and its signature created by the TSA.

- o At the point-in-time the response arrives (called "right"), a signed tickstamp counter value is acquired from the hardware RoT again, using a hash of the signed TSA timestamp as a nonce.

The three time-related values -- the relative timestamps provided by the hardware RoT ("left" and "right") and the TSA timestamp -- and their corresponding signatures are aggregated in order to create a corresponding Sync Token to be used as a TUDA Information Element that can be conveyed as evidence to a Verifier.

The drift of a clock incorporated in the hardware RoT that drives the increments of the tick counter constitutes one of the triggers that can initiate a TUDA Information Element Update Cycle in respect to the freshness of the available Sync Token.

content TBD

6. IANA Considerations

This memo includes requests to IANA, including registrations for media type definitions.

TBD

7. Security Considerations

There are Security Considerations. TBD

8. Change Log

Changes from version 04 to I2NSF related document version 00: * Refactored main document to be more technology agnostic * Added first draft of procedures for TPM 2.0 * Improved content consistency and structure of all sections

Changes from version 03 to version 04:

- Refactoring of Introduction, intend, scope and audience
- Added first draft of Sync Base Prootoll section illustrated background for interaction with TSA
- Added YANG module
- Added missing changelog entry

Changes from version 02 to version 03:

- Moved base concept out of Introduction
- First refactoring of Introduction and Concept
- First restructuring of Appendices and improved references

Changes from version 01 to version 02:

- Restructuring of Introduction, highlighting conceptual prerequisites

- o Restructuring of Concept to better illustrate differences to hand-shake based attestation and deciding factors regarding freshness properties

- o Subsection structure added to Terminology

- o Clarification of descriptions of approach (these were the FIXMEs)

- o Correction of RestrictionInfo structure: Added missing signature member

Changes from version 00 to version 01:

Major update to the SNMP MIB and added a table for the Concise SWID profile Reference Hashes that provides additional information to be compared with the measurement logs.

9. Contributors

TBD

10. References

10.1. 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>.

10.2. Informative References

[AIK-Credential]
TCG Infrastructure Working Group, "TCG Credential Profile", 2007, <https://www.trustedcomputinggroup.org/wp-content/uploads/IWG-Credential_Profiles_V1_R1_14.pdf>.

[AIK-Enrollment]
TCG Infrastructure Working Group, "A CMC Profile for AIK Certificate Enrollment", 2011, <https://www.trustedcomputinggroup.org/wp-content/uploads/IWG_CMC_Profile_Cert_Enrollment_v1_r7.pdf>.

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", draft-ietf-cbor-cddl-05 (work in progress), August 2018.

[I-D.ietf-core-comi]
Veillette, M., Stok, P., Pelov, A., and A. Bierman, "CoAP Management Interface", draft-ietf-core-comi-03 (work in progress), June 2018.

[I-D.ietf-sacm-coswid]
Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identifiers", draft-ietf-sacm-coswid-06 (work in progress), July 2018.

[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-15 (work in progress), June 2018.

[IEEE1609]
IEEE Computer Society, "1609.4-2016 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) -- Multi-Channel Operation", IEEE Std 1609.4, 2016.

[IEEE802.1AR]
IEEE Computer Society, "802.1AR-2009 - IEEE Standard for Local and metropolitan area networks - Secure Device Identity", IEEE Std 802.1AR, 2009.

[PRIRA] Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., and B. Sniffen, "Principles of Remote Attestation", Springer International Journal of Information Security, Vol. 10, pp. 63-81, DOI 10.1007/s10207-011-0124-7, April 2011.

[PTS] TCG TNC Working Group, "TCG Attestation PTS Protocol Binding to TNC IF-M", 2011, <https://www.trustedcomputinggroup.org/wp-content/uploads/IFM_PTS_v1_0_r28.pdf>.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation, University of California, Irvine, 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

[RFC1213] McCloghrie, K. and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991, <https://www.rfc-editor.org/info/rfc1213>.

[RFC2790] Waldbusser, S. and P. Grillo, "Host Resources MIB", RFC 2790, DOI 10.17487/RFC2790, March 2000, <https://www.rfc-editor.org/info/rfc2790>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <https://www.rfc-editor.org/info/rfc3411>.

[RFC3418] Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, DOI 10.17487/RFC3418, December 2002, <https://www.rfc-editor.org/info/rfc3418>.

[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>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[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>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[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>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <https://www.rfc-editor.org/info/rfc7320>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[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>.

[SCALE] Fuchs, A., "Improving Scalability for Remote Attestation", Master Thesis (Diplomarbeit), Technische Universitaet Darmstadt, Germany, 2008.

[SFKE2008] Stumpf, F., Fuchs, A., Katzenbeisser, S., and C. Eckert, "Improving the scalability of platform attestation", ACM Proceedings of the 3rd ACM workshop on Scalable trusted computing - STC '08 , page 1-10, DOI 10.1145/1456455.1456457, 2008.

[STD62] "Internet Standard 62", STD 62, RFCs 3411 to 3418, December 2002.

[TCGGLOSS] TCG, "TCG Glossary", 2012, <https://www.trustedcomputinggroup.org/wp-content/uploads/TCG_Glossary_Board-Approved_12.13.2012.pdf>.

[TEE] Global Platform, "TEE System Architecture v1.1, GPD_SPE_009", 2017.

[TPM12] "Information technology -- Trusted Platform Module -- Part 1: Overview", ISO/IEC 11889-1, 2009.

[TPM2] "Trusted Platform Module Library Specification, Family 2.0, Level 00, Revision 01.16 ed., Trusted Computing Group", 2014.

Appendix A. REST Realization

Each of the seven data items is defined as a media type (Section 6). Representations of resources for each of these media types can be retrieved from URIs that are defined by the respective servers [RFC7320]. As can be derived from the URI, the actual retrieval is via one of the HTTPs ([RFC7230], [RFC7540]) or CoAP [RFC7252]. How a client obtains these URIs is dependent on the application; e.g., CoRE Web links [RFC6690] can be used to obtain the relevant URIs from the self-description of a server, or they could be prescribed by a RESTCONF data model [RFC8040].

Appendix B. SNMP Realization

SNMPv3 [STD62] [RFC3411] is widely available on computers and also constrained devices. To transport the TUDA information elements, an SNMP MIB is defined below which encodes each of the seven TUDA information elements into a table. Each row in a table contains a single read-only columnar SNMP object of datatype OCTET-STRING. The values of a set of rows in each table can be concatenated to reconstitute a CBOR-encoded TUDA information element. The Verifier can retrieve the values for each CBOR fragment by using SNMP GetNext requests to "walk" each table and can decode each of the CBOR-encoded data items based on the corresponding CDDL [I-D.ietf-cbor-cddl] definition.

Design Principles:

1. Over time, TUDA attestation values age and should no longer be used. Every table in the TUDA MIB has a primary index with the value of a separate scalar cycle counter object that disambiguates the transition from one attestation cycle to the next.

2. Over time, the measurement log information (for example) may grow large. Therefore, read-only cycle counter scalar objects in all TUDA MIB object groups facilitate more efficient access with SNMP GetNext requests.

3. Notifications are supported by an SNMP trap definition with all of the cycle counters as bindings, to alert a Verifier that a new attestation cycle has occurred (e.g., synchronization data, measurement log, etc. have been updated by adding new rows and possibly deleting old rows).

B.1. Structure of TUDA MIB

The following table summarizes the object groups, tables and their indexes, and conformance requirements for the TUDA MIB:

Group/Table	Cycle	Instance	Fragment	Required
General				x
AIKCert	x	x	x	
TSACert	x	x	x	
SyncToken	x		x	x
Restrict	x			x
Measure	x	x		
VerifyToken	x			x
SWIDTag	x	x	x	

B.1.1. Cycle Index

A tudaV1<Group>CycleIndex is the:

1. first index of a row (element instance or element fragment) in the tudaV1<Group>Table;

2. identifier of an update cycle on the table, when rows were added and/or deleted from the table (bounded by tudaV1<Group>Cycles); and

3. binding in the tudaV1TrapV2Cycles notification for directed polling.

B.1.2. Instance Index

A tudaV1<Group>InstanceIndex is the:

1. second index of a row (element instance or element fragment) in the tudaV1<Group>Table; except for

2. a row in the tudaV1SyncTokenTable (that has only one instance per cycle).

B.1.3. Fragment Index

A tudaV1<Group>FragmentIndex is the:

1. last index of a row (always an element fragment) in the tudaV1<Group>Table; and

2. accomodation for SNMP transport mapping restrictions for large string elements that require fragmentation.

B.2. Relationship to Host Resources MIB

The General group in the TUDA MIB is analogous to the System group in the Host Resources MIB [RFC2790] and provides context information for the TUDA attestation process.

The Verify Token group in the TUDA MIB is analogous to the Device group in the Host MIB and represents the verifiable state of a TPM device and its associated system.

The SWID Tag group (containing a Concise SWID reference hash profile [I-D.ietf-sacm-coswid]) in the TUDA MIB is analogous to the Software Installed and Software Running groups in the Host Resources MIB [RFC2790].

B.3. Relationship to Entity MIB

The General group in the TUDA MIB is analogous to the Entity General group in the Entity MIB v4 [RFC6933] and provides context information for the TUDA attestation process.

The SWID Tag group in the TUDA MIB is analogous to the Entity Logical group in the Entity MIB v4 [RFC6933].

B.4. Relationship to Other MIBs

The General group in the TUDA MIB is analogous to the System group in MIB-II [RFC1213] and the System group in the SNMPv2 MIB [RFC3418] and provides context information for the TUDA attestation process.

B.5. Definition of TUDA MIB

```
<CODE BEGINS>
TUDA-V1-ATTESTATION-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, Counter32,
    enterprises, NOTIFICATION-TYPE
        FROM SNMPv2-SMI                  -- RFC 2578
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF                 -- RFC 2580
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB;         -- RFC 3411

tudaV1MIB MODULE-IDENTITY
```

```
       LAST-UPDATED    "201805030000Z" --  03 May 2018
       ORGANIZATION
           "Fraunhofer SIT"
       CONTACT-INFO
           "Andreas Fuchs
           Fraunhofer Institute for Secure Information Technology
           Email: andreas.fuchs@sit.fraunhofer.de

           Henk Birkholz
           Fraunhofer Institute for Secure Information Technology
           Email: henk.birkholz@sit.fraunhofer.de

           Ira E McDonald
           High North Inc
           Email: blueroofmusic@gmail.com

           Carsten Bormann
           Universitaet Bremen TZI
           Email: cabo@tzi.org"

       DESCRIPTION
           "The MIB module for monitoring of time-based unidirectional
           attestation information from a network endpoint system,
           based on the Trusted Computing Group TPM 1.2 definition.

           Copyright (C) High North Inc (2018)."

       REVISION "201805030000Z" -- 03 May 2018
       DESCRIPTION
           "Seventh version, published as draft-birkholz-i2nsf-tuda-03."

       REVISION "201805020000Z" -- 02 May 2018
       DESCRIPTION
           "Sixth version, published as draft-birkholz-i2nsf-tuda-02."

       REVISION "201710300000Z" -- 30 October 2017
       DESCRIPTION
           "Fifth version, published as draft-birkholz-i2nsf-tuda-01."

       REVISION "201701090000Z" -- 09 January 2017
       DESCRIPTION
           "Fourth version, published as draft-birkholz-i2nsf-tuda-00."

       REVISION "201607080000Z" -- 08 July 2016
       DESCRIPTION
           "Third version, published as draft-birkholz-tuda-02."

       REVISION "201603210000Z" -- 21 March 2016
```

```
       DESCRIPTION
           "Second version, published as draft-birkholz-tuda-01."

       REVISION "201510180000Z" -- 18 October 2015
       DESCRIPTION
           "Initial version, published as draft-birkholz-tuda-00."

           ::= { enterprises fraunhofersit(21616) mibs(1) tudaV1MIB(1) }

   tudaV1MIBNotifications      OBJECT IDENTIFIER ::= { tudaV1MIB 0 }
   tudaV1MIBObjects            OBJECT IDENTIFIER ::= { tudaV1MIB 1 }
   tudaV1MIBConformance        OBJECT IDENTIFIER ::= { tudaV1MIB 2 }

   --
   --  General
   --
   tudaV1General           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 1 }

   tudaV1GeneralCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of TUDA update cycles that have occurred, i.e.,
           sum of all the individual group cycle counters.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1General 1 }

   tudaV1GeneralVersionInfo OBJECT-TYPE
       SYNTAX      SnmpAdminString (SIZE(0..255))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Version information for TUDA MIB, e.g., specific release
           version of TPM 1.2 base specification and release version
           of TPM 1.2 errata specification and manufacturer and model
           TPM module itself."
       DEFVAL      { "" }
       ::= { tudaV1General 2 }

   --
   --  AIK Cert
   --
   tudaV1AIKCert           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 2 }

   tudaV1AIKCertCycles OBJECT-TYPE
       SYNTAX      Counter32
```

```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of AIK Certificate chain update cycles that have
            occurred.

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1AIKCert 1 }

    tudaV1AIKCertTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF TudaV1AIKCertEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of fragments of AIK Certificate data."
        ::= { tudaV1AIKCert 2 }

    tudaV1AIKCertEntry OBJECT-TYPE
        SYNTAX      TudaV1AIKCertEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one fragment of AIK Certificate data."
        INDEX       { tudaV1AIKCertCycleIndex,
                      tudaV1AIKCertInstanceIndex,
                      tudaV1AIKCertFragmentIndex }
        ::= { tudaV1AIKCertTable 1 }

    TudaV1AIKCertEntry ::=
        SEQUENCE {
            tudaV1AIKCertCycleIndex         Integer32,
            tudaV1AIKCertInstanceIndex      Integer32,
            tudaV1AIKCertFragmentIndex      Integer32,
            tudaV1AIKCertData               OCTET STRING
        }

    tudaV1AIKCertCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "High-order index of this AIK Certificate fragment.
            Index of an AIK Certificate chain update cycle that has
            occurred (bounded by the value of tudaV1AIKCertCycles).

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1AIKCertEntry 1 }
```

```
   tudaV1AIKCertInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Middle index of this AIK Certificate fragment.
           Ordinal of this AIK Certificate in this chain, where the AIK
           Certificate itself has an ordinal of '1' and higher ordinals
           go *up* the certificate chain to the Root CA.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1AIKCertEntry 2 }

   tudaV1AIKCertFragmentIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this AIK Certificate fragment.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1AIKCertEntry 3 }

   tudaV1AIKCertData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "A fragment of CBOR encoded AIK Certificate data."
       DEFVAL      { "" }
       ::= { tudaV1AIKCertEntry 4 }

   --
   --  TSA Cert
   --
   tudaV1TSACert           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 3 }

   tudaV1TSACertCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of TSA Certificate chain update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1TSACert 1 }
```

```
   tudaV1TSACertTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1TSACertEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "A table of fragments of TSA Certificate data."
       ::= { tudaV1TSACert 2 }

   tudaV1TSACertEntry OBJECT-TYPE
       SYNTAX      TudaV1TSACertEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry for one fragment of TSA Certificate data."
       INDEX       { tudaV1TSACertCycleIndex,
                     tudaV1TSACertInstanceIndex,
                     tudaV1TSACertFragmentIndex }
       ::= { tudaV1TSACertTable 1 }

   TudaV1TSACertEntry ::=
       SEQUENCE {
           tudaV1TSACertCycleIndex         Integer32,
           tudaV1TSACertInstanceIndex      Integer32,
           tudaV1TSACertFragmentIndex      Integer32,
           tudaV1TSACertData               OCTET STRING
       }

   tudaV1TSACertCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "High-order index of this TSA Certificate fragment.
           Index of a TSA Certificate chain update cycle that has
           occurred (bounded by the value of tudaV1TSACertCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1TSACertEntry 1 }

   tudaV1TSACertInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Middle index of this TSA Certificate fragment.
           Ordinal of this TSA Certificate in this chain, where the TSA
           Certificate itself has an ordinal of '1' and higher ordinals
           go *up* the certificate chain to the Root CA.
```

```
            DEFVAL intentionally omitted - index object."
        ::= { tudaV1TSACertEntry 2 }

    tudaV1TSACertFragmentIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Low-order index of this TSA Certificate fragment.

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1TSACertEntry 3 }

    tudaV1TSACertData OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE(0..1024))
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "A fragment of CBOR encoded TSA Certificate data."
        DEFVAL      { "" }
        ::= { tudaV1TSACertEntry 4 }

    --
    --  Sync Token
    --
    tudaV1SyncToken         OBJECT IDENTIFIER ::= { tudaV1MIBObjects 4 }

    tudaV1SyncTokenCycles OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of Sync Token update cycles that have
            occurred.

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1SyncToken 1 }

    tudaV1SyncTokenInstances OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of Sync Token instance entries that have
            been recorded (some entries MAY have been pruned).

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1SyncToken 2 }
```

```
   tudaV1SyncTokenTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1SyncTokenEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "A table of fragments of Sync Token data."
       ::= { tudaV1SyncToken 3 }

   tudaV1SyncTokenEntry OBJECT-TYPE
       SYNTAX      TudaV1SyncTokenEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry for one fragment of Sync Token data."
       INDEX       { tudaV1SyncTokenCycleIndex,
                     tudaV1SyncTokenInstanceIndex,
                     tudaV1SyncTokenFragmentIndex }
       ::= { tudaV1SyncTokenTable 1 }

   TudaV1SyncTokenEntry ::=
       SEQUENCE {
           tudaV1SyncTokenCycleIndex       Integer32,
           tudaV1SyncTokenInstanceIndex    Integer32,
           tudaV1SyncTokenFragmentIndex    Integer32,
           tudaV1SyncTokenData             OCTET STRING
       }

   tudaV1SyncTokenCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "High-order index of this Sync Token fragment.
           Index of a Sync Token update cycle that has
           occurred (bounded by the value of tudaV1SyncTokenCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1SyncTokenEntry 1 }

   tudaV1SyncTokenInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Middle index of this Sync Token fragment.
           Ordinal of this instance of Sync Token data
           (NOT bounded by the value of tudaV1SyncTokenInstances).
```

```
           DEFVAL intentionally omitted - index object."
       ::= { tudaV1SyncTokenEntry 2 }

   tudaV1SyncTokenFragmentIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this Sync Token fragment.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1SyncTokenEntry 3 }

   tudaV1SyncTokenData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "A fragment of CBOR encoded Sync Token data."
       DEFVAL      { "" }
       ::= { tudaV1SyncTokenEntry 4 }

   --
   --  Restriction Info
   --
   tudaV1Restrict          OBJECT IDENTIFIER ::= { tudaV1MIBObjects 5 }

   tudaV1RestrictCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of Restriction Info update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1Restrict 1 }

   tudaV1RestrictTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1RestrictEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "A table of instances of Restriction Info data."
       ::= { tudaV1Restrict 2 }

   tudaV1RestrictEntry OBJECT-TYPE
       SYNTAX      TudaV1RestrictEntry
```

```
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one instance of Restriction Info data."
        INDEX       { tudaV1RestrictCycleIndex }
        ::= { tudaV1RestrictTable 1 }

    TudaV1RestrictEntry ::=
        SEQUENCE {
            tudaV1RestrictCycleIndex        Integer32,
            tudaV1RestrictData              OCTET STRING
        }

    tudaV1RestrictCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Index of this Restriction Info entry.
            Index of a Restriction Info update cycle that has
            occurred (bounded by the value of tudaV1RestrictCycles).

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1RestrictEntry 1 }


    tudaV1RestrictData OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE(0..1024))
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "An instance of CBOR encoded Restriction Info data."
        DEFVAL      { "" }
        ::= { tudaV1RestrictEntry 2 }

    --
    --  Measurement Log
    --
    tudaV1Measure           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 6 }

    tudaV1MeasureCycles OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of Measurement Log update cycles that have
            occurred.
```

```
            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1Measure 1 }

    tudaV1MeasureInstances OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of Measurement Log instance entries that have
            been recorded (some entries MAY have been pruned).

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1Measure 2 }

    tudaV1MeasureTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF TudaV1MeasureEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of instances of Measurement Log data."
        ::= { tudaV1Measure 3 }

    tudaV1MeasureEntry OBJECT-TYPE
        SYNTAX      TudaV1MeasureEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one instance of Measurement Log data."
        INDEX       { tudaV1MeasureCycleIndex,
                      tudaV1MeasureInstanceIndex }
        ::= { tudaV1MeasureTable 1 }

    TudaV1MeasureEntry ::=
        SEQUENCE {
            tudaV1MeasureCycleIndex         Integer32,
            tudaV1MeasureInstanceIndex      Integer32,
            tudaV1MeasureData               OCTET STRING
        }

    tudaV1MeasureCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "High-order index of this Measurement Log entry.
            Index of a Measurement Log update cycle that has
            occurred (bounded by the value of tudaV1MeasureCycles).
```

```
           DEFVAL intentionally omitted - index object."
       ::= { tudaV1MeasureEntry 1 }

   tudaV1MeasureInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this Measurement Log entry.
           Ordinal of this instance of Measurement Log data
           (NOT bounded by the value of tudaV1MeasureInstances).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1MeasureEntry 2 }

   tudaV1MeasureData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "A instance of CBOR encoded Measurement Log data."
       DEFVAL      { "" }
       ::= { tudaV1MeasureEntry 3 }

   --
   --  Verify Token
   --
   tudaV1VerifyToken       OBJECT IDENTIFIER ::= { tudaV1MIBObjects 7 }

   tudaV1VerifyTokenCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of Verify Token update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1VerifyToken 1 }

   tudaV1VerifyTokenTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1VerifyTokenEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "A table of instances of Verify Token data."
       ::= { tudaV1VerifyToken 2 }
```

```
   tudaV1VerifyTokenEntry OBJECT-TYPE
       SYNTAX      TudaV1VerifyTokenEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry for one instance of Verify Token data."
       INDEX       { tudaV1VerifyTokenCycleIndex }
       ::= { tudaV1VerifyTokenTable 1 }

   TudaV1VerifyTokenEntry ::=
       SEQUENCE {
           tudaV1VerifyTokenCycleIndex     Integer32,
           tudaV1VerifyTokenData           OCTET STRING
       }

   tudaV1VerifyTokenCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Index of this instance of Verify Token data.
           Index of a Verify Token update cycle that has
           occurred (bounded by the value of tudaV1VerifyTokenCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1VerifyTokenEntry 1 }

   tudaV1VerifyTokenData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "A instance of CBOR encoded Verify Token data."
       DEFVAL      { "" }
       ::= { tudaV1VerifyTokenEntry 2 }

   --
   --  SWID Tag
   --
   tudaV1SWIDTag           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 8 }

   tudaV1SWIDTagCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of SWID Tag update cycles that have occurred.
```

```
            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1SWIDTag 1 }

    tudaV1SWIDTagTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF TudaV1SWIDTagEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of fragments of SWID Tag data."
        ::= { tudaV1SWIDTag 2 }

    tudaV1SWIDTagEntry OBJECT-TYPE
        SYNTAX      TudaV1SWIDTagEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one fragment of SWID Tag data."
        INDEX       { tudaV1SWIDTagCycleIndex,
                      tudaV1SWIDTagInstanceIndex,
                      tudaV1SWIDTagFragmentIndex }
        ::= { tudaV1SWIDTagTable 1 }

    TudaV1SWIDTagEntry ::=
        SEQUENCE {
            tudaV1SWIDTagCycleIndex         Integer32,
            tudaV1SWIDTagInstanceIndex      Integer32,
            tudaV1SWIDTagFragmentIndex      Integer32,
            tudaV1SWIDTagData               OCTET STRING
        }

    tudaV1SWIDTagCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "High-order index of this SWID Tag fragment.
            Index of an SWID Tag update cycle that has
            occurred (bounded by the value of tudaV1SWIDTagCycles).

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1SWIDTagEntry 1 }

    tudaV1SWIDTagInstanceIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Middle index of this SWID Tag fragment.
```

```
            Ordinal of this SWID Tag instance in this update cycle.

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1SWIDTagEntry 2 }

    tudaV1SWIDTagFragmentIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Low-order index of this SWID Tag fragment.

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1SWIDTagEntry 3 }

    tudaV1SWIDTagData OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE(0..1024))
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "A fragment of CBOR encoded SWID Tag data."
        DEFVAL      { "" }
        ::= { tudaV1SWIDTagEntry 4 }

    --
    --  Trap Cycles
    --
    tudaV1TrapV2Cycles NOTIFICATION-TYPE
        OBJECTS {
            tudaV1GeneralCycles,
            tudaV1AIKCertCycles,
            tudaV1TSACertCycles,
            tudaV1SyncTokenCycles,
            tudaV1SyncTokenInstances,
            tudaV1RestrictCycles,
            tudaV1MeasureCycles,
            tudaV1MeasureInstances,
            tudaV1VerifyTokenCycles,
            tudaV1SWIDTagCycles
        }
        STATUS  current
        DESCRIPTION
            "This trap is sent when the value of any cycle or instance
            counter changes (i.e., one or more tables are updated).

            Note:  The value of sysUpTime in IETF MIB-II (RFC 1213) is
            always included in SNMPv2 traps, per RFC 3416."
        ::= { tudaV1MIBNotifications 1 }
```

```
   --
   --  Conformance Information
   --
   tudaV1Compliances           OBJECT IDENTIFIER
       ::= { tudaV1MIBConformance 1 }

   tudaV1ObjectGroups          OBJECT IDENTIFIER
       ::= { tudaV1MIBConformance 2 }

   tudaV1NotificationGroups    OBJECT IDENTIFIER
       ::= { tudaV1MIBConformance 3 }

   --
   --  Compliance Statements
   --
   tudaV1BasicCompliance MODULE-COMPLIANCE
       STATUS  current
       DESCRIPTION
           "An implementation that complies with this module MUST
           implement all of the objects defined in the mandatory
           group tudaV1BasicGroup."
       MODULE  -- this module
       MANDATORY-GROUPS { tudaV1BasicGroup }

       GROUP   tudaV1OptionalGroup
       DESCRIPTION
           "The optional TUDA MIB objects.
           An implementation MAY implement this group."

       GROUP   tudaV1TrapGroup
       DESCRIPTION
           "The TUDA MIB traps.
           An implementation SHOULD implement this group."
       ::= { tudaV1Compliances 1 }

   --
   --  Compliance Groups
   --
   tudaV1BasicGroup OBJECT-GROUP
       OBJECTS {
           tudaV1GeneralCycles,
           tudaV1GeneralVersionInfo,
           tudaV1SyncTokenCycles,
           tudaV1SyncTokenInstances,
           tudaV1SyncTokenData,
           tudaV1RestrictCycles,
           tudaV1RestrictData,
           tudaV1VerifyTokenCycles,
```

```
            tudaV1VerifyTokenData
        }
        STATUS  current
        DESCRIPTION
            "The basic mandatory TUDA MIB objects."
        ::= { tudaV1ObjectGroups 1 }

    tudaV1OptionalGroup OBJECT-GROUP
        OBJECTS {
            tudaV1AIKCertCycles,
            tudaV1AIKCertData,
            tudaV1TSACertCycles,
            tudaV1TSACertData,
            tudaV1MeasureCycles,
            tudaV1MeasureInstances,
            tudaV1MeasureData,
            tudaV1SWIDTagCycles,
            tudaV1SWIDTagData
        }
        STATUS  current
        DESCRIPTION
            "The optional TUDA MIB objects."
        ::= { tudaV1ObjectGroups 2 }

    tudaV1TrapGroup NOTIFICATION-GROUP
        NOTIFICATIONS { tudaV1TrapV2Cycles }
        STATUS      current
        DESCRIPTION
            "The recommended TUDA MIB traps - notifications."
        ::= { tudaV1NotificationGroups 1 }

    END
    <CODE ENDS>
```

Appendix C. YANG Realization

```
<CODE BEGINS>
module TUDA-V1-ATTESTATION-MIB {

  namespace "urn:ietf:params:xml:ns:yang:smiv2:TUDA-V1-ATTESTATION-MIB";
  prefix "tuda-v1";

  import SNMP-FRAMEWORK-MIB { prefix "snmp-framework"; }
  import yang-types         { prefix "yang"; }

  organization
   "Fraunhofer SIT";
```

```
  contact
   "Andreas Fuchs
    Fraunhofer Institute for Secure Information Technology
    Email: andreas.fuchs@sit.fraunhofer.de

    Henk Birkholz
    Fraunhofer Institute for Secure Information Technology
    Email: henk.birkholz@sit.fraunhofer.de

    Ira E McDonald
    High North Inc
    Email: blueroofmusic@gmail.com

    Carsten Bormann
    Universitaet Bremen TZI
    Email: cabo@tzi.org";

  description
   "The MIB module for monitoring of time-based unidirectional
    attestation information from a network endpoint system,
    based on the Trusted Computing Group TPM 1.2 definition.

    Copyright (C) High North Inc (2017).";

  revision "2017-10-30" {
    description
     "Fifth version, published as draft-birkholz-tuda-04.";
    reference
     "draft-birkholz-tuda-04";
  }
  revision "2017-01-09" {
    description
     "Fourth version, published as draft-birkholz-tuda-03.";
    reference
     "draft-birkholz-tuda-03";
  }
  revision "2016-07-08" {
    description
     "Third version, published as draft-birkholz-tuda-02.";
    reference
     "draft-birkholz-tuda-02";
  }
  revision "2016-03-21" {
    description
     "Second version, published as draft-birkholz-tuda-01.";
    reference
     "draft-birkholz-tuda-01";
  }
```

```
  revision "2015-10-18" {
    description
     "Initial version, published as draft-birkholz-tuda-00.";
    reference
     "draft-birkholz-tuda-00";
  }

  container tudaV1General {
  description
    "TBD";

    leaf tudaV1GeneralCycles {
      type yang:counter32;
      config false;
      description
       "Count of TUDA update cycles that have occurred, i.e.,
        sum of all the individual group cycle counters.

        DEFVAL intentionally omitted - counter object.";
    }

    leaf tudaV1GeneralVersionInfo {
      type snmp-framework:SnmpAdminString {
        length "0..255";
      }
      config false;
      description
       "Version information for TUDA MIB, e.g., specific release
        version of TPM 1.2 base specification and release version
        of TPM 1.2 errata specification and manufacturer and model
        TPM module itself.";
    }
  }

  container tudaV1AIKCert {
  description
    "TBD";

    leaf tudaV1AIKCertCycles {
      type yang:counter32;
      config false;
      description
       "Count of AIK Certificate chain update cycles that have
        occurred.

        DEFVAL intentionally omitted - counter object.";
    }
```

```
    /* XXX table comments here XXX */

    list tudaV1AIKCertEntry {

      key "tudaV1AIKCertCycleIndex tudaV1AIKCertInstanceIndex
           tudaV1AIKCertFragmentIndex";
        config false;
      description
       "An entry for one fragment of AIK Certificate data.";


      leaf tudaV1AIKCertCycleIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "High-order index of this AIK Certificate fragment.
          Index of an AIK Certificate chain update cycle that has
          occurred (bounded by the value of tudaV1AIKCertCycles).

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1AIKCertInstanceIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Middle index of this AIK Certificate fragment.
          Ordinal of this AIK Certificate in this chain, where the AIK
          Certificate itself has an ordinal of '1' and higher ordinals
          go *up* the certificate chain to the Root CA.

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1AIKCertFragmentIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Low-order index of this AIK Certificate fragment.

          DEFVAL intentionally omitted - index object.";
      }
```

```
      leaf tudaV1AIKCertData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "A fragment of CBOR encoded AIK Certificate data.";
      }
    }
  }

  container tudaV1TSACert {
  description
    "TBD";

    leaf tudaV1TSACertCycles {
      type yang:counter32;
      config false;
      description
       "Count of TSA Certificate chain update cycles that have
        occurred.

        DEFVAL intentionally omitted - counter object.";
    }


    /* XXX table comments here XXX */

    list tudaV1TSACertEntry {

      key "tudaV1TSACertCycleIndex tudaV1TSACertInstanceIndex
           tudaV1TSACertFragmentIndex";
      config false;
      description
       "An entry for one fragment of TSA Certificate data.";


      leaf tudaV1TSACertCycleIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "High-order index of this TSA Certificate fragment.
          Index of a TSA Certificate chain update cycle that has
          occurred (bounded by the value of tudaV1TSACertCycles).

          DEFVAL intentionally omitted - index object.";
```

```
      }

      leaf tudaV1TSACertInstanceIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Middle index of this TSA Certificate fragment.
          Ordinal of this TSA Certificate in this chain, where the TSA
          Certificate itself has an ordinal of '1' and higher ordinals
          go *up* the certificate chain to the Root CA.

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1TSACertFragmentIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Low-order index of this TSA Certificate fragment.

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1TSACertData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "A fragment of CBOR encoded TSA Certificate data.";
      }
    }
  }

  container tudaV1SyncToken {
  description
    "TBD";

    leaf tudaV1SyncTokenCycles {
      type yang:counter32;
      config false;
      description
       "Count of Sync Token update cycles that have
        occurred.
```

```
          DEFVAL intentionally omitted - counter object.";
      }

      leaf tudaV1SyncTokenInstances {
        type yang:counter32;
        config false;
        description
         "Count of Sync Token instance entries that have
          been recorded (some entries MAY have been pruned).

          DEFVAL intentionally omitted - counter object.";
      }

      list tudaV1SyncTokenEntry {

        key "tudaV1SyncTokenCycleIndex
             tudaV1SyncTokenInstanceIndex
             tudaV1SyncTokenFragmentIndex";
        config false;
        description
         "An entry for one fragment of Sync Token data.";


        leaf tudaV1SyncTokenCycleIndex {
          type int32 {
            range "1..2147483647";
          }
          config false;
          description
           "High-order index of this Sync Token fragment.
            Index of a Sync Token update cycle that has
            occurred (bounded by the value of tudaV1SyncTokenCycles).

            DEFVAL intentionally omitted - index object.";
        }

        leaf tudaV1SyncTokenInstanceIndex {
          type int32 {
            range "1..2147483647";
          }
          config false;
          description
           "Middle index of this Sync Token fragment.
            Ordinal of this instance of Sync Token data
            (NOT bounded by the value of tudaV1SyncTokenInstances).

            DEFVAL intentionally omitted - index object.";
        }
```

```
      leaf tudaV1SyncTokenFragmentIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Low-order index of this Sync Token fragment.

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1SyncTokenData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "A fragment of CBOR encoded Sync Token data.";
      }
    }
  }

  container tudaV1Restrict {
  description
    "TBD";

    leaf tudaV1RestrictCycles {
      type yang:counter32;
      config false;
      description
       "Count of Restriction Info update cycles that have
        occurred.

        DEFVAL intentionally omitted - counter object.";
    }


    /* XXX table comments here XXX */

    list tudaV1RestrictEntry {

      key "tudaV1RestrictCycleIndex";
      config false;
      description
       "An entry for one instance of Restriction Info data.";


      leaf tudaV1RestrictCycleIndex {
```

```
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Index of this Restriction Info entry.
          Index of a Restriction Info update cycle that has
          occurred (bounded by the value of tudaV1RestrictCycles).

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1RestrictData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "An instance of CBOR encoded Restriction Info data.";
      }
    }
  }

  container tudaV1Measure {
  description
    "TBD";

    leaf tudaV1MeasureCycles {
      type yang:counter32;
      config false;
      description
       "Count of Measurement Log update cycles that have
        occurred.

        DEFVAL intentionally omitted - counter object.";
    }

    leaf tudaV1MeasureInstances {
      type yang:counter32;
      config false;
      description
       "Count of Measurement Log instance entries that have
        been recorded (some entries MAY have been pruned).

        DEFVAL intentionally omitted - counter object.";
    }

    list tudaV1MeasureEntry {
```

```
      key "tudaV1MeasureCycleIndex tudaV1MeasureInstanceIndex";
      config false;
      description
       "An entry for one instance of Measurement Log data.";


      leaf tudaV1MeasureCycleIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "High-order index of this Measurement Log entry.
          Index of a Measurement Log update cycle that has
          occurred (bounded by the value of tudaV1MeasureCycles).

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1MeasureInstanceIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Low-order index of this Measurement Log entry.
          Ordinal of this instance of Measurement Log data
          (NOT bounded by the value of tudaV1MeasureInstances).

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1MeasureData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "A instance of CBOR encoded Measurement Log data.";
      }
    }
  }

  container tudaV1VerifyToken {
  description
    "TBD";

    leaf tudaV1VerifyTokenCycles {
```

```
      type yang:counter32;
      config false;
      description
       "Count of Verify Token update cycles that have
        occurred.

        DEFVAL intentionally omitted - counter object.";
    }


    /* XXX table comments here XXX */

    list tudaV1VerifyTokenEntry {

      key "tudaV1VerifyTokenCycleIndex";
      config false;
      description
       "An entry for one instance of Verify Token data.";


      leaf tudaV1VerifyTokenCycleIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Index of this instance of Verify Token data.
          Index of a Verify Token update cycle that has
          occurred (bounded by the value of tudaV1VerifyTokenCycles).

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1VerifyTokenData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "A instanc-V1-ATTESTATION-MIB.yang
      }
    }
  }

  container tudaV1SWIDTag {
  description
    "see CoSWID and YANG SIWD module for now"
```

```
    leaf tudaV1SWIDTagCycles {
      type yang:counter32;
      config false;
      description
       "Count of SWID Tag update cycles that have occurred.

        DEFVAL intentionally omitted - counter object.";
    }

    list tudaV1SWIDTagEntry {

      key "tudaV1SWIDTagCycleIndex tudaV1SWIDTagInstanceIndex
           tudaV1SWIDTagFragmentIndex";
      config false;
      description
       "An entry for one fragment of SWID Tag data.";


      leaf tudaV1SWIDTagCycleIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "High-order index of this SWID Tag fragment.
          Index of an SWID Tag update cycle that has
          occurred (bounded by the value of tudaV1SWIDTagCycles).

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1SWIDTagInstanceIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
        description
         "Middle index of this SWID Tag fragment.
          Ordinal of this SWID Tag instance in this update cycle.

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1SWIDTagFragmentIndex {
        type int32 {
          range "1..2147483647";
        }
        config false;
```

```
        description
         "Low-order index of this SWID Tag fragment.

          DEFVAL intentionally omitted - index object.";
      }

      leaf tudaV1SWIDTagData {
        type binary {
          length "0..1024";
        }
        config false;
        description
         "A fragment of CBOR encoded SWID Tag data.";
      }
    }
  }

  notification tudaV1TrapV2Cycles {
    description
     "This trap is sent when the value of any cycle or instance
      counter changes (i.e., one or more tables are updated).

      Note:  The value of sysUpTime in IETF MIB-II (RFC 1213) is
      always included in SNMPv2 traps, per RFC 3416.";

    container tudaV1TrapV2Cycles-tudaV1GeneralCycles {
      description
       "TPD"
      leaf tudaV1GeneralCycles {
        type yang:counter32;
        description
         "Count of TUDA update cycles that have occurred, i.e.,
          sum of all the individual group cycle counters.

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1AIKCertCycles {
      description
       "TPD"
      leaf tudaV1AIKCertCycles {
        type yang:counter32;
        description
         "Count of AIK Certificate chain update cycles that have
          occurred.

          DEFVAL intentionally omitted - counter object.";
```

```
      }
    }

    container tudaV1TrapV2Cycles-tudaV1TSACertCycles {
      description
       "TPD"
      leaf tudaV1TSACertCycles {
        type yang:counter32;
        description
         "Count of TSA Certificate chain update cycles that have
          occurred.

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1SyncTokenCycles {
      description
       "TPD"
      leaf tudaV1SyncTokenCycles {
        type yang:counter32;
        description
         "Count of Sync Token update cycles that have
          occurred.

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1SyncTokenInstances {
      description
       "TPD"
      leaf tudaV1SyncTokenInstances {
        type yang:counter32;
        description
         "Count of Sync Token instance entries that have
          been recorded (some entries MAY have been pruned).

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1RestrictCycles {
      description
       "TPD"
      leaf tudaV1RestrictCycles {
        type yang:counter32;
        description
```

```
          "Count of Restriction Info update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1MeasureCycles {
      description
       "TPD"
      leaf tudaV1MeasureCycles {
        type yang:counter32;
        description
         "Count of Measurement Log update cycles that have
          occurred.

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1MeasureInstances {
      description
       "TPD"
      leaf tudaV1MeasureInstances {
        type yang:counter32;
        description
         "Count of Measurement Log instance entries that have
          been recorded (some entries MAY have been pruned).

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1VerifyTokenCycles {
      description
       "TPD"
      leaf tudaV1VerifyTokenCycles {
        type yang:counter32;
        description
         "Count of Verify Token update cycles that have
          occurred.

          DEFVAL intentionally omitted - counter object.";
      }
    }

    container tudaV1TrapV2Cycles-tudaV1SWIDTagCycles {
      description
```

```
        "TPD"
       leaf tudaV1SWIDTagCycles {
         type yang:counter32;
         description
          "Count of SWID Tag update cycles that have occurred.

           DEFVAL intentionally omitted - counter object.";
       }
     }

   }
}
<CODE ENDS>
```

Appendix D. Realization with TPM functions

D.1. TPM Functions

The following TPM structures, resources and functions are used within this approach. They are based upon the TPM specifications [TPM12] and [TPM2].

D.1.1. Tick-Session and Tick-Stamp

On every boot, the TPM initializes a new Tick-Session. Such a tick-session consists of a nonce that is randomly created upon each boot to identify the current boot-cycle - the phase between boot-time of the device and shutdown or power-off - and prevent replaying of old tick-session values. The TPM uses its internal entropy source that guarantees virtually no collisions of the nonce values between two of such boot cycles.

It further includes an internal timer that is being initialize to Zero on each reboot. From this point on, the TPM increments this timer continuously based upon its internal secure clocking information until the device is powered down or set to sleep. By its hardware design, the TPM will detect attacks on any of those properties.

The TPM offers the function TPM_TickStampBlob, which allows the TPM to create a signature over the current tick-session and two externally provided input values. These input values are designed to serve as a nonce and as payload data to be included in a TickStampBlob: TickstampBlob := sig(TPM-key, currentTicks || nonce || externalData).

As a result, one is able to proof that at a certain point in time (relative to the tick-session) after the provisioning of a certain nonce, some certain externalData was known and provided to the TPM. If an approach however requires no input values or only one input value (such as the use in this document) the input values can be set to well-known value. The convention used within TCG specifications and within this document is to use twenty bytes of zero h'0000000000000000000000000000000000000000' as well-known value.

D.1.2. Platform Configuration Registers (PCRs)

The TPM is a secure cryptoprocessor that provides the ability to store measurements and metrics about an endpoint's configuration and state in a secure, tamper-proof environment. Each of these security relevant metrics can be stored in a volatile Platform Configuration Register (PCR) inside the TPM. These measurements can be conducted at any point in time, ranging from an initial BIOS boot-up sequence to measurements taken after hundreds of hours of uptime.

The initial measurement is triggered by the Platforms so-called pre-BIOS or ROM-code. It will conduct a measurement of the first loadable pieces of code; i.e.\ the BIOS. The BIOS will in turn measure its Option ROMs and the BootLoader, which measures the OS-Kernel, which in turn measures its applications. This describes a so-called measurement chain. This typically gets recorded in a so-called measurement log, such that the values of the PCRs can be reconstructed from the individual measurements for validation.

Via its PCRs, a TPM provides a Root of Trust that can, for example, support secure boot or remote attestation. The attestation of an endpoint's identity or security posture is based on the content of an TPM's PCRs (platform integrity measurements).

D.1.3. PCR restricted Keys

Every key inside the TPM can be restricted in such a way that it can only be used if a certain set of PCRs are in a predetermined state. For key creation the desired state for PCRs are defined via the PCRInfo field inside the keyInfo parameter. Whenever an operation using this key is performed, the TPM first checks whether the PCRs are in the correct state. Otherwise the operation is denied by the TPM.

D.1.4. CertifyInfo

The TPM offers a command to certify the properties of a key by means of a signature using another key. This includes especially the keyInfo which in turn includes the PCRInfo information used during key creation. This way, a third party can be assured about the fact that a key is only usable if the PCRs are in a certain state.

D.2. IE Generation Procedures for TPM 1.2

D.2.1. AIK and AIK Certificate

Attestations are based upon a cryptographic signature performed by the TPM using a so-called Attestation Identity Key (AIK). An AIK has the properties that it cannot be exported from a TPM and is used for attestations. Trust in the AIK is established by an X.509 Certificate emitted by a Certificate Authority. The AIK certificate is either provided directly or via a so-called PrivacyCA [AIK-Enrollment].

This element consists of the AIK certificate that includes the AIK's public key used during verification as well as the certificate chain up to the Root CA for validation of the AIK certificate itself.

```
TUDA-Cert = [AIK-Cert, TSA-Cert]; maybe split into two for SNMP
AIK-Cert = Cert
TSA-Cert = Cert
```

Figure 2: TUDA-Cert element in CDDL

The TSA-Cert is a standard certificate of the TSA.

The AIK-Cert may be provisioned in a secure environment using standard means or it may follow the PrivacyCA protocols. Figure 3 gives a rough sketch of this protocol. See [AIK-Enrollment] for more information.

The X.509 Certificate is built from the AIK public key and the corresponding PKCS #7 certificate chain, as shown in Figure 3.

Required TPM functions:

```
create_AIK_Cert(...) = {
  AIK = TPM_MakeIdentity()
  IdReq = CollateIdentityRequest(AIK,EK)
  IdRes = Call(AIK-CA, IdReq)
  AIK-Cert = TPM_ActivateIdentity(AIK, IdRes)
}

/* Alternative */

create_AIK_Cert(...) = {
  AIK = TPM_CreateWrapKey(Identity)
  AIK-Cert = Call(AIK-CA, AIK.pubkey)
}
```

Figure 3: Creating the TUDA-Cert element

D.2.2. Synchronization Token

The reference for Attestations are the Tick-Sessions of the TPM. In order to put Attestations into relation with a Real Time Clock (RTC), it is necessary to provide a cryptographic synchronization between the tick session and the RTC. To do so, a synchronization protocol is run with a Time Stamp Authority (TSA) that consists of three steps:

- The TPM creates a TickStampBlob using the AIK
- This TickstampBlob is used as nonce to the Timestamp of the TSA
- Another TickStampBlob with the AIK is created using the TSA's Timestamp a nonce

The first TickStampBlob is called "left" and the second "right" in a reference to their position on a time-axis.

These three elements, with the TSA's certificate factored out, form the synchronization token

```
TUDA-Synctoken = [
  left: TickStampBlob-Output,
  timestamp: TimeStampToken,
  right: TickStampBlob-Output,
]

TimeStampToken = bytes ; RFC 3161

TickStampBlob-Output = [
  currentTicks: TPM-CURRENT-TICKS,
  sig: bytes,
]

TPM-CURRENT-TICKS = [
  currentTicks: uint
  ? (
    tickRate: uint
    tickNonce: TPM-NONCE
  )
]
; Note that TickStampBlob-Output "right" can omit the values for
;   tickRate and tickNonce since they are the same as in "left"

TPM-NONCE = bytes .size 20
```

Figure 4: TUDA-Sync element in CDDL

Required TPM functions:

```
| dummyDigest = h'0000000000000000000000000000000000000000'
| dummyNonce = dummyDigest
|
| create_sync_token(AIKHandle, TSA) = {
|   ts_left = TPM_TickStampBlob(
|       keyHandle = AIK_Handle,       /*TPM_KEY_HANDLE*/
|       antiReplay = dummyNonce,      /*TPM_NONCE*/
|       digestToStamp = dummyDigest  /*TPM_DIGEST*/)
|
|   ts = TSA_Timestamp(TSA, nonce = hash(ts_left))
|
|   ts_right = TPM_TickStampBlob(
|       keyHandle = AIK_Handle,       /*TPM_KEY_HANDLE*/
|       antiReplay = dummyNonce,      /*TPM_NONCE*/
|       digestToStamp = hash(ts))     /*TPM_DIGEST*/
|
|   TUDA-SyncToken = [[ts_left.ticks, ts_left.sig], ts,
|                     [ts_right.ticks.currentTicks, ts_right.sig]]
|   /* Note: skip the nonce and tickRate field for ts_right.ticks */
| }
```

Figure 5: Creating the Sync-Token element

D.2.3. RestrictionInfo

The attestation relies on the capability of the TPM to operate on restricted keys. Whenever the PCR values for the machine to be attested change, a new restricted key is created that can only be operated as long as the PCRs remain in their current state.

In order to prove to the Verifier that this restricted temporary key actually has these properties and also to provide the PCR value that it is restricted, the TPM command TPM_CertifyInfo is used. It creates a signed certificate using the AIK about the newly created restricted key.

This token is formed from the list of:

o PCR list,

o the newly created restricted public key, and

o the certificate.

```
TUDA-RestrictionInfo = [Composite,
                        restrictedKey_Pub: Pubkey,
                        CertifyInfo]
```

```
PCRSelection = bytes .size (2..4) ; used as bit string

Composite = [
  bitmask: PCRSelection,
  values: [*PCR-Hash],
]

Pubkey = bytes ; may be extended to COSE pubkeys

CertifyInfo = [
  TPM-CERTIFY-INFO,
  sig: bytes,
]

TPM-CERTIFY-INFO = [
  ; we don't encode TPM-STRUCT-VER:
  ; these are 4 bytes always equal to h'01010000'
  keyUsage: uint, ; 4byte? 2byte?
  keyFlags: bytes .size 4, ; 4byte
  authDataUsage: uint, ; 1byte (enum)
  algorithmParms: TPM-KEY-PARMS,
  pubkeyDigest: Hash,
  ; we don't encode TPM-NONCE data, which is 20 bytes, all zero
  parentPCRStatus: bool,
  ; no need to encode pcrinfosize
  pcrinfo: TPM-PCR-INFO,        ; we have exactly one
]

TPM-PCR-INFO = [
    pcrSelection: PCRSelection; /* TPM_PCR_SELECTION */
    digestAtRelease: PCR-Hash;  /* TPM_COMPOSITE_HASH */
    digestAtCreation: PCR-Hash; /* TPM_COMPOSITE_HASH */
]

TPM-KEY-PARMS = [
  ; algorithmID: uint, ; <= 4 bytes -- not encoded, constant for TPM1.2
  encScheme: uint, ; <= 2 bytes
  sigScheme: uint, ; <= 2 bytes
  parms: TPM-RSA-KEY-PARMS,
]

TPM-RSA-KEY-PARMS = [
  ; "size of the RSA key in bits":
  keyLength: uint
  ; "number of prime factors used by this RSA key":
  numPrimes: uint
  ; "This SHALL be the size of the exponent":
  exponentSize: null / uint / biguint
```

```
   ; "If the key is using the default exponent then the exponentSize
   ; MUST be 0" -> we represent this case as null
 ]
```

Figure 6: TUDA-Key element in CDDL

Required TPM functions:

```
dummyDigest = h'0000000000000000000000000000000000000000'
dummyNonce = dummyDigest

create_Composite

create_restrictedKey_Pub(pcrsel) = {
  PCRInfo = {pcrSelection = pcrsel,
             digestAtRelease = hash(currentValues(pcrSelection))
             digestAtCreation = dummyDigest}
  / * PCRInfo is a TPM_PCR_INFO and thus also a TPM_KEY */

  wk = TPM_CreateWrapKey(keyInfo = PCRInfo)
  wk.keyInfo.pubKey
}

create_TPM-Certify-Info = {
  CertifyInfo = TPM_CertifyKey(
      certHandle = AIK,           /* TPM_KEY_HANDLE */
      keyHandle = wk,             /* TPM_KEY_HANDLE */
      antiReply = dummyNonce)     /* TPM_NONCE */

  CertifyInfo.strip()
  /* Remove those values that are not needed */
}
```

Figure 7: Creating the pubkey

D.2.4. Measurement Log

Similarly to regular attestations, the Verifier needs a way to reconstruct the PCRs' values in order to estimate the trustworthiness of the device. As such, a list of those elements that were extended into the PCRs is reported. Note though that for certain environments, this step may be optional if a list of valid PCR configurations exists and no measurement log is required.

```
TUDA-Measurement-Log = [*PCR-Event]
PCR-Event = [
  type: PCR-Event-Type,
  pcr: uint,
  template-hash: PCR-Hash,
  filedata-hash: tagged-hash,
  pathname: text; called filename-hint in ima (non-ng)
]

PCR-Event-Type = &(
  bios: 0
  ima: 1
  ima-ng: 2
)

; might want to make use of COSE registry here
; however, that might never define a value for sha1
tagged-hash /= [sha1: 0, bytes .size 20]
tagged-hash /= [sha256: 1, bytes .size 32]
```

D.2.5. Implicit Attestation

The actual attestation is then based upon a TickStampBlob using the restricted temporary key that was certified in the steps above. The TPM-Tickstamp is executed and thereby provides evidence that at this point in time (with respect to the TPM internal tick-session) a certain configuration existed (namely the PCR values associated with the restricted key). Together with the synchronization token this tick-related timing can then be related to the real-time clock.

This element consists only of the TPM_TickStampBlock with no nonce.

```
TUDA-Verifytoken = TickStampBlob-Output
```

Figure 8: TUDA-Verify element in CDDL

Required TPM functions:

```
| imp_att = TPM_TickStampBlob(
|     keyHandle = restrictedKey_Handle,     /*TPM_KEY_HANDLE*/
|     antiReplay = dummyNonce,              /*TPM_NONCE*/
|     digestToStamp = dummyDigest)          /*TPM_DIGEST*/
|
| VerifyToken = imp_att
```

Figure 9: Creating the Verify Token

D.2.6. Attestation Verification Approach

The seven TUDA information elements transport the essential content that is required to enable verification of the attestation statement at the Verifier. The following listings illustrate the verification algorithm to be used at the Verifier in pseudocode. The pseudocode provided covers the entire verification task. If only a subset of TUDA elements changed (see Section 4.1), only the corresponding code listings need to be re-executed.

```
TSA_pub = verifyCert(TSA-CA, Cert.TSA-Cert)
AIK_pub = verifyCert(AIK-CA, Cert.AIK-Cert)
```

Figure 10: Verification of Certificates

```
ts_left = Synctoken.left
ts_right = Synctoken.right

/* Reconstruct ts_right's omitted values; Alternatively assert == */
ts_right.currentTicks.tickRate = ts_left.currentTicks.tickRate
ts_right.currentTicks.tickNonce = ts_left.currentTicks.tickNonce

ticks_left = ts_left.currentTicks
ticks_right = ts_right.currentTicks

/* Verify Signatures */
verifySig(AIK_pub, dummyNonce || dummyDigest || ticks_left)
verifySig(TSA_pub, hash(ts_left) || timestamp.time)
verifySig(AIK_pub, dummyNonce || hash(timestamp) || ticks_right)

delta_left = timestamp.time -
    ticks_left.currentTicks * ticks_left.tickRate / 1000

delta_right = timestamp.time -
    ticks_right.currentTicks * ticks_right.tickRate / 1000
```

Figure 11: Verification of Synchronization Token

```
compositeHash = hash_init()
for value in Composite.values:
    hash_update(compositeHash, value)
compositeHash = hash_finish(compositeHash)

certInfo = reconstruct_static(TPM-CERTIFY-INFO)

assert(Composite.bitmask == ExpectedPCRBitmask)
assert(certInfo.pcrinfo.PCRSelection == Composite.bitmask)
assert(certInfo.pcrinfo.digestAtRelease == compositeHash)
assert(certInfo.pubkeyDigest == hash(restrictedKey_Pub))

verifySig(AIK_pub, dummyNonce || certInfo)
```

Figure 12: Verification of Restriction Info

```
for event in Measurement-Log:
    if event.pcr not in ExpectedPCRBitmask:
        continue
    if event.type == BIOS:
        assert_whitelist-bios(event.pcr, event.template-hash)
    if event.type == ima:
        assert(event.pcr == 10)
        assert_whitelist(event.pathname, event.filedata-hash)
        assert(event.template-hash ==
               hash(event.pathname || event.filedata-hash))
    if event.type == ima-ng:
        assert(event.pcr == 10)
        assert_whitelist-ng(event.pathname, event.filedata-hash)
        assert(event.template-hash ==
               hash(event.pathname || event.filedata-hash))

    virtPCR[event.pcr] = hash_extend(virtPCR[event.pcr],
                                     event.template-hash)

for pcr in ExpectedPCRBitmask:
    assert(virtPCR[pcr] == Composite.values[i++]
```

Figure 13: Verification of Measurement Log

```
ts = Verifytoken

/* Reconstruct ts's omitted values; Alternatively assert == */
ts.currentTicks.tickRate = ts_left.currentTicks.tickRate
ts.currentTicks.tickNonce = ts_left.currentTicks.tickNonce

verifySig(restrictedKey_pub, dummyNonce || dummyDigest || ts)

ticks = ts.currentTicks

time_left = delta_right + ticks.currentTicks * ticks.tickRate / 1000
time_right = delta_left + ticks.currentTicks * ticks.tickRate / 1000

[time_left, time_right]
```

Figure 14: Verification of Attestation Token

D.3. IE Generation Procedures for TPM 2.0

The pseudo code below includes general operations that are conducted as specific TPM commands:

- hash() : description TBD
- sig() : description TBD
- X.509-Certificate() : description TBD

These represent the output structure of that command in the form of a byte string value.

D.3.1. AIK and AIK Certificate

Attestations are based upon a cryptographic signature performed by the TPM using a so-called Attestation Identity Key (AIK). An AIK has the properties that it cannot be exported from a TPM and is used for attestations. Trust in the AIK is established by an X.509 Certificate emitted by a Certificate Authority. The AIK certificate is either provided directly or via a so-called PrivacyCA [AIK-Enrollment].

This element consists of the AIK certificate that includes the AIK's public key used during verification as well as the certificate chain up to the Root CA for validation of the AIK certificate itself.

```
TUDA-Cert = [AIK-Cert, TSA-Cert]; maybe split into two for SNMP
AIK-Certificate = X.509-Certificate(AIK-Key,Restricted-Flag)
TSA-Certificate = X.509-Certificate(TSA-Key, TSA-Flag)
```

Figure 15: TUDA-Cert element for TPM 2.0

D.3.2. Synchronization Token

The synchronization token uses a different TPM command, TPM2 GetTime() instead of TPM TickStampBlob(). The TPM2 GetTime() command contains the clock and time information of the TPM. The clock information is the equivalent of TUDA v1's tickSession information.

```
TUDA-SyncToken = [
  left_GetTime = sig(AIK-Key,
                     TimeInfo = [
                       time,
                       resetCount,
                       restartCount
                     ]
                    ),
  middle_TimeStamp = sig(TSA-Key,
                         hash(left_TickStampBlob),
                         UTC-localtime
                        ),
  right_TickStampBlob = sig(AIK-Key,
                            hash(middle_TimeStamp),
                            TimeInfo = [
                              time,
                              resetCount,
                              restartCount
                            ]
                           )
]
```

Figure 16: TUDA-Sync element for TPM 2.0

D.3.3. Measurement Log

The creation procedure is identical to Appendix D.2.4.

```
Measurement-Log = [
  * [ EventName,
      PCR-Num,
      Event-Hash ]
]
```

Figure 17: TUDA-Log element for TPM 2.0

D.3.4. Explicit time-based Attestation

The TUDA attestation token consists of the result of TPM2_Quote() or a set of TPM2_PCR_READ followed by a TPM2_GetSessionAuditDigest. It proves that -- at a certain point-in-time with respect to the TPM's internal clock -- a certain configuration of PCRs was present, as denoted in the keys restriction information.

```
TUDA-AttestationToken = TUDA-AttestationToken_quote / TUDA-AttestationToken_audi
t

TUDA-AttestationToken_quote = sig(AIK-Key,
                                  TimeInfo = [
                                    time,
                                    resetCount,
                                    restartCount
                                  ],
                                  PCR-Selection = [ * PCR],
                                  PCR-Digest := PCRDigest
                                 )

TUDA-AttestationToken_audit = sig(AIK-key,
                                  TimeInfo = [
                                    time,
                                    resetCount,
                                    restartCount
                                  ],
                                  Session-Digest := PCRDigest
                                 )
```

Figure 18: TUDA-Attest element for TPM 2.0

D.3.5. Sync Proof

In order to proof to the Verifier that the TPM's clock was not 'fast-forwarded' the result of a TPM2_GetTime() is sent after the TUDA-AttestationToken.

```
TUDA-SyncProof = sig(AIK-Key,
                     TimeInfo = [
                       time,
                       resetCount,
                       restartCount
                     ]
                    ),
```

Figure 19: TUDA-Proof element for TPM 2.0

Acknowledgements

Authors' Addresses

Andreas Fuchs
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

Email: andreas.fuchs@sit.fraunhofer.de

Henk Birkholz
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

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

Email: blueroofmusic@gmail.com

Carsten Bormann
Universitaet Bremen TZI
Bibliothekstr. 1
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Network Working Group S. Hares
Internet-Draft Huawei
Intended status: Standards Track J. Jeong
Expires: September 6, 2018 J. Kim
Sungkyunkwan University
R. Moskowitz
HTT Consulting
Q. Lin
Huawei
March 5, 2018

I2NSF Capability YANG Data Model
draft-hares-i2nsf-capability-data-model-06

Abstract

This document defines a YANG data model for capabilities that enables an I2NSF user to control various network security functions in network security devices.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As the industry becomes more sophisticated and network devices (i.e., IoT, Intelligent Vehicle, and VoIP/VoLTE Phone), service providers has a lot of problems [RFC8192]. To resolve this problem, [i2nsf-nsf-cap-im] standardize capabilities of network security functions.

This document provides a YANG data model that defines the capabilities to express capabilities of security devices. The security devices can register own capabilities to Network Operator Mgmt System with this YANG data model through registration interface. After the capabilities of the devices are registered, this YANG data model can be used by the IN2SF user or Service Function Forwarder (SFF) [i2nsf-sfc] to acquire appropriate NSFs that can be controlled by the Network Operator Mgmt System. This document defines a YANG [RFC6020] data model based on the [i2nsf-nsf-cap-im]. Terms used in document are defined in [i2nsf-terminology].

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules.

The "ietf-i2nsf-capability" YANG module defined in this document provides the following features:

- o Configuration of identification for generic network security function policy
- o Configuration of event capabilities for generic network security function policy
- o Configuration of condition capabilities for generic network security function policy
- o Configuration of action capabilities for generic network security function policy
- o Configuration of strategy capabilities for generic network security function policy
- o Configuration of default action capabilities for generic network security function policy
- o RPC for acquiring appropriate network security function according to type of NSF and/or target devices.

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 terminology described in [i2nsf-nsf-cap-im] [i2rs-rib-data-model][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol.

- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] 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. Overview

This section explains overview how the YANG data model can be used by I2NSF User, Developer’s Mgmt System, and SFF. Figure 1 shows capabilities of NSFs in I2NSF Framework. As shown in this figure, Developer’s Mgmt System can register NSFs with capabilities that the device can support. To register NSFs in this way, the Developer's Mgmt System utilizes this standardized capabilities YANG data model through registration interface. Through this registration of capabilities, the a lot of problems [RFC8192] can be resolved. The following shows use cases.

Note [i2nsf-nsf-yang] is used to configure rules of NSFs in I2NSF Framework.

```
       +-------------------------------------------------------+
       |  I2NSF User (e.g., Overlay Network Mgmt, Enterprise  |
       |  Network Mgmt, another network domain's mgmt, etc.)  |
       +--------------------+----------------------------------+
                            |
  Consumer-Facing Interface |
                            |
                            |                 I2NSF
          +-----------------+------------+ Registration  +-------------+
          | Network Operator Mgmt System |  Interface    | Developer's |
          | (i.e., Security Controller)  | < --------- > | Mgmt System |
          +-----------------+------------+               +-------------+
                            |                          New NSF
                            |                          E = {}
  NSF-Facing Interface      |                          C = {IPv4, IPv6}
                            |                          A = {Allow, Deny}
                            |
       +---------------+----+------------+-----------------+
       |               |                 |                 |
   +---+---+       +---+---+         +---+---+         +---+---+
   | NSF-1 |  ...  | NSF-m |         | NSF-1 |   ...   | NSF-n |  ...
   +-------+       +-------+         +-------+         +-------+
     NSF-1           NSF-m             NSF-1             NSF-n
 E = {}            E = {user}        E = {dev}         E = {time}
 C = {IPv4}        C = {IPv6}        C = {IPv4, IPv6}  C = {IPv4}
 A = {Allow, Deny} A = {Allow, Deny} A = {Allow, Deny} A = {Allow, Deny}

   Developer Mgmt System A            Developer Mgmt System B
```

Figure 1: Capabilities of NSFs in I2NSF Framework

o If I2NSF User wants to apply rules about blocking malicious users, it is a tremendous burden to apply all of these rules to NSFs one by one. This problem can be resolved by standardizing the capabilities of NSFs. If I2NSF User wants to block malicious users with IPv6, I2NSF User sends the rules about blocking the users to Network Operator Mgmt System. When the Network Operator Mgmt System receives the rules, it sends that rules to appropriate NSFs (i.e., NSF-m in Developer Mgmt System A and NSF-1 in

Developer Mgmt System B) which can support the capabilities (i.e., IPv6). Therefore, I2NSF User need not consider NSFs where to apply the rules.

- o If NSFs find the malicious packets, it is a tremendous burden for I2NSF User to apply the rule about blocking the malicious packets to NSFs one by one. This problem can be resolved by standardizing the capabilities of NSFs. If NSFs find the malicious packets with IPv4, they can ask the Network Operator Mgmt System to alter specific rules and/or configurations. When the Network Operator Mgmt System receives the rules for malicious packets, it inspects whether the rules are reasonable and sends the rules to appropriate NSFs (i.e., NSF-1 in Developer Mgmt System A and NSF-1 and NSF-n in Developer Mgmt System B) which can support the capabilities (i.e., IPv4). Therefore, the new rules can be applied to appropriate NSFs without control of I2NSF USer.

- o If NSFs of Service Function Chaining (SFC) [i2nsf-sfc] fail, it is a tremendous burden for I2NSF User to reconfigure the policy of SFC immediately. This problem can be resolved by periodically acquiring information of appropriate NSFs of SFC. If SFF needs information of Web Application Firewall for SFC, it can ask the Network Operator Mgmt System to acquire the location information of appropriate Web Application Firewall. When the Network Operator Mgmt System receives requested information from SFF, it sends location information of Web Application Firewall to the SFF. Therefore, the policy about the NSFs of SFC can be periodically updated without control of I2NSF USer.

5. The Structure and Objective of NSF Capabilities

5.1. Generic Network Security Function Identification

This shows a identification for generic network security functions. These objects are defined as location information and target device information.

5.2. Event Capabilities

This shows a event capabilities for generic network security functions policy. This is used to specify capabilities about any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of I2NSF Policy Rules, it is used to determine whether the Condition clause of the I2NSF Policy Rule can be evaluated or not. These object of event capabilities is defined as user security event capabilities, device security event capabilities, system security event capabilities, and time security event capabilities.

These object of event capabilities can be extended according to specific vendor event features.

5.3. Condition Capabilities

This shows a condition capabilities for generic network security functions policy. This is used to specify capabilities about a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to determine whether or not the set of Actions in that (imperative) I2NSF Policy Rule can be executed or not. These object of condition capabilities is defined as packet security condition capabilities, packet payload security condition capabilities, target security condition capabilities, user security condition capabilities, context condition capabilities, and generic context condition capabilities. These object of condition capabilities can be extended according to specific vendor condition features.

5.4. Action Capabilities

This shows a action capabilities for generic network security functions policy. This is used to specify capabilities to control and monitor aspects of flow-based NSFs when the event and condition clauses are satisfied. NSFs provide security functions by executing various Actions. These object of action capabilities is defined as ingress action capabilities, egress action capabilities, and apply profile action capabilities. These object of action capabilities can be extended according to specific vendor action features.

5.5. Resolution Strategy Capabilities

This shows a resolution strategy capabilities for generic network security functions policy. This can be used to specify capabilities how to resolve conflicts that occur between the actions of the same or different policy rules that are matched and contained in this particular NSF. These objects are defined as first-matching-rule capability and last-matching-rule capability. These objects can be extended according to specific vendor resolution strategy features.

5.6. Default Action Capabilities

This shows a default action policy for generic network security functions. This can be used to specify capabilities about a predefined action when no other alternative action was matched by the currently executing I2NSF Policy Rule.

5.7. RPC for Acquiring Appropriate Network Security Function

This shows a RPC for acquiring an appropriate network security function according to type of NSF and/or target devices. If the SFF [i2nsf-sfc]does not have the location information of network security functions that it should send in own cache table, this can be used to acquire the information. These objects are defined as input data (i.e., NSF type and target devices) and output data (i.e., location information of NSF).

6. Data Model Structure

This section shows an overview of a structure tree of capabilities for generic network security functions, as defined in the [i2nsf-nsf-cap-im].

6.1. Network Security Function Identification

The data model for network security function identification has the following structure:

```
module: ietf-i2nsf-capability
    +--rw nsf* [nsf-name]
       +--rw nsf-name                         string
       +--rw nsf-type?                        nsf-type
       +--rw nsf-address
       |  +--rw (nsf-address-type)?
       |     +--:(ipv4-address)
       |     |  +--rw ipv4-address    inet:ipv4-address
       |     +--:(ipv6-address)
       |        +--rw ipv6-address    inet:ipv6-address
       +--rw target-device
       |  +--rw pc?                 boolean
       |  +--rw mobile-phone?       boolean
       |  +--rw voip-volte-phone?   boolean
       |  +--rw tablet?             boolean
       |  +--rw iot?                boolean
       |  +--rw vehicle?            boolean
       +--rw generic-nsf-capabilities
       |  +--rw net-sec-capabilities
       |     uses net-sec-caps
       +--rw complete-nsf-capabilities
          +--rw con-sec-control-capabilities
          |  uses i2nsf-con-sec-control-caps
          +--rw attack-mitigation-capabilities
             uses i2nsf-attack-mitigation-control-caps
```

Figure 2: Data Model Structure for NSF-Identification

This draft also utilizes the concepts originated in Basile, Lioy, Pitscheider, and Zhao[2015] concerning conflict resolution, use of external data, and target device. The authors are grateful to Cataldo for pointing out this excellent work.

The nsf-type object can be used for configuration about type of a NSF. The types of NSF consists of Network Firewall, Web Application Firewall, Anti-Virus, IDS, IPS, and DDoS Mitigator. The nsf-address object can be used for configuration about location of a NSF. The target-device object can be used for configuration about target devices. We will add additional type of a NSF for more generic network security functions.

6.2. Capabilities of Generic Network Security Function

The data model for Generic NSF capabilities has the following structure:

```
       +--rw generic-nsf-capabilities
          +--rw net-sec-capabilities
             uses i2nsf-net-sec-caps
```

Figure 3: Data Model Structure for Capabilities of Network Security Function

6.2.1. Event Capabilities

The data model for event capabilities has the following structure:

```
  +--rw i2nsf-net-sec-caps
          +--rw net-sec-capabilities* [nsc-capabilities-name]
             +--rw nsc-capabilities-name    string
             +--rw rule-description?        boolean
             +--rw rule-rev?                boolean
             +--rw rule-priority?           boolean
             +--rw time-zone
             |  +--rw absolute-time-zone
             |  |  +--rw time?   boolean
             |  |  +--rw date?   boolean
             |  +--rw periodic-time-zone
             |     +--rw day?    boolean
             |     +--rw month?  boolean
             +--rw event
             |  +--rw (event-type)?
             |     +--:(usr-event)
             |     |  +--rw usr-manual?                     string
             |     |  +--rw usr-sec-event-content?          boolean
```

```
                |     |  +--rw usr-sec-event-format
                |     |  |  +--rw unknown?   boolean
                |     |  |  +--rw guid?      boolean
                |     |  |  +--rw uuid?      boolean
                |     |  |  +--rw uri?       boolean
                |     |  |  +--rw fqdn?      boolean
                |     |  |  +--rw fqpn?      boolean
                |     |  +--rw usr-sec-event-type
                |     |     +--rw unknown?                 boolean
                |     |     +--rw user-created?            boolean
                |     |     +--rw user-grp-created?        boolean
                |     |     +--rw user-deleted?            boolean
                |     |     +--rw user-grp-deleted?        boolean
                |     |     +--rw user-logon?              boolean
                |     |     +--rw user-logoff?             boolean
                |     |     +--rw user-access-request?     boolean
                |     |     +--rw user-access-granted?     boolean
                |     |     +--rw user-access-violation?   boolean
                |     +--:(dev-event)
                |     |  +--rw dev-manual?                          string
                |     |  +--rw dev-sec-event-content                boolean
                |     |  +--rw dev-sec-event-format
                |     |  |  +--rw unknown?   boolean
                |     |  |  +--rw guid?      boolean
                |     |  |  +--rw uuid?      boolean
                |     |  |  +--rw uri?       boolean
                |     |  |  +--rw fqdn?      boolean
                |     |  |  +--rw fqpn?      boolean
                |     |  +--rw dev-sec-event-type
                |     |  |  +--rw unknown?                      boolean
                |     |  |  +--rw comm-alarm?                   boolean
                |     |  |  +--rw quality-of-service-alarm?     boolean
                |     |  |  +--rw process-err-alarm?            boolean
                |     |  |  +--rw equipment-err-alarm?          boolean
                |     |  |  +--rw environmental-err-alarm?      boolean
                |     |  +--rw dev-sec-event-type-severity
                |     |     +--rw unknown?         boolean
                |     |     +--rw cleared?         boolean
                |     |     +--rw indeterminate?   boolean
                |     |     +--rw critical?        boolean
                |     |     +--rw major?           boolean
                |     |     +--rw minor?           boolean
                |     |     +--rw warning?         boolean
                |     +--:(sys-event)
                |     |  +--rw sys-manual?                          string
                |     |  +--rw sys-sec-event-content?               boolean
                |     |  +--rw sys-sec-event-format
                |     |  |  +--rw unknown?   boolean
```

```
                |     |  |  +--rw guid?      boolean
                |     |  |  +--rw uuid?      boolean
                |     |  |  +--rw uri?       boolean
                |     |  |  +--rw fqdn?      boolean
                |     |  |  +--rw fqpn?      boolean
                |     |  +--rw sys-sec-event-type
                |     |     +--rw unknown?                 boolean
                |     |     +--rw audit-log-written-to?   boolean
                |     |     +--rw audit-log-cleared?      boolean
                |     |     +--rw policy-created?         boolean
                |     |     +--rw policy-edited?          boolean
                |     |     +--rw policy-deleted?         boolean
                |     |     +--rw policy-executed?        boolean
                |     +--:(time-event)
                |        +--rw time-manual?                   string
                |        +--rw time-sec-event-begin?          boolean
                |        +--rw time-sec-event-end?            boolean
                |        +--rw time-sec-event-time-zone?      boolean
                +--rw condition
                |  ...
               +--rw action
                |  ...
                +--rw resolution-strategy
                |  ...
                +--rw default-action
                     ...
```

Figure 4: Data Model Structure for Event Capabilities of Network Security Function

These objects are defined as capabilities of user security event, device security event, system security event, and time security event. These objects can be extended according to specific vendor event features. We will add additional event objects for more generic network security functions.

6.2.2. Condition Capabilities

The data model for condition capabilities has the following structure:

```
 +--rw i2nsf-net-sec-caps
       +--rw net-sec-capabilities* [nsc-capabilities-name]
          +--rw nsc-capabilities-name    string
          +--rw rule-description?        boolean
          +--rw rule-rev?                boolean
          +--rw time-zone
```

```
          |  +--rw absolute-time-zone
          |  |  +--rw time?   boolean
          |  |  +--rw date?   boolean
          |  +--rw periodic-time-zone
          |     +--rw day?    boolean
          |     +--rw month?  boolean
          +--rw event
          |  ...
          +--rw condition
          |  +--rw (condition-type)?
          |     +--:(packet-security-condition)
          |     |  +--rw packet-manual?                    string
          |     |  +--rw packet-security-mac-condition
          |     |  |  +--rw pkt-sec-cond-mac-dest?         boolean
          |     |  |  +--rw pkt-sec-cond-mac-src?          boolean
          |     |  |  +--rw pkt-sec-cond-mac-8021q?        boolean
          |     |  |  +--rw pkt-sec-cond-mac-ether-type?   boolean
          |     |  |  +--rw pkt-sec-cond-mac-tci?          string
          |     |  +--rw packet-security-ipv4-condition
          |     |  |  +--rw pkt-sec-cond-ipv4-header-length?     boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-tos?               boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-total-length?      boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-id?                boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-fragment?          boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-fragment-offset?   boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-ttl?               boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-protocol?          boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-src?               boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-dest?              boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-ipopts?            boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-sameip?            boolean
          |     |  |  +--rw pkt-sec-cond-ipv4-geoip?             boolean
          |     |  +--rw packet-security-ipv6-condition
          |     |  |  +--rw pkt-sec-cond-ipv6-dscp?             boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-ecn?              boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-traffic-class?    boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-flow-label?       boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-payload-length?   boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-next-header?      boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-hop-limit?        boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-src?              boolean
          |     |  |  +--rw pkt-sec-cond-ipv6-dest?             boolean
          |     |  +--rw packet-security-tcp-condition
          |     |  |  +--rw pkt-sec-cond-tcp-src-port?         boolean
          |     |  |  +--rw pkt-sec-cond-tcp-dest-port?         boolean
          |     |  |  +--rw pkt-sec-cond-tcp-seq-num?      boolean
          |     |  |  +--rw pkt-sec-cond-tcp-ack-num?      boolean
          |     |  |  +--rw pkt-sec-cond-tcp-window-size?   boolean
```

```
          |     |  |  +--rw pkt-sec-cond-tcp-flags?         boolean
          |     |  +--rw packet-security-udp-condition
          |     |  |  +--rw pkt-sec-cond-udp-src-port?         boolean
          |     |  |  +--rw pkt-sec-cond-udp-dest-port?        boolean
          |     |  |  +--rw pkt-sec-cond-udp-length?   boolean
          |     |  +--rw packet-security-icmp-condition
          |     |     +--rw pkt-sec-cond-icmp-type?      boolean
          |     |     +--rw pkt-sec-cond-icmp-code?      boolean
          |     |     +--rw pkt-sec-cond-icmp-seg-num?   boolean
          |     +--:(packet-payload-condition)
          |     |  +--rw packet-payload-manual?            string
          |     |  +--rw pkt-payload-content?              boolean
          |     +--:(target-condition)
          |     |  +--rw target-manual?                    string
          |     |  +--rw device-sec-context-cond?          boolean
          |     +--:(users-condition)
          |     |  +--rw users-manual?                     string
          |     |  +--rw user
          |     |  |  +--rw (user-name)?
          |     |  |     +--:(tenant)
          |     |  |     |  +--rw tenant?   boolean
          |     |  |     +--:(vn-id)
          |     |  |        +--rw vn-id?    boolean
          |     |  +--rw group
          |     |     +--rw (group-name)?
          |     |        +--:(tenant)
          |     |        |  +--rw tenant?   boolean
          |     |        +--:(vn-id)
          |     |           +--rw vn-id?    boolean
          |     +--:(context-condition)
          |     |  +--rw context-manual?                   string
          |     +--:(gen-context-condition)
          |        +--rw gen-context-manual?               string
          |        +--rw geographic-location
          |           +--rw src-geographic-location?    boolean
          |           +--rw dest-geographic-location?   boolean
          +--rw action
          |  ...
          +--rw resolution-strategy
          |  ...
          +--rw default-action
             ...
```

Figure 5: Data Model Structure for Condition Capabilities of Network Security Function

These objects are defined as capabilities of packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. These objects can be extended according to specific vendor condition features. We will add additional condition objects for more generic network security functions.

6.2.3. Action Capabilities

The data model for action capabilities has the following structure:

```
   +--rw i2nsf-net-sec-caps
         +--rw net-sec-capabilities* [nsc-capabilities-name]
            +--rw nsc-capabilities-name    string
            +--rw rule-description?        boolean
            +--rw rule-rev?                boolean
            +--rw rule-priority?           boolean
            +--rw time-zone
            |  +--rw absolute-time-zone
            |  |  +--rw time?   boolean
            |  |  +--rw date?   boolean
            |  +--rw periodic-time-zone
            |     +--rw day?    boolean
            |     +--rw month?  boolean
            +--rw event
            |  ...
            +--rw condition
            |  ...
            +--rw action
            |  +--rw (action-type)?
            |     +--:(ingress-action)
            |     |  +--rw ingress-manual?        string
            |     |  +--rw ingress-action-type
            |     |     +--rw pass?     boolean
            |     |     +--rw drop?     boolean
            |     |     +--rw reject?   boolean
            |     |     +--rw alert?    boolean
            |     |     +--rw mirror?   boolean
            |     +--:(egress-action)
            |        +--rw egress-manual?         string
            |        +--rw egress-action-type
            |           +--rw invoke-signaling?       boolean
            |           +--rw tunnel-encapsulation?   boolean
            |           +--rw forwarding?             boolean
            |           +--rw redirection?            boolean
            +--rw resolution-strategy
            |     ...
            +--rw default-action
                  ...
```

Figure 6: Data Model Structure for Action Capabilities of Network Security Function

These objects are defined capabilities as ingress action, egress action, and apply profile action. These objects can be extended according to specific vendor action feature. We will add additional action objects for more generic network security functions.

6.2.4. Resolution Strategy Capabilities

The data model for resolution strategy capabilities has the following structure:

```
+--rw i2nsf-net-sec-caps
      +--rw net-sec-capabilities* [nsc-capabilities-name]
         +--rw nsc-capabilities-name    string
         +--rw rule-description?        boolean
         +--rw rule-rev?                boolean
         +--rw rule-priority?           boolean
         +--rw time-zone
         |  +--rw absolute-time-zone
         |  |  +--rw time?   boolean
         |  |  +--rw date?   boolean
         |  +--rw periodic-time-zone
         |     +--rw day?    boolean
         |     +--rw month?  boolean
         +--rw event
         |  ...
         +--rw condition
         |  ...
         +--rw action
         |  ...
         +--rw resolution-strategy
         |  +--rw first-matching-rule?   boolean
         |  +--rw last-matching-rule?    boolean
         +--rw default-action
               ...
```

Figure 7: Data Model Structure for Resolution Strategy Capabilities of Network Security Function

These objects are defined capabilities as first-matching-rule and last-matching-rule. These objects can be extended according to specific vendor resolution strategy features. We will add additional resolution strategy objects for more generic network security functions.

6.2.5. Default Action Capabilities

The data model for default action capabilities has the following structure:

```
   +--rw i2nsf-net-sec-caps
         +--rw net-sec-capabilities* [nsc-capabilities-name]
            +--rw nsc-capabilities-name    string
            +--rw rule-description?        boolean
            +--rw rule-rev?                boolean
            +--rw rule-priority?           boolean
            +--rw time-zone
            |  +--rw absolute-time-zone
            |  |  +--rw time?   boolean
            |  |  +--rw date?   boolean
            |  +--rw periodic-time-zone
            |     +--rw day?    boolean
            |     +--rw month?  boolean
            +--rw event
            |  ...
            +--rw condition
            |  ...
            +--rw action
            |  ...
            +--rw resolution-strategy
            |  ...
            +--rw default-action
               +--rw default-action-type
                  +--rw ingress-action-type
                     +--rw pass?     boolean
                     +--rw drop?     boolean
                     +--rw reject?   boolean
                     +--rw alert?    boolean
                     +--rw mirror?   boolean
```

Figure 8: Data Model Structure for Default Action Capabilities of Network Security Function

6.2.6. RPC for Acquiring Appropriate Network Security Function

The data model for RPC for Acquiring Appropriate Network Security Function has the following structure:

```
     rpcs:
       +---x call-appropriate-nsf
          +---w input
          |  +---w nsf-type         nsf-type
          |  +---w target-device
          |     +---w pc?                 boolean
          |     +---w mobile-phone?       boolean
          |     +---w voip-volte-phone?   boolean
          |     +---w tablet?             boolean
          |     +---w iot?                boolean
          |     +---w vehicle?            boolean
          +--ro output
             +--ro nsf-address
                +--ro (nsf-address-type)?
                   +--:(ipv4-address)
                   |  +--ro ipv4-address    inet:ipv4-address
                   +--:(ipv6-address)
                      +--ro ipv6-address    inet:ipv6-address
```

Figure 9: RPC for Acquiring Appropriate Network Security Function

This shows a RPC for acquiring an appropriate network security function according to type of NSF and/or target devices. If the SFF [i2nsf-sfc]does not have the location information of network security functions that it should send in own cache table, this can be used to acquire the information. These objects are defined as input data (i.e., NSF type and target devices) and output data (i.e., location information of NSF).

7. YANG Modules

7.1. I2NSF Capability YANG Data Module

This section introduces a YANG module for the information model of network security functions, as defined in the [i2nsf-nsf-cap-im].

```
<CODE BEGINS> file "ietf-i2nsf-capability@2018-03-05.yang"


module ietf-i2nsf-capability {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability";
  prefix
    i2nsf-capability;

  import ietf-inet-types{
    prefix inet;
  }
```

```
     organization
       "IETF I2NSF (Interface to Network Security Functions)
        Working Group";

     contact
       "WG Web: <http://tools.ietf.org/wg/i2nsf>
        WG List: <mailto:i2nsf@ietf.org>

        WG Chair: Adrian Farrel
        <mailto:Adrain@olddog.co.uk>

        WG Chair: Linda Dunbar
        <mailto:Linda.duhbar@huawei.com>

        Editor: Susan Hares
        <mailto:shares@ndzh.com>

        Editor: Jaehoon Paul Jeong
        <mailto:pauljeong@skku.edu>

        Editor: Jinyong Tim Kim
        <mailto:timkim@skku.edu>";

     description
       "This module describes a capability model
       for I2NSF devices.";

     revision "2018-03-05"{
       description "The fourth revision";
       reference
         "draft-ietf-i2nsf-capability-00";
     }



     grouping i2nsf-nsf-location {
       description
         "This provides a location for capabilities.";
       container nsf-address {
         description
          "This is location information for capabilities.";
         choice nsf-address-type {
           description
             "nsf address type: ipv4 and ipv4";
           case ipv4-address {
             description
               "ipv4 case";
             leaf ipv4-address {
```

```
              type inet:ipv4-address;
              mandatory true;
              description
                "nsf address type is ipv4";
            }
          }
          case ipv6-address {
            description
              "ipv6 case";
            leaf ipv6-address {
              type inet:ipv6-address;
              mandatory true;
              description
                "nsf address type is ipv6";
            }
          }
        }
      }
    }

    typedef nsf-type {
        type enumeration {
          enum network-firewall {
            description
              "If type of a NSF is Network Firewall.";
          }

          enum web-app-firewall {
            description
              "If type of a NSF is Web Application
              Firewall.";
          }

          enum anti-virus {
            description
              "If type of a NSF is Anti-Virus";
          }

          enum ids {
            description
              "If type of a NSF is IDS.";
          }

          enum ips {
            description
              "If type of a NSF is IPS.";
          }
```

```
          enum ddos-mitigator {
            description
              "If type of a NSF is DDoS Mitigator.";
          }
        }
        description
          "This is used for type of NSF.";
    }

    grouping i2nsf-it-resources {
      description
        "This provides a link between capabilities
         and IT resources. This has a list of IT resources
         by name.";
      container target-device {
        description
          "it-resources";

        leaf pc {
          type boolean;
          description
            "If type of a device is PC.";
        }

        leaf mobile-phone {
          type boolean;
          description
            "If type of a device is mobile-phone.";
        }

        leaf voip-volte-phone {
          type boolean;
          description
            "If type of a device is voip-volte-phone.";
        }

        leaf tablet {
          type boolean;
          description
            "If type of a device is tablet.";
        }


        leaf iot {
          type boolean;
          description
            "If type of a device is Internet of Things.";
        }
```

```
        leaf vehicle {
          type boolean;
          description
            "If type of a device is vehicle.";
        }

      }
    }

    grouping capabilities-information {
      description
        "This includes information of capabilities.";

      leaf nsf-type {
        type nsf-type;
        description
          "This is type of NSF.";
      }
      uses i2nsf-nsf-location;
      uses i2nsf-it-resources;
    }

    grouping i2nsf-net-sec-caps {
      description
        "i2nsf-net-sec-caps";
      list net-sec-capabilities {
        key "nsc-capabilities-name";
        description
          "net-sec-capabilities";
        leaf nsc-capabilities-name {
          type string;
          mandatory true;
          description
            "nsc-capabilities-name";
        }

        leaf rule-description {
          type boolean;
          description
            "This is rule-description.";
        }
        leaf rule-rev {
          type boolean;
          description
            "This is rule-revision";
        }
        leaf rule-priority {
          type boolean;
```

```
          description
            "This is rule-priority";
        }

        container time-zone {
          description
            "This can be used to apply rules according to time zone";
          container absolute-time-zone {
            description
              "This can be used to apply rules according to
               absolute time zone";
            leaf time {
              type boolean;
              description
                "This is time for absolute time zone";
            }
            leaf date {
              type boolean;
              description
                "This is date for absolute time zone";
            }
          }
          container periodic-time-zone {
            description
              "This can be used to apply rules according to
               periodic time zone";
            leaf day {
              type boolean;
              description
                "This is day for periodic time zone";
            }
            leaf month {
              type boolean;
              description
                "This is month for periodic time zone";
            }
          }
        }

        container event {
          description
            " This is abstract. An event is defined as any important
              occurrence in time of a change in the system being
              managed, and/or in the environment of the system being
              managed. When used in the context of policy rules for
              a flow-based NSF, it is used to determine whether the
              Condition clause of the Policy Rule can be evaluated
              or not. Examples of an I2NSF event include time and
```

```
              user actions (e.g., logon, logoff, and actions that
              violate any ACL.).";


          choice event-type {
            description
              "Vendors can use YANG data model to configure rules
              by concreting this event type";
            case usr-event {
              leaf usr-manual {
                type string;
                description
                  "This is manual for user event.
                  Vendors can write instructions for user event
                  that vendor made";
              }

              leaf usr-sec-event-content {
                type boolean;
                description
                 "This is a mandatory string that contains the content
                  of the UserSecurityEvent. The format of the content
                  is specified in the usrSecEventFormat class
                  attribute, and the type of event is defined in the
                  usrSecEventType class attribute. An example of the
                  usrSecEventContent attribute is a string hrAdmin,
                  with the usrSecEventFormat set to 1 (GUID) and the
                  usrSecEventType attribute set to 5 (new logon).";
              }

              container usr-sec-event-format {
                description
                 "This is a mandatory uint 8 enumerated integer, which
                  is used to specify the data type of the
                  usrSecEventContent attribute. The content is
                  specified in the usrSecEventContent class attribute,
                  and the type of event is defined in the
                  usrSecEventType class attribute. An example of the
                  usrSecEventContent attribute is string hrAdmin,
                  with the usrSecEventFormat attribute set to 1 (GUID)
                  and the usrSecEventType attribute set to 5
                  (new logon).";
                leaf unknown {
                  type boolean;
                  description
                    "If SecEventFormat is unknown";
                }
                leaf guid {
```

```
              type boolean;
              description
                "If SecEventFormat is GUID
                (Generic Unique IDentifier)";
            }
            leaf uuid {
              type boolean;
              description
                "If SecEventFormat is UUID
                (Universal Unique IDentifier)";
            }
            leaf uri {
              type boolean;
              description
                "If SecEventFormat is URI
                (Uniform Resource Identifier)";
            }
            leaf fqdn {
              type boolean;
              description
                "If SecEventFormat is FQDN
                (Fully Qualified Domain Name)";
            }
            leaf fqpn {
              type boolean;
              description
                "If SecEventFormat is FQPN
                (Fully Qualified Path Name)";
            }
          }

          container usr-sec-event-type {
            leaf unknown {
                type boolean;
                description
                  "If usrSecEventType is unknown";
            }
            leaf user-created {
                type boolean;
                description
                  "If usrSecEventType is new user
                  created";
            }
            leaf user-grp-created {
                type boolean;
                description
                  "If usrSecEventType is new user
                  group created";
```

```
                }
                leaf user-deleted {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      deleted";
                }
                leaf user-grp-deleted {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      group deleted";
                }
                leaf user-logon {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      logon";
                }
                leaf user-logoff {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      logoff";
                }
                leaf user-access-request {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      access request";
                }
                leaf user-access-granted {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      granted";
                }
                leaf user-access-violation {
                    type boolean;
                    description
                      "If usrSecEventType is user
                      violation";
                }
                description
                 "This is a mandatory uint 8 enumerated integer, which
                  is used to specify the type of event that involves
                  this user. The content and format are specified in
                  the usrSecEventContent and usrSecEventFormat class
```

```
                  attributes, respectively. An example of the
                  usrSecEventContent attribute is string hrAdmin,
                  with the usrSecEventFormat attribute set to 1 (GUID)
                  and the usrSecEventType attribute set to 5
                 (new logon).";
              }


            }
            case dev-event {
              leaf dev-manual {
                type string;
                description
                  "This is manual for device event.
                  Vendors can write instructions for device event
                  that vendor made";
              }

              leaf dev-sec-event-content {
                type boolean;
                mandatory true;
                description
                 "This is a mandatory string that contains the content
                  of the DeviceSecurityEvent. The format of the
                  content is specified in the devSecEventFormat class
                  attribute, and the type of event is defined in the
                  devSecEventType class attribute. An example of the
                  devSecEventContent attribute is alarm, with the
                  devSecEventFormat attribute set to 1 (GUID), the
                  devSecEventType attribute set to 5 (new logon).";
              }

              container dev-sec-event-format {
                description
                 "This is a mandatory uint 8 enumerated integer,
                  which is used to specify the data type of the
                  devSecEventContent attribute.";

                leaf unknown {
                  type boolean;
                  description
                    "If SecEventFormat is unknown";
                }
                leaf guid {
                  type boolean;
                  description
                    "If SecEventFormat is GUID
                    (Generic Unique IDentifier)";
```

```
                }
                leaf uuid {
                  type boolean;
                  description
                    "If SecEventFormat is UUID
                    (Universal Unique IDentifier)";
                }
                leaf uri {
                  type boolean;
                  description
                    "If SecEventFormat is URI
                    (Uniform Resource Identifier)";
                }
                leaf fqdn {
                  type boolean;
                  description
                    "If SecEventFormat is FQDN
                    (Fully Qualified Domain Name)";
                }
                leaf fqpn {
                  type boolean;
                  description
                    "If SecEventFormat is FQPN
                    (Fully Qualified Path Name)";
                }
              }

              container dev-sec-event-type {
                description
                 "This is a mandatory uint 8 enumerated integer,
                  which is used to specify the type of event
                  that was generated by this device.";

                leaf unknown {
                    type boolean;
                    description
                      "If devSecEventType is unknown";
                }
                leaf comm-alarm {
                    type boolean;
                    description
                      "If devSecEventType is communications
                      alarm";
                }
                leaf quality-of-service-alarm {
                    type boolean;
                    description
                      "If devSecEventType is quality of service
```

```
                      alarm";
                }
                leaf process-err-alarm {
                    type boolean;
                    description
                      "If devSecEventType is processing error
                      alarm";
                }
                leaf equipment-err-alarm {
                    type boolean;
                    description
                      "If devSecEventType is equipment error
                      alarm";
                }
                leaf environmental-err-alarm {
                    type boolean;
                    description
                      "If devSecEventType is environmental error
                      alarm";
                }

              }

              container dev-sec-event-type-severity {
                description
                 "This is a mandatory uint 8 enumerated integer,
                  which is used to specify the perceived
                  severity of the event generated by this
                  Device.";

                leaf unknown {
                    type boolean;
                    description
                      "If devSecEventType is unknown";
                }
                leaf cleared {
                    type boolean;
                    description
                      "If devSecEventTypeSeverity is cleared";
                }
                leaf indeterminate {
                    type boolean;
                    description
                      "If devSecEventTypeSeverity is
                      indeterminate";
                }
                leaf critical {
                    type boolean;
```

```
                    description
                      "If devSecEventTypeSeverity is critical";
                }
                leaf major{
                    type boolean;
                    description
                      "If devSecEventTypeSeverity is major";
                }
                leaf minor {
                    type boolean;
                    description
                      "If devSecEventTypeSeverity is minor";
                }
                leaf warning {
                    type boolean;
                    description
                      "If devSecEventTypeSeverity is warning";
                }
              }
            }
            case sys-event {
              leaf sys-manual {
                type string;
                description
                  "This is manual for system event.
                  Vendors can write instructions for system event
                  that vendor made";
              }

              leaf sys-sec-event-content {
                type boolean;
                description
                 "This is a mandatory string that contains a content
                  of the SystemSecurityEvent. The format of a content
                  is specified in a sysSecEventFormat class attribute,
                  and the type of event is defined in the
                  sysSecEventType class attribute. An example of the
                  sysSecEventContent attribute is string sysadmin3,
                  with the sysSecEventFormat attribute set to 1(GUID),
                  and the sysSecEventType attribute set to 2
                  (audit log cleared).";
              }

              container sys-sec-event-format {
                description
                 "This is a mandatory uint 8 enumerated integer, which
                  is used to specify the data type of the
                  sysSecEventContent attribute.";
```

```
                leaf unknown {
                  type boolean;
                  description
                    "If SecEventFormat is unknown";
                }
                leaf guid {
                  type boolean;
                  description
                    "If SecEventFormat is GUID
                    (Generic Unique IDentifier)";
                }
                leaf uuid {
                  type boolean;
                  description
                    "If SecEventFormat is UUID
                    (Universal Unique IDentifier)";
                }
                leaf uri {
                  type boolean;
                  description
                    "If SecEventFormat is URI
                    (Uniform Resource Identifier)";
                }
                leaf fqdn {
                  type boolean;
                  description
                    "If SecEventFormat is FQDN
                    (Fully Qualified Domain Name)";
                }
                leaf fqpn {
                  type boolean;
                  description
                    "If SecEventFormat is FQPN
                    (Fully Qualified Path Name)";
                }
              }

              container sys-sec-event-type {
                description
                 "This is a mandatory uint 8 enumerated integer, which
                  is used to specify the type of event that involves
                  this device.";

                leaf unknown {
                    type boolean;
                    description
                      "If sysSecEventType is unknown";
                }
```

```
                leaf audit-log-written-to {
                    type boolean;
                    description
                    "If sysSecEventTypeSeverity
                     is that audit log is written to";
                }
                leaf audit-log-cleared {
                    type boolean;
                    description
                    "If sysSecEventTypeSeverity
                     is that audit log is cleared";
                }
                leaf policy-created {
                    type boolean;
                    description
                    "If sysSecEventTypeSeverity
                     is that policy is created";
                }
                leaf policy-edited{
                    type boolean;
                    description
                    "If sysSecEventTypeSeverity
                     is that policy is edited";
                }
                leaf policy-deleted{
                    type boolean;
                    description
                    "If sysSecEventTypeSeverity
                     is that policy is deleted";
                }
                leaf policy-executed{
                    type boolean;
                    description
                    "If sysSecEventTypeSeverity
                     is that policy is executed";
                }
              }
            }
            case time-event {
              leaf time-manual {
                type string;
                description
                  "This is manual for time event.
                  Vendors can write instructions for time event
                  that vendor made";
              }
              leaf time-sec-event-begin {
                type boolean;
```

```
                description
                  "This is a mandatory DateTime attribute, and
                  represents the beginning of a time period.
                  It has a value that has a date and/or a time
                  component (as in the Java or Python libraries).";
              }

              leaf time-sec-event-end {
                type boolean;
                description
                  "This is a mandatory DateTime attribute, and
                   represents the end of a time period. It has
                   a value that has a date and/or a time component
                   (as in the Java or Python libraries). If this is
                   a single event occurrence, and not a time period
                   when the event can occur, then the
                   timeSecEventPeriodEnd attribute may be ignored.";
              }

              leaf time-sec-event-time-zone {
                type boolean;
                description
                  "This is a mandatory string attribute, and defines a
                   time zone that this event occurred in using the
                   format specified in ISO8601.";
              }
            }
          }
        }

        container condition {
          description
            " This is abstract.  A condition is defined as a set
            of attributes, features, and/or values that are to be
            compared with a set of known attributes, features,
            and/or values in order to determine whether or not the
            set of Actions in that (imperative) I2NSF Policy Rule
            can be executed or not. Examples of I2NSF Conditions
            include matching attributes of a packet or flow, and
            comparing the internal state of an NSF to a desired state.";

          choice condition-type {
            description
              "Vendors can use YANG data model to configure rules
              by concreting this condition type";

            case packet-security-condition {
              leaf packet-manual {
```

```
                type string;
                description
                  "This is manual for packet condition.
                  Vendors can write instructions for packet condition
                  that vendor made";
              }

              container packet-security-mac-condition {
                description
                  "The purpose of this Class is to represent packet MAC
                   packet header information that can be used as part of
                   a test to determine if the set of Policy Actions in
                   this ECA Policy Rule should be execute or not.";

                leaf pkt-sec-cond-mac-dest {
                  type boolean;
                  description
                    "The MAC destination address (6 octets long).";
                }

                leaf pkt-sec-cond-mac-src {
                  type boolean;
                  description
                    "The MAC source address (6 octets long).";
                }

                leaf pkt-sec-cond-mac-8021q {
                  type boolean;
                  description
                    "This is an optional string attribute, and defines
                     The 802.1Q tab value (2 octets long).";
                }

                leaf pkt-sec-cond-mac-ether-type {
                  type boolean;
                  description
                    "The EtherType field (2 octets long). Values up to
                     and including 1500 indicate the size of the payload
                     in octets; values of 1536 and above define which
                     protocol is encapsulated in the payload of the
                     frame.";
                }

                leaf pkt-sec-cond-mac-tci {
                  type string;
                  description
                    "This is an optional string attribute, and defines
                     the Tag Control Information. This consists of a 3
```

```
                     bit user priority field, a drop eligible indicator
                     (1 bit), and a VLAN identifier (12 bits).";
                }
              }

              container packet-security-ipv4-condition {
                description
                  "The purpose of this Class is to represent packet IPv4
                   packet header information that can be used as part of
                   a test to determine if the set of Policy Actions in
                   this ECA Policy Rule should be executed or not.";

                leaf pkt-sec-cond-ipv4-header-length {
                  type boolean;
                  description
                    "The IPv4 packet header consists of 14 fields,
                     of which 13 are required.";
                }

                leaf pkt-sec-cond-ipv4-tos {
                  type boolean;
                  description
                    "The ToS field could specify a datagram's priority
                     and request a route for low-delay, high-throughput,
                     or highly-reliable service..";
                }

                leaf pkt-sec-cond-ipv4-total-length {
                  type boolean;
                  description
                    "This 16-bit field defines the entire packet size,
                     including header and data, in bytes.";
                }

                leaf pkt-sec-cond-ipv4-id {
                  type boolean;
                  description
                    "This field is an identification field and is
                     primarily used for uniquely identifying
                     the group of fragments of a single IP datagram.";
                }

                leaf pkt-sec-cond-ipv4-fragment {
                  type boolean;
                  description
                    "IP fragmentation is an Internet Protocol (IP)
                     process that breaks datagrams into smaller pieces
                     (fragments), so that packets may be formed that
```

```
                     can pass through a link with a smaller maximum
                     transmission unit (MTU) than the original
                     datagram size.";
                }

                leaf pkt-sec-cond-ipv4-fragment-offset {
                  type boolean;
                  description
                    "Fragment offset field along with Don’t Fragment
                     and More Fragment flags in the IP protocol
                     header are used for fragmentation and reassembly
                     of IP datagrams.";
                }

                leaf pkt-sec-cond-ipv4-ttl {
                  type boolean;
                  description
                    "The ttl keyword is used to check for a specific
                     IP time-to-live value in the header of
                     a packet.";
                }

                leaf pkt-sec-cond-ipv4-protocol {
                  type boolean;
                  description
                    "Internet Protocol version 4(IPv4) is the fourth
                     version of the Internet Protocol (IP).";
                }

                leaf pkt-sec-cond-ipv4-src {
                  type boolean;
                  description
                    "Defines the IPv4 Source Address.";
                }

                leaf pkt-sec-cond-ipv4-dest {
                  type boolean;
                  description
                    "Defines the IPv4 Destination Address.";
                }

                leaf pkt-sec-cond-ipv4-ipopts {
                  type boolean;
                  description
                    "With the ipopts keyword you can check if
                     a specific ip option is set. Ipopts has
                     to be used at the beginning of a rule.";
                }
```

```
                leaf pkt-sec-cond-ipv4-sameip {
                  type boolean;
                  description
                    "Every packet has a source IP-address and
                     a destination IP-address. It can be that
                     the source IP is the same as
                     the destination IP.";
                }

                leaf pkt-sec-cond-ipv4-geoip {
                  type boolean;
                  description
                    "The geoip keyword enables you to match on
                     the source, destination or source and destination
                     IP addresses of network traffic and to see to
                     which country it belongs. To do this, Suricata
                     uses GeoIP API with MaxMind database format.";
                }
              }

              container packet-security-ipv6-condition {
                description
                   "The purpose of this Class is to represent packet
                   IPv6 packet header information that can be used as
                   part of a test to determine if the set of Policy
                   Actions in this ECA Policy Rule should be executed
                   or not.";

                leaf pkt-sec-cond-ipv6-dscp {
                  type boolean;
                  description
                    "Differentiated Services Code Point (DSCP)
                     of ipv6.";
                }

                leaf pkt-sec-cond-ipv6-ecn {
                  type boolean;
                  description
                    "ECN allows end-to-end notification of network
                     congestion without dropping packets.";
                }

                leaf pkt-sec-cond-ipv6-traffic-class {
                  type boolean;
                  description
                    "The bits of this field hold two values. The 6
                     most-significant bits are used for
                     differentiated services, which is used to
```

```
                        classify packets.";
                   }

                   leaf pkt-sec-cond-ipv6-flow-label {
                     type boolean;
                     description
                       "The flow label when set to a non-zero value
                        serves as a hint to routers and switches
                        with multiple outbound paths that these
                        packets should stay on the same path so that
                        they will not be reordered.";
                   }

                   leaf pkt-sec-cond-ipv6-payload-length {
                     type boolean;
                     description
                       "The size of the payload in octets,
                        including any extension headers.";
                   }

                   leaf pkt-sec-cond-ipv6-next-header {
                     type boolean;
                     description
                       "Specifies the type of the next header.
                        This field usually specifies the transport
                        layer protocol used by a packet's payload.";
                   }

                   leaf pkt-sec-cond-ipv6-hop-limit {
                     type boolean;
                     description
                       "Replaces the time to live field of IPv4.";
                   }

                   leaf pkt-sec-cond-ipv6-src {
                     type boolean;
                     description
                       "The IPv6 address of the sending node.";
                   }

                   leaf pkt-sec-cond-ipv6-dest {
                     type boolean;
                     description
                       "The IPv6 address of the destination node(s).";
                   }
                 }

                 container packet-security-tcp-condition {
```

```
                description
                  "The purpose of this Class is to represent packet
                   TCP packet header information that can be used as
                   part of a test to determine if the set of Policy
                   Actions in this ECA Policy Rule should be executed
                   or not.";

                leaf pkt-sec-cond-tcp-src-port {
                  type boolean;
                  description
                    "This is a mandatory string attribute, and
                    defines the Source Port number (16 bits).";
                }

                leaf pkt-sec-cond-tcp-dest-port {
                  type boolean;
                  description
                    "This is a mandatory string attribute, and
                    defines the Destination Port number (16 bits).";
                }

                leaf pkt-sec-cond-tcp-seq-num {
                  type boolean;
                  description
                    "If the SYN flag is set (1), then this is the
                     initial sequence number.";
                }

                leaf pkt-sec-cond-tcp-ack-num {
                  type boolean;
                  description
                    "If the ACK flag is set then the value of this
                     field is the next sequence number that the sender
                     is expecting.";
                }

                leaf pkt-sec-cond-tcp-window-size {
                  type boolean;
                  description
                    "The size of the receive window, which specifies
                     the number of windows size units (by default,bytes)
                     (beyond the segment identified by the sequence
                     number in the acknowledgment field) that the sender
                     of this segment is currently willing to recive.";
                }

                leaf pkt-sec-cond-tcp-flags {
                  type boolean;
```

```
            description
              "This is a mandatory string attribute, and defines
               the nine Control bit flags (9 bits).";
          }
        }

        container packet-security-udp-condition {
          description
            "The purpose of this Class is to represent packet UDP
             packet header information that can be used as part
             of a test to determine if the set of Policy Actions
             in this ECA Policy Rule should be executed or not.";

          leaf-list pkt-sec-cond-udp-src-port {
            type boolean;
            description
              "This is a mandatory string attribute, and
              defines the UDP Source Port number (16 bits).";
          }

          leaf-list pkt-sec-cond-udp-dest-port {
            type boolean;
            description
              "This is a mandatory string attribute, and
              defines the UDP Destination Port number (16 bits).";
          }

          leaf pkt-sec-cond-udp-length {
            type boolean;
            description
              "This is a mandatory string attribute, and defines
               the length in bytes of the UDP header and data
               (16 bits).";
          }
        }

        container packet-security-icmp-condition {
          description
            "The internet control message protocol condition.";

          leaf pkt-sec-cond-icmp-type {
            type boolean;
            description
              "ICMP type, see Control messages.";
          }

          leaf pkt-sec-cond-icmp-code {
            type boolean;
```

```
                  description
                    "ICMP subtype, see Control messages.";
                }

                leaf pkt-sec-cond-icmp-seg-num {
                  type boolean;
                  description
                    "The icmp Sequence Number.";
                }
              }
            }

            case packet-payload-condition {
              leaf packet-payload-manual {
                type string;
                description
                  "This is manual for payload condition.
                  Vendors can write instructions for payload condition
                  that vendor made";
              }
              leaf pkt-payload-content {
                type boolean;
                description
                  "The content keyword is very important in
                   signatures. Between the quotation marks you
                   can write on what you would like the
                   signature to match.";
              }
            }
            case target-condition {
              leaf target-manual {
                type string;
                description
                  "This is manual for target condition.
                  Vendors can write instructions for target condition
                  that vendor made";
              }

              leaf device-sec-context-cond {
                type boolean;
                description
                  "The device attribute that can identify a device,
                   including the device type (i.e., router, switch,
                   pc, ios, or android) and the device’s owner as
                   well.";
              }
            }
            case users-condition {
```

```
              leaf users-manual {
                type string;
                description
                  "This is manual for user condition.
                  Vendors can write instructions for user condition
                  that vendor made";
              }

              container user{
                description
                  "The user (or user group) information with which
                   network flow is associated: The user has many
                   attributes such as name, id, password, type,
                   authentication mode and so on. Name/id is often
                   used in the security policy to identify the user.
                   Besides, NSF is aware of the IP address of the
                   user provided by a unified user management system
                   via network. Based on name-address association,
                   NSF is able to enforce the security functions
                   over the given user (or user group)";

                choice user-name {
                  description
                    "The name of the user.
                     This must be unique.";

                  case tenant {
                    description
                      "Tenant information.";

                    leaf tenant {
                      type boolean;
                      description
                        "User’s tenant information.";
                    }
                  }

                  case vn-id {
                    description
                      "VN-ID information.";

                    leaf vn-id {
                      type boolean;
                      description
                        "User’s VN-ID information.";
                    }
                  }
                }
```

```
              }
              container group {
                description
                  "The user (or user group) information with which
                   network flow is associated: The user has many
                   attributes such as name, id, password, type,
                   authentication mode and so on. Name/id is often
                   used in the security policy to identify the user.
                   Besides, NSF is aware of the IP address of the
                   user provided by a unified user management system
                   via network. Based on name-address association,
                   NSF is able to enforce the security functions
                   over the given user (or user group)";

                choice group-name {
                  description
                    "The name of the user.
                     This must be unique.";

                  case tenant {
                    description
                      "Tenant information.";

                    leaf tenant {
                      type boolean;
                      description
                        "User’s tenant information.";
                    }
                  }

                  case vn-id {
                    description
                      "VN-ID information.";

                    leaf vn-id {
                      type boolean;
                      description
                        "User’s VN-ID information.";
                    }
                  }
                }
              }

            }
            case context-condition {
              leaf context-manual {
                type string;
                description
```

```
                  "This is manual for context condition.
                  Vendors can write instructions for context condition
                  that vendor made";
              }
            }
            case gen-context-condition {
              leaf gen-context-manual {
                type string;
                description
                  "This is manual for generic context condition.
                  Vendors can write instructions for generic context
                  condition that vendor made";
              }

              container geographic-location {
                description
                  "The location where network traffic is associated
                   with. The region can be the geographic location
                   such as country, province, and city,
                   as well as the logical network location such as
                   IP address, network section, and network domain.";

                leaf src-geographic-location {
                  type boolean;
                  description
                    "This is mapped to ip address. We can acquire
                     source region through ip address stored the
                     database.";
                }
                leaf dest-geographic-location {
                  type boolean;
                  description
                    "This is mapped to ip address. We can acquire
                     destination region through ip address stored
                     the database.";
                }
              }
            }
          }
        }
        container action {
          description
            "An action is used to control and monitor aspects of
             flow-based NSFs when the event and condition clauses
             are satisfied. NSFs provide security functions by
             executing various Actions. Examples of I2NSF Actions
             include providing intrusion detection and/or protection,
             web and flow filtering, and deep packet inspection
```

```
             for packets and flows.";

          choice action-type {
            description
              "Vendors can use YANG data model to configure rules
              by concreting this action type";
            case ingress-action {
              leaf ingress-manual {
                type string;
                description
                  "This is manual for ingress action.
                  Vendors can write instructions for ingress action
                  that vendor made";
              }
              container ingress-action-type {
                description
                  "Ingress action type: permit, deny, and mirror.";
                leaf pass {
                  type boolean;
                  description
                    "If ingress action is pass";
                }
                leaf drop {
                  type boolean;
                  description
                    "If ingress action is drop";
                }
                leaf reject {
                  type boolean;
                  description
                    "If ingress action is reject";
                }
                leaf alert {
                  type boolean;
                  description
                    "If ingress action is alert";
                }
                leaf mirror {
                  type boolean;
                  description
                    "If ingress action is mirror";
                }
              }
            }
            case egress-action {
              leaf egress-manual {
                type string;
```

```
                description
                  "This is manual for egress action.
                  Vendors can write instructions for egress action
                  that vendor made";
              }
              container egress-action-type {
                description
                  "Egress-action-type: invoke-signaling,
                   tunnel-encapsulation, and forwarding.";
                leaf invoke-signaling {
                  type boolean;
                  description
                    "If egress action is invoke signaling";
                }
                leaf tunnel-encapsulation {
                  type boolean;
                  description
                    "If egress action is tunnel encapsulation";
                }
                leaf forwarding {
                  type boolean;
                  description
                    "If egress action is forwarding";
                }
                leaf redirection {
                  type boolean;
                  description
                    "If egress action is redirection";
                }
              }
            }
          }
        }
        container resolution-strategy {
          description
            "The resolution strategies can be used to
            specify how to resolve conflicts that occur between
            the actions of the same or different policy rules that
            are matched and contained in this particular NSF";

          leaf first-matching-rule {
            type boolean;
            description
              "If the resolution strategy is first matching rule";
          }

          leaf last-matching-rule {
            type boolean;
```

```
            description
              "If the resolution strategy is last matching rule";
          }
        }
        container default-action {
          description
            "This default action can be used to specify a predefined
            action when no other alternative action was matched
            by the currently executing I2NSF Policy Rule. An analogy
            is the use of a default statement in a C switch statement.";

          container default-action-type {
            description
              "Ingress action type: permit, deny, and mirror.";

            container ingress-action-type {
              description
                "Ingress action type: permit, deny, and mirror.";
              leaf pass {
                type boolean;
                description
                  "If ingress action is pass";
              }
              leaf drop {
                type boolean;
                description
                  "If ingress action is drop";
              }
              leaf reject {
                type boolean;
                description
                  "If ingress action is reject";
              }
              leaf alert {
                type boolean;
                description
                  "If ingress action is alert";
              }
              leaf mirror {
                type boolean;
                description
                  "If ingress action is mirror";
              }
            }
          }
        }
      }
    }
```

```
    grouping i2nsf-con-sec-control-caps {
      description
        "i2nsf-con-sec-control-caps";

      container con-sec-control-capabilities {
        description
          "content-security-control-capabilities";


        leaf anti-virus {
          type boolean;
          description
            "antivirus";
        }
        leaf ips {
          type boolean;
          description
            "ips";
        }

        leaf ids {
          type boolean;
          description
            "ids";
        }

        leaf url-filter {
          type boolean;
          description
            "url-filter";
        }
        leaf data-filter {
          type boolean;
          description
            "data-filter";
        }
        leaf mail-filter {
          type boolean;
          description
            "mail-filter";
        }
        leaf sql-filter {
          type boolean;
          description
            "sql-filter";
        }
        leaf file-blocking {
          type boolean;
```

```
          description
            "file-blocking";
        }
        leaf file-isolate {
          type boolean;
          description
            "file-isolate";
        }
        leaf pkt-capture {
          type boolean;
          description
            "pkt-capture";
        }
        leaf application-behavior {
          type boolean;
          description
            "application-behavior";
        }
        leaf voip-volte {
          type boolean;
          description
            "voip-volte";
        }
      }


    }

    grouping i2nsf-attack-mitigation-control-caps {
      description
        "i2nsf-attack-mitigation-control-caps";

      container attack-mitigation-capabilities {
        description
          "attack-mitigation-capabilities";
        choice attack-mitigation-control-type {
          description
            "attack-mitigation-control-type";
          case ddos-attack {
            description
              "ddos-attack";
            choice ddos-attack-type {
              description
                "ddos-attack-type";
              case network-layer-ddos-attack {
                description
                  "network-layer-ddos-attack";
                container network-layer-ddos-attack-types {
```

```
                  description
                    "network-layer-ddos-attack-type";
                  leaf syn-flood-attack {
                    type boolean;
                    description
                      "syn-flood-attack";
                  }
                  leaf udp-flood-attack {
                    type boolean;
                    description
                      "udp-flood-attack";
                  }
                  leaf icmp-flood-attack {
                    type boolean;
                    description
                      "icmp-flood-attack";
                  }
                  leaf ip-fragment-flood-attack {
                    type boolean;
                    description
                      "ip-fragment-flood-attack";
                  }
                  leaf ipv6-related-attack {
                    type boolean;
                    description
                      "ip-fragment-flood-attack";
                  }
                }
              }
              case app-layer-ddos-attack {
                description
                  "app-layer-ddos-attack";
                container app-layer-ddos-attack-types {
                  description
                    "app-layer-ddos-attack-types";
                  leaf http-flood-attack {
                    type boolean;
                    description
                      "http-flood-attack";
                  }
                  leaf https-flood-attack {
                    type boolean;
                    description
                      "https-flood-attack";
                  }
                  leaf dns-flood-attack {
                    type boolean;
                    description
```

```
                    "dns-flood-attack";
                }
                leaf dns-amp-flood-attack {
                  type boolean;
                  description
                    "dns-amp-flood-attack";
                }
                leaf ssl-flood-attack {
                  type boolean;
                  description
                    "ssl-flood-attack";
                }
              }
            }
          }
        }

        case single-packet-attack {
          description
            "single-packet-attack";
          choice single-packet-attack-type {
            description
              "single-packet-attack-type";
            case scan-and-sniff-attack {
              description
                "scan-and-sniff-attack";
              leaf ip-sweep-attack {
                type boolean;
                description
                  "ip-sweep-attack";
              }
              leaf port-scanning-attack {
                type boolean;
                description
                  "port-scanning-attack";
              }
            }
            case malformed-packet-attack {
              description
                "malformed-packet-attack";
              leaf ping-of-death-attack {
                type boolean;
                description
                  "ping-of-death-attack";
              }
              leaf teardrop-attack {
                type boolean;
                description
```

```
                    "teardrop-attack";
                }
              }
              case special-packet-attack {
                description
                  "special-packet-attack";
                leaf oversized-icmp-attack {
                  type boolean;
                  description
                    "oversized-icmp-attack";
                }
                leaf tracert-attack {
                  type boolean;
                  description
                    "tracert-attack";
                }
              }
            }
          }
        }
      }
    }


    list nsf {
      key "nsf-name";
      description
        "nsf-name";
      leaf nsf-name {
        type string;
        mandatory true;
        description
          "nsf-name";
      }
      uses capabilities-information;

      container generic-nsf-capabilities {
        description
          "generic-nsf-capabilities";
        uses i2nsf-net-sec-caps;
      }
    }


    rpc call-appropriate-nsf {
      description
        "We can acquire appropriate NSF that we want
        If we give type of NSF that we want to use,
```

```
        we acquire the location information of NSF";

      input {
          leaf nsf-type {
              type nsf-type;
              mandatory true;
              description
                "This is used to acquire NSF
                This is mandatory";
          }
          uses i2nsf-it-resources;
      }
      output {
          uses i2nsf-nsf-location;
      }
   }
  }
```

<CODE ENDS>

Figure 10: YANG Data Module of I2NSF Capability

8. IANA Considerations

No IANA considerations exist for this document at this time. URL will be added.

9. Security Considerations

This document introduces no additional security threats and SHOULD follow the security requirements as stated in [RFC8329].

10. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

11. Contributors

I2NSF is a group effort. I2NSF has had a number of contributing authors. The following are considered co-authors:

- o Hyoungshick Kim (Sungkyunkwan University)
- o Daeyoung Hyun (Sungkyunkwan University)
- o Dongjin Hong (Sungkyunkwan University)
- o Liang Xia (Huawei)
- o Jung-Soo Park (ETRI)
- o Tae-Jin Ahn (Korea Telecom)
- o Se-Hui Lee (Korea Telecom)

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, July 2017.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

12.2. Informative References

[i2nsf-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-00 (work in progress), September 2017.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[i2rs-rib-data-model] Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-10 (work in progress), February 2018.

[supa-policy-info-model] Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-03 (work in progress), May 2017.

[i2nsf-sfc] Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-05 (work in progress), March 2018.

[i2nsf-nsf-yang] Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-kim-i2nsf-nsf-facing-interface-data-model-05 (work in progress), March 2018.

Appendix A. Example: Extended VoIP-VoLTE Security Function Capabilities Module

This section gives a simple example of how VoIP-VoLTE Security Function Capabilities module could be extended.

```
module
ex-voip-volte-capa {
  namespace "http://example.com/voip-volte-capa";
  prefix "voip-volte-capa";

  import ietf-i2nsf-capability {
    prefix capa;
  }
```

```
       augment "/capa:nsf/capa:generic-nsf-capabilities/"
               + "capa:net-sec-control-capabilities/"
               + "capa:condition/capa:condition-type" {
         case voice-condition {
           leaf sip-header-method {
             type boolean;
             description
               "SIP header method.";
           }

           leaf sip-header-uri {
             type boolean;
             description
               "SIP header URI.";
           }

           leaf sip-header-from {
             type boolean;
             description
               "SIP header From.";
           }

           leaf sip-header-to {
             type boolean;
             description
               "SIP header To.";
           }

           leaf sip-header-expire-time {
             type boolean;
             description
               "SIP header expire time.";
           }

           leaf sip-header-user-agent {
             type boolean;
             description
               "SIP header user agent.";
           }
         }
       }
     }
```

Figure 11: Example: Extended VoIP-VoLTE Security Function Capabilities Module

Appendix B. Example: Configuration XML of Capability Module

This section gives a xml examples for a configuration of Capability module according to a requirement.

B.1. Example: Configuration XML of Generic Network Security Function Capabilities

This section gives a xml example for generic network security function capability configuration according to a requirement.

Requirement: Register packet filter according to requirements.

1. The location of the NSF is 221.159.112.150.

2. The NSF can obtain the best effect if the packet was generated by PC or IoT.

3. The NSF can apply policies according to time.

4. The NSF should be able to block the source packets or destination packets with IPv4 address.

5. The NSF should be able to pass, reject, or alert packets.

6. Here is XML example for the generic network security function capability configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
 <target>
  <running />
 </target>
 <config>
  <nsf xmlns="urn:ietf:params:xml:ns:yang:" +
                     "ietf-i2nsf-capability">
    <nsf-name>Huawei-Firewall</nsf-name>
    <nsf-address>
     <ipv4-address>221.159.112.150</ipv4-address>
    </nsf-address>
    <target-device>
     <pc>true</pc>
    </target-device>
    <target-device>
     <iot>true</iot>
    </target-device>
    <generic-nsf-capabilities>
     <net-sec-control-capabilities>
      <nsc-capabilities-name>ipv4-packet-filter<nsc-capabilities-name>
      <time-zone>
       <start-time>true</start-time>
       <end-time>true</end-time>
      </time-zone>
      <condition>
        <packet-security-ipv4-condition>
         <pkt-sec-cond-ipv4-src>true</pkt-sec-cond-ipv4-src>
         <pkt-sec-cond-ipv4-dest>true</pkt-sec-cond-ipv4-dest>
        </packet-security-ipv4-condition>
      </condition>
      <action>
       <ingress-action-type>
        <pass>true</pass>
        <reject>true</reject>
        <alert>true</alert>
       </ingress-action-type>
      </action>
    </net-sec-control-capabilities>
   </generic-nsf-capabilities>
  </nsf>
 </config>
</edit-config>
</rpc>
```


Figure 12: Example: Configuration XML for Generic Network Security Function Capability

B.2. Example: Configuration XML of Extended VoIP/VoLTE Security Function Capabilities Module

This section gives a xml example for extended VoIP-VoLTE security function capabilities (See Figure 11) configuration according to a requirement.

Requirement: Register VoIP/VoLTe security function according to requirements.

1. The location of the NSF is 221.159.112.151.

2. The NSF can obtain the best effect if the packet was generated by VoIP-VoLTE phone.

3. The NSF should be able to block the malicious sip packets with user agent.

4. The NSF should be able to pass, reject, or alert packets.

Here is XML example for the VoIP-VoLTE security function capabilities configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
 <target>
  <running />
 </target>
 <config>
  <nsf xmlns="urn:ietf:params:xml:ns:yang:" +
             "ietf-i2nsf-capability">
  <nsf-name>Cisco-VoIP-VoLTE</nsf-name>
  <nsf-address>
    <ipv4-address>221.159.112.151</ipv4-address>
  </nsf-address>
  <generic-nsf-capabilities>
   <net-sec-control-capabilities>
    <nsc-capabilities-name>sip-packet-filter<nsc-capabilities-name>
     <condition>
       <sip-header-user-agent>true</sip-header-user-agent>
     </condition>
     <action>
      <ingress-action-type>
        <pass>true</pass>
        <reject>true</reject>
        <alert>true</alert>
      </ingress-action-type>
     </action>
   </net-sec-control-capabilities>
  </generic-nsf-capabilities>
  </nsf>
 </config>
</edit-config>
</rpc>
```

Figure 13: Example: Configuration XML for Extended VoIP/VoLTE Security Function Capabilities

Appendix C. Changes from draft-hares-i2nsf-capability-data-model-05

The following changes are made from draft-hares-i2nsf-capability-data-model-05:

1. We modified the data model to support not only absolute time zone but also periodic time zone.

2. We added port number to the condition clause.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
EMail: shares@ndzh.com

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Jinyong Tim Kim
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 8273 0930
EMail: timkim@skku.edu

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

Phone: +1-248-968-9809
EMail: rgm@htt-consult.com

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

Phone:
EMail: linqiushi@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

D. Hong
J. Jeong
J. Kim
Sungkyunkwan University
S. Hares
L. Xia
Huawei
H. Birkholz
Fraunhofer SIT
March 5, 2018

YANG Data Model for Monitoring I2NSF Network Security Functions
draft-hong-i2nsf-nsf-monitoring-data-model-02

Abstract

This document proposes a YANG data model for monitoring Network Security Functions (NSFs) in the Interface to Network Security Functions (I2NSF) system. If the monitoring of NSFs is performed in a comrehensive way, it is possible to detect the indication of malicious activity, anomalous behavior or the potential sign of denial of service attacks in a timely manner. This monitoring functionality is based on the monitoring information that is generated by NSFs. Thus, this document describes not only a data tree to specify an information model for monitoring NSFs, but also the corresponding YANG data model for monitoring NSFs.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for monitoring Network Security Functions (NSFs). This monitoring means the aquisition of vital information about NSFs via notifications, events, records or counters. The data model for the monitoring presented in this document is derived from the information model for monitoring NSFs through the NSF-Facing Interface specified in [i2nsf-monitoring-im].

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 terminology described in [i2nsf-terminology][i2nsf-framework]. Especially, the following terms are from [i2nsf-monitoring-im].

- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.

- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] 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. Information Model Structure

Figure 1 shows the overview of a structure tree of monitoring information based on the [i2nsf-monitoring-im].

```
module: ietf-i2nsf-nsf-monitoring-dm
   +--rw counters
      +--rw system-interface
      |  +--rw interface-name?            string
      |  +--rw in-total-traffic-pkts?     uint32
      |  +--rw out-total-traffic-pkts?    uint32
      |  +--rw in-total-traffic-bytes?    uint32
```

```
         |  +--rw out-total-traffic-bytes?   uint32
         |  +--rw in-drop-traffic-pkts?      uint32
         |  +--rw out-drop-traffic-pkts?     uint32
         |  +--rw in-drop-traffic-bytes?     uint32
         |  +--rw out-drop-traffic-bytes?    uint32
         |  +--rw total-traffic?             uint32
         |  +--rw in-traffic-ave-rate?       uint32
         |  +--rw in-traffic-peak-rate?      uint32
         |  +--rw in-traffic-ave-speed?      uint32
         |  +--rw in-traffic-peak-speed?     uint32
         |  +--rw out-traffic-ave-rate?      uint32
         |  +--rw out-traffic-peak-rate?     uint32
         |  +--rw out-traffic-ave-speed?     uint32
         |  +--rw out-traffic-peak-speed?    uint32
         +--rw nsf-firewall
         |  +--rw src-ip?                   inet:ipv4-address
         |  +--rw dst-ip?                   inet:ipv4-address
         |  +--rw src-port?                 inet:port-number
         |  +--rw dst-port?                 inet:port-number
         |  +--rw src-zone?                 string
         |  +--rw dst-zone?                 string
         |  +--rw src-region?               string
         |  +--rw dst-region?               string
         |  +--rw policy-id                 uint8
         |  +--rw policy-name               string
         |  +--rw src-user?                 string
         |  +--rw protocol
         |  |  +--rw tcp?      boolean
         |  |  +--rw udp?      boolean
         |  |  +--rw icmp?     boolean
         |  |  +--rw icmpv6?   boolean
         |  |  +--rw ip?       boolean
         |  |  +--rw http?     boolean
         |  |  +--rw ftp?      boolean
         |  +--rw total-traffic?            uint32
         |  +--rw in-traffic-ave-rate?      uint32
         |  +--rw in-traffic-peak-rate?     uint32
         |  +--rw in-traffic-ave-speed?     uint32
         |  +--rw in-traffic-peak-speed?    uint32
         |  +--rw out-traffic-ave-rate?     uint32
         |  +--rw out-traffic-peak-rate?    uint32
         |  +--rw out-traffic-ave-speed?    uint32
         |  +--rw out-traffic-peak-speed?   uint32
         |  +--rw diretcions
         |     +--rw in-interface?    boolean
         |     +--rw out-interface?   boolean
         +--rw nsf-policy-hits
            +--rw src-ip?                   inet:ipv4-address
```

```
            +--rw dst-ip?                     inet:ipv4-address
            +--rw src-port?                   inet:port-number
            +--rw dst-port?                   inet:port-number
            +--rw src-zone?                   string
            +--rw dst-zone?                   string
            +--rw src-region?                 string
            +--rw dst-region?                 string
            +--rw policy-id                   uint8
            +--rw policy-name                 string
            +--rw src-user?                   string
            +--rw protocol
            |  +--rw tcp?      boolean
            |  +--rw udp?      boolean
            |  +--rw icmp?     boolean
            |  +--rw icmpv6?   boolean
            |  +--rw ip?       boolean
            |  +--rw http?     boolean
            |  +--rw ftp?      boolean
            +--rw total-traffic?              uint32
            +--rw in-traffic-ave-rate?        uint32
            +--rw in-traffic-peak-rate?       uint32
            +--rw in-traffic-ave-speed?       uint32
            +--rw in-traffic-peak-speed?      uint32
            +--rw out-traffic-ave-rate?       uint32
            +--rw out-traffic-peak-rate?      uint32
            +--rw out-traffic-ave-speed?      uint32
            +--rw out-traffic-peak-speed?     uint32
            +--rw hit-times?                  uint32
   notifications:
      +---n system-detection-alarm
      |  +--ro alarm-catagory?   identityref
      |  +--ro usage?            uint8
      |  +--ro threshold?        uint8
      |  +--ro message           string
      |  +--ro time-stamp        yang:date-and-time
      |  +--ro severity          severity
      +---n system-detection-access-violation
      |  +--ro group             string
      |  +--ro login-ip-addr     inet:ipv4-address
      |  +--ro authentication?   identityref
      |  +--ro message           string
      |  +--ro time-stamp        yang:date-and-time
      |  +--ro severity          severity
      +---n system-detection-config-change
      |  +--ro group             string
      |  +--ro login-ip-addr     inet:ipv4-address
      |  +--ro authentication?   identityref
      |  +--ro message           string
```

```
      |  +--ro time-stamp        yang:date-and-time
      |  +--ro severity          severity
      +---n nsf-detection-flood
      |  +--ro event-message?    string
      |  +--ro src-ip?           inet:ipv4-address
      |  +--ro dst-ip?           inet:ipv4-address
      |  +--ro src-port?         inet:port-number
      |  +--ro dst-port?         inet:port-number
      |  +--ro src-zone?         string
      |  +--ro dst-zone?         string
      |  +--ro rule-id           uint8
      |  +--ro rule-name         string
      |  +--ro profile?          string
      |  +--ro raw-info?         string
      |  +--ro flood-catagory?   identityref
      |  +--ro start-time        yang:date-and-time
      |  +--ro end-time          yang:date-and-time
      |  +--ro attack-rate?      uint32
      |  +--ro attack-speed?     uint32
      |  +--ro vendor-name?      string
      |  +--ro nsf-name?         string
      |  +--ro message           string
      |  +--ro time-stamp        yang:date-and-time
      |  +--ro severity          severity
      +---n nsf-detection-session-table
      |  +--ro current-session?     uint8
      |  +--ro maximum-session?     uint8
      |  +--ro threshold?           uint8
      |  +--ro table-indentifier?   string
      |  +--ro vendor-name?         string
      |  +--ro nsf-name?            string
      |  +--ro message              string
      |  +--ro time-stamp           yang:date-and-time
      |  +--ro severity             severity
      +---n nsf-detection-virus
      |  +--ro event-message?   string
      |  +--ro src-ip?          inet:ipv4-address
      |  +--ro dst-ip?          inet:ipv4-address
      |  +--ro src-port?        inet:port-number
      |  +--ro dst-port?        inet:port-number
      |  +--ro src-zone?        string
      |  +--ro dst-zone?        string
      |  +--ro rule-id          uint8
      |  +--ro rule-name        string
      |  +--ro profile?         string
      |  +--ro raw-info?        string
      |  +--ro virus?           identityref
      |  +--ro virus-name?      string
```

```
       |  +--ro file-type?       string
       |  +--ro file-name?       string
       |  +--ro vendor-name?     string
       |  +--ro nsf-name?        string
       |  +--ro message          string
       |  +--ro time-stamp       yang:date-and-time
       |  +--ro severity         severity
       +---n nsf-detection-intrusion
       |  +--ro event-message?   string
       |  +--ro src-ip?          inet:ipv4-address
       |  +--ro dst-ip?          inet:ipv4-address
       |  +--ro src-port?        inet:port-number
       |  +--ro dst-port?        inet:port-number
       |  +--ro src-zone?        string
       |  +--ro dst-zone?        string
       |  +--ro rule-id          uint8
       |  +--ro rule-name        string
       |  +--ro profile?         string
       |  +--ro raw-info?        string
       |  +--ro protocol
       |  |  +--ro tcp?      boolean
       |  |  +--ro udp?      boolean
       |  |  +--ro icmp?     boolean
       |  |  +--ro icmpv6?   boolean
       |  |  +--ro ip?       boolean
       |  |  +--ro http?     boolean
       |  |  +--ro ftp?      boolean
       |  +--ro intrusion?       identityref
       |  +--ro vendor-name?     string
       |  +--ro nsf-name?        string
       |  +--ro message          string
       |  +--ro time-stamp       yang:date-and-time
       |  +--ro severity         severity
       +---n nsf-detection-botnet
       |  +--ro event-message?   string
       |  +--ro src-ip?          inet:ipv4-address
       |  +--ro dst-ip?          inet:ipv4-address
       |  +--ro src-port?        inet:port-number
       |  +--ro dst-port?        inet:port-number
       |  +--ro src-zone?        string
       |  +--ro dst-zone?        string
       |  +--ro rule-id          uint8
       |  +--ro rule-name        string
       |  +--ro profile?         string
       |  +--ro raw-info?        string
       |  +--ro attack-type?     identityref
       |  +--ro protocol
       |  |  +--ro tcp?      boolean
```

```
|  |  +--ro udp?      boolean
|  |  +--ro icmp?     boolean
|  |  +--ro icmpv6?   boolean
|  |  +--ro ip?       boolean
|  |  +--ro http?     boolean
|  |  +--ro ftp?      boolean
|  +--ro botnet-name?     string
|  +--ro role?            string
|  +--ro vendor-name?     string
|  +--ro nsf-name?        string
|  +--ro message          string
|  +--ro time-stamp       yang:date-and-time
|  +--ro severity         severity
+---n nsf-detection-web-attack
|  +--ro event-message?   string
|  +--ro src-ip?          inet:ipv4-address
|  +--ro dst-ip?          inet:ipv4-address
|  +--ro src-port?        inet:port-number
|  +--ro dst-port?        inet:port-number
|  +--ro src-zone?        string
|  +--ro dst-zone?        string
|  +--ro rule-id          uint8
|  +--ro rule-name        string
|  +--ro profile?         string
|  +--ro raw-info?        string
|  +--ro web-attack?      identityref
|  +--ro protocol
|  |  +--ro tcp?      boolean
|  |  +--ro udp?      boolean
|  |  +--ro icmp?     boolean
|  |  +--ro icmpv6?   boolean
|  |  +--ro ip?       boolean
|  |  +--ro http?     boolean
|  |  +--ro ftp?      boolean
|  +--ro request?         identityref
|  +--ro req-uri?         string
|  +--ro uri-category?    string
|  +--ro filter*          identityref
|  +--ro vendor-name?     string
|  +--ro nsf-name?        string
|  +--ro message          string
|  +--ro time-stamp       yang:date-and-time
|  +--ro severity         severity
+---n system-log-access-event
|  +--ro login-ip          inet:ipv4-address
|  +--ro administrator?    string
|  +--ro login-mode?       login-mode
|  +--ro operation-type?   operation-type
```

```
     |  +--ro result?           string
     |  +--ro content?          string
     |  +--ro vendor-name?      string
     |  +--ro nsf-name?         string
     +---n system-log-res-util-report
     |  +--ro system-status?       string
     |  +--ro cpu-usage?           uint8
     |  +--ro memory-usage?        uint8
     |  +--ro disk-usage?          uint8
     |  +--ro disk-left?           uint8
     |  +--ro session-num?         uint8
     |  +--ro process-num?         uint8
     |  +--ro in-traffic-rate?     uint32
     |  +--ro out-traffic-rate?    uint32
     |  +--ro in-traffic-speed?    uint32
     |  +--ro out-traffic-speed?   uint32
     |  +--ro vendor-name?         string
     |  +--ro nsf-name?            string
     +---n system-log-user-activity-event
     |  +--ro user               string
     |  +--ro group              string
     |  +--ro login-ip           inet:ipv4-address
     |  +--ro authentication?    identityref
     |  +--ro accese?            identityref
     |  +--ro online-duration?   string
     |  +--ro logout-duration?   string
     |  +--ro addtional-info?    string
     |  +--ro vendor-name?       string
     |  +--ro nsf-name?          string
     +---n nsf-log-ddos
     |  +--ro attack-type?        identityref
     |  +--ro attack-ave-rate?    uint32
     |  +--ro attack-ave-speed?   uint32
     |  +--ro attack-pkt-num?     uint32
     |  +--ro attack-src-ip?      inet:ipv4-address
     |  +--ro action?             log-action
     |  +--ro os?                 string
     |  +--ro vendor-name?        string
     |  +--ro nsf-name?           string
     |  +--ro message             string
     |  +--ro time-stamp          yang:date-and-time
     |  +--ro severity            severity
     +---n nsf-log-virus
     |  +--ro attack-type?   identityref
     |  +--ro action?        log-action
     |  +--ro os?            string
     |  +--ro time           yang:date-and-time
     |  +--ro vendor-name?   string
```

```
     |  +--ro nsf-name?      string
     |  +--ro message        string
     |  +--ro time-stamp     yang:date-and-time
     |  +--ro severity       severity
     +---n nsf-log-intrusion
     |  +--ro attack-type?    identityref
     |  +--ro action?         log-action
     |  +--ro time            yang:date-and-time
     |  +--ro attack-rate?    uint32
     |  +--ro attack-speed?   uint32
     |  +--ro vendor-name?    string
     |  +--ro nsf-name?       string
     |  +--ro message         string
     |  +--ro time-stamp      yang:date-and-time
     |  +--ro severity        severity
     +---n nsf-log-botnet
     |  +--ro attack-type?      identityref
     |  +--ro action?           log-action
     |  +--ro botnet-pkt-num?   uint8
     |  +--ro os?               string
     |  +--ro vendor-name?      string
     |  +--ro nsf-name?         string
     |  +--ro message           string
     |  +--ro time-stamp        yang:date-and-time
     |  +--ro severity          severity
     +---n nsf-log-dpi
     |  +--ro attack-type?   dpi-type
     |  +--ro src-ip?        inet:ipv4-address
     |  +--ro dst-ip?        inet:ipv4-address
     |  +--ro src-port?      inet:port-number
     |  +--ro dst-port?      inet:port-number
     |  +--ro src-zone?      string
     |  +--ro dst-zone?      string
     |  +--ro src-region?    string
     |  +--ro dst-region?    string
     |  +--ro policy-id      uint8
     |  +--ro policy-name    string
     |  +--ro src-user?      string
     |  +--ro vendor-name?   string
     |  +--ro nsf-name?      string
     |  +--ro message        string
     |  +--ro time-stamp     yang:date-and-time
     |  +--ro severity       severity
     +---n nsf-log-vuln-scan
     |  +--ro vulnerability-id?   uint8
     |  +--ro victim-ip?          inet:ipv4-address
     |  +--ro protocol
     |  |  +--ro tcp?      boolean
```

```
      |  |  +--ro udp?      boolean
      |  |  +--ro icmp?     boolean
      |  |  +--ro icmpv6?   boolean
      |  |  +--ro ip?       boolean
      |  |  +--ro http?     boolean
      |  |  +--ro ftp?      boolean
      |  +--ro port-num?           inet:port-number
      |  +--ro level?              severity
      |  +--ro os?                 string
      |  +--ro addtional-info?     string
      |  +--ro vendor-name?        string
      |  +--ro nsf-name?           string
      |  +--ro message             string
      |  +--ro time-stamp          yang:date-and-time
      |  +--ro severity            severity
      +---n nsf-log-web-attack
         +--ro attack-type?     identityref
         +--ro rsp-code?        string
         +--ro req-clientapp?   string
         +--ro req-cookies?     string
         +--ro req-host?        string
         +--ro raw-info?        string
         +--ro vendor-name?     string
         +--ro nsf-name?        string
         +--ro message          string
         +--ro time-stamp       yang:date-and-time
         +--ro severity         severity
```

Figure 1: Information Model for NSF Monitoring

5. YANG Data Model

This section introduces a YANG data model for the information model of monitoring inforamtion based on [i2nsf-monitoring-im].

<CODE BEGINS> file "ietf-i2nsf-nsf-monitoring-dm@2018-03-05.yang"

```
module ietf-i2nsf-nsf-monitoring-dm {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring-dm";
  prefix
    monitoring-information;
  import ietf-inet-types{
    prefix inet;
  }
  import ietf-yang-types {
```

```
       prefix yang;
     }
     organization
       "IETF I2NSF (Interface to Network Security Functions)
        Working Group";

     contact
       "WG Web: <http://tools.ietf.org/wg/i2nsf>
        WG List: <mailto:i2nsf@ietf.org>

        WG Chair: Linda Dunbar
        <mailto:Linda.duhbar@huawei.com>

        Editor: Dongjin Hong
        <mailto:dong.jin@skku.edu>

        Editor: Jaehoon Paul Jeong
        <mailto:pauljeong@skku.edu>";

     description
       "This module defines a YANG data module for monitoring NSFs.";

     revision "2018-03-05" {
       description "Third revision";
       reference
         "draft-zhang-i2nsf-info-model-monitoring-05";
     }

     typedef severity {
       type enumeration {
         enum high {
           description
             "high-level";
         }
         enum middle {
           description
             "middle-level";
         }
         enum low {
           description
             "low-level";
         }
       }
       description
         "An indicator representing severity";
     }
     typedef log-action {
       type enumeration {
```

```
          enum allow {
            description
              "If action is allow";
          }
          enum alert {
            description
              "If action is alert";
          }
          enum block {
            description
              "If action is block";
          }
          enum discard {
            description
              "If action is discard";
          }
          enum declare {
            description
              "If action is declare";
          }
          enum block-ip {
            description
              "If action is block-ip";
          }
          enum block-service{
            description
              "If action is block-service";
          }
        }
        description
          "This is used for protocol";
      }
      typedef dpi-type{
        type enumeration {
          enum file-blocking{
            description
              "DPI for blocking file";
          }
          enum data-filtering{
            description
              "DPI for filtering data";
          }
          enum application-behavior-control{
            description
              "DPI for controlling application behavior";
          }
        }
        description
```

```
         "This is used for dpi type";
     }
     typedef operation-type{
       type enumeration {
         enum login{
           description
             "Login operation";
         }
         enum logout{
           description
             "Logout operation";
         }
         enum configuration{
           description
             "Configuration operation";
         }
       }
       description
         "An indicator representing operation-type";
     }
     typedef login-mode{
       type enumeration {
         enum root{
           description
             "Root login-mode";
         }
         enum user{
           description
             "User login-mode";
         }
         enum guest{
           description
             "Guest login-mode";
         }
       }
       description
         "An indicater representing login-mode";
     }
     identity authentication-mode {
       description
         "User authentication mode types: e.g., Local Authentication,
          Third-Party Server Authentication,
          Authentication Exemption, or SSO Authentication.";
     }
     identity local-authentication {
       base authentication-mode;
       description
         "Authentication-mode : local authentication.";
```

```
     }
     identity third-party-server-authentication {
       base authentication-mode;
       description
         "If authentication-mode is
         third-part-server-authentication";
     }
     identity exemption-authentication {
       base authentication-mode;
       description
       "If authentication-mode is
       exemption-authentication";
     }
     identity sso-authentication {
       base authentication-mode;
       description
       "If authentication-mode is
       sso-authentication";
     }
     identity alarm-type {
       description
         "Base identity for detectable alarm types";
     }
     identity memory-alarm {
       base alarm-type;
       description
         "A memory alarm is alerted";
     }
     identity cpu-alarm {
       base alarm-type;
       description
         "A cpu alarm is alerted";
     }
     identity disk-alarm {
       base alarm-type;
       description
         "A disk alarm is alerted";
     }
     identity hardware-alarm {
       base alarm-type;
       description
         "A hardware alarm is alerted";
     }
     identity interface-alarm {
       base alarm-type;
       description
         "An interface alarm is alerted";
     }
```

```
     identity flood-type {
       description
         "Base identity for detectable flood types";
     }
     identity syn-flood {
       base flood-type;
       description
         "A SYN flood is detected";
     }
     identity ack-flood {
       base flood-type;
       description
         "An ACK flood is detected";
     }
     identity syn-ack-flood {
       base flood-type;
       description
         "An SYN-ACK flood is detected";
     }
     identity fin-rst-flood {
       base flood-type;
       description
         "A FIN-RST flood is detected";
     }
     identity tcp-con-flood {
       base flood-type;
       description
         "A TCP connection flood is detected";
     }
     identity udp-flood {
       base flood-type;
       description
         "A UDP flood is detected";
     }
     identity icmp-flood {
       base flood-type;
       description
         "An ICMP flood is detected";
     }
     identity https-flood {
       base flood-type;
       description
         "A HTTPS flood is detected";
     }
     identity http-flood {
       base flood-type;
       description
         "A HTTP flood is detected";
```

```
     }
     identity dns-reply-flood {
       base flood-type;
       description
         "A DNS reply flood is detected";
     }
     identity dns-query-flood {
       base flood-type;
       description
         "A DNS query flood is detected";
     }
     identity sip-flood {
       base flood-type;
       description
         "A SIP flood is detected";
     }
     identity attack-type {
       description
         "The root ID of attack based notification
         in the notification taxonomy";
     }
     identity system-attack-type {
       base attack-type;
       description
         "This ID is intended to be used
         in the context of system events";
     }
     identity nsf-attack-type {
       base attack-type;
       description
         "This ID is intended to be used in the context of nsf event";
     }
     identity botnet-attack-type {
       base nsf-attack-type;
       description
         "This is a ID stub limited to indicating
         that this attack type is botnet.
         The usual semantic/taxonomy is missing
         and name is used.";
     }
     identity virus-type {
       base nsf-attack-type;
       description
         "The type of virus. Can be multiple types at once. This attack
          type is associated with a detected system-log virus-attack";
     }
     identity trojan {
       base virus-type;
```

```
       description
         "The detected virus type is trojan";
     }
     identity worm {
       base virus-type;
       description
         "The detected virus type is worm";
     }
     identity macro {
       base virus-type;
       description
         "The detected virus type is macro";
     }
     identity intrusion-attack-type {
       base nsf-attack-type;
       description
         "The attack type is associatied with
         a detectedsystem-log intrusion";
     }
     identity brute-force {
       base intrusion-attack-type;
       description
         "The intrusion type is brute-force";
     }
     identity buffer-overflow {
       base intrusion-attack-type;
       description
         "The intrusion type is buffer-overflow";
     }
     identity web-attack-type {
       base nsf-attack-type;
       description
         "The attack type associated with
         a detected system-log web-attack";
     }
     identity command-injection {
       base web-attack-type;
       description
         "The detected web attack type is command injection";
     }
     identity xss {
       base web-attack-type;
       description
         "The detected web attack type is XSS";
     }
     identity csrf {
       base web-attack-type;
       description
```

```
         "The detected web attack type is CSRF";
     }
     identity ddos-attack-type {
       base nsf-attack-type;
       description
         "The attack type is associated with a detected nsf-log event";
     }

     identity req-method {
       description
         "A set of request types (if applicable).
          For instance, PUT or GET in HTTP";
     }
     identity put-req {
       base req-method;
       description
         "The detected request type is PUT";
     }
     identity get-req {
       base req-method;
       description
         "The detected request type is GET";
     }

     identity filter-type {
       description
         "The type of filter used to detect, for example,
          a web-attack. Can be applicable to more than
          web-attacks. Can be more than one type.";
     }
     identity whitelist {
       base filter-type;
       description
         "The applied filter type is whitelist";
     }
     identity blacklist {
       base filter-type;
       description
         "The applied filter type is blacklist";
     }
     identity user-defined {
       base filter-type;
       description
         "The applied filter type is user-defined";
     }
     identity balicious-category {
       base filter-type;
       description
```

```
          "The applied filter is balicious category";
      }
      identity unknown-filter {
        base filter-type;
        description
          "The applied filter is unknown";
      }

      identity access-mode {
        description
          "TBD";
      }
      identity ppp {
        base access-mode;
        description
          "Access-mode : ppp";
      }
      identity svn {
        base access-mode;
        description
          "Access-mode : svn";
      }
      identity local {
        base access-mode;
        description
          "Access-mode : local";
      }

      grouping protocol {
        description
          "A set of protocols";
        container protocol {
          description
          "Protocol types:
           TCP, UDP, ICMP, ICMPv6, IP, HTTP, FTP and etc.";
          leaf tcp {
            type boolean;
            description
              "TCP protocol type.";
          }
          leaf udp {
            type boolean;
            description
              "UDP protocol type.";
          }
          leaf icmp {
            type boolean;
            description
```

```
             "ICMP protocol type.";
         }
         leaf icmpv6 {
           type boolean;
           description
             "ICMPv6 protocol type.";
         }
         leaf ip {
           type boolean;
           description
             "IP protocol type.";
         }
         leaf http {
           type boolean;
           description
             "HTTP protocol type.";
         }
         leaf ftp {
           type boolean;
           description
             "ftp protocol type.";
         }
       }
     }
     grouping common-notification-content {
       description
       "TBD";
       leaf message {
         type string;
         mandatory true;
         description
           "This is a freetext annotation of
           monitoring notification content";
       }
       leaf time-stamp {
         type yang:date-and-time;
         mandatory true;
         description
           "Indicates the time of message generation";
       }
       leaf severity {
         type severity;
         mandatory true;
         description
           "The severity of the alarm such
           asvcritical, high, middle, low.";
       }
     }
```

```
     grouping common-nsf-notification-content {
       description
       "TBD";
       leaf vendor-name {
         type string;
         description
           "The name of the NSF vendor";
       }
       leaf nsf-name {
         type string;
         description
           "The name (or IP) of the NSF
           generating the message";
       }
     }
     grouping i2nsf-system-alarm-type-content {
       description
         "A set of system alarm type contents";
       leaf usage {
         type uint8;
         description
           "specifies the amount of usage";
       }
       leaf threshold {
         type uint8;
         description
           "The threshold triggering the alarm or the event";
       }
     }
     grouping i2nsf-system-event-type-content {
       description
         "System event metadata associated with system events caused
          by user activity.";
       leaf group {
         type string;
         mandatory true;
         description
           "Group to which a user belongs.";
       }
       leaf login-ip-addr {
         type inet:ipv4-address;
         mandatory true;
         description
           "Login IP address of a user.";
       }
       leaf authentication {
         type identityref {
           base authentication-mode;
```

```
         }
         description
           "TBD";
       }
     }

     grouping i2nsf-nsf-event-type-content {
       description
         "A set of common IPv4-related NSF event
         content elements";
       leaf event-message {
         type string;
         description
           "The message for nsf events";
       }
       leaf src-ip {
         type inet:ipv4-address;
         description
           "The source IP address of the packet";
       }
       leaf dst-ip {
         type inet:ipv4-address;
         description
           "The destination IP address of the packet";
       }
       leaf src-port {
         type inet:port-number;
         description
           "The source port of the packet";
       }
       leaf dst-port {
         type inet:port-number;
         description
           "The destination port of the packet";
       }
       leaf src-zone {
         type string;
         description
           "The source security zone of the packet";
       }
       leaf dst-zone {
         type string;
         description
           "The destination security zone of the packet";
       }
       leaf rule-id {
         type uint8;
         mandatory true;
```

```
         description
           "The ID of the rule being triggered";
       }
       leaf rule-name {
         type string;
         mandatory true;
         description
           "The name of the rule being triggered";
       }
       leaf profile {
         type string;
         description
           "Security profile that traffic matches.";
       }
       leaf raw-info {
         type string;
         description
           "The information describing the packet
           triggering the event.";
       }
     }
     grouping traffic-rates {
       description
         "A set of traffic rates
         for statistics data";
       leaf total-traffic {
         type uint32;
         description
           "Total traffic";
       }
       leaf in-traffic-ave-rate {
         type uint32;
         description
           "Inbound traffic average rate in pps";
       }
       leaf in-traffic-peak-rate {
         type uint32;
         description
           "Inbound traffic peak rate in pps";
       }
       leaf in-traffic-ave-speed {
         type uint32;
         description
           "Inbound traffic average speed in bps";
       }
       leaf in-traffic-peak-speed {
         type uint32;
         description
```

```
            "Inbound traffic peak speed in bps";
        }
        leaf out-traffic-ave-rate {
          type uint32;
          description
            "Outbound traffic average rate in pps";
        }
        leaf out-traffic-peak-rate {
          type uint32;
          description
            "Outbound traffic peak rate in pps";
        }
        leaf out-traffic-ave-speed {
          type uint32;
          description
            "Outbound traffic average speed in bps";
        }
        leaf out-traffic-peak-speed {
          type uint32;
          description
            "Outbound traffic peak speed in bps";
        }
      }
      grouping i2nsf-system-counter-type-content{
        description
          "A set of system counter type contents";
        leaf interface-name {
          type string;
          description
            "Network interface name configured in NSF";
        }
        leaf in-total-traffic-pkts {
          type uint32;
          description
            "Total inbound packets";
        }
        leaf out-total-traffic-pkts {
          type uint32;
          description
            "Total outbound packets";
        }
        leaf in-total-traffic-bytes {
          type uint32;
          description
            "Total inbound bytes";
        }
        leaf out-total-traffic-bytes {
          type uint32;
```

```
         description
           "Total outbound bytes";
       }
       leaf in-drop-traffic-pkts {
         type uint32;
         description
           "Total inbound drop packets";
       }
       leaf out-drop-traffic-pkts {
         type uint32;
         description
           "Total outbound drop packets";
       }
       leaf in-drop-traffic-bytes {
         type uint32;
         description
           "Total inbound drop bytes";
       }
       leaf out-drop-traffic-bytes {
         type uint32;
         description
           "Total outbound drop bytes";
       }
       uses traffic-rates;
     }
     grouping i2nsf-nsf-counters-type-content{
       description
         "A set of nsf counters type contents";
       leaf src-ip {
         type inet:ipv4-address;
         description
           "The source IP address of the packet";
       }
       leaf dst-ip {
         type inet:ipv4-address;
         description
           "The destination IP address of the packet";
       }
       leaf src-port {
         type inet:port-number;
         description
           "The source port of the packet";
       }
       leaf dst-port {
         type inet:port-number;
         description
           "The destination port of the packet";
       }
```

```
       leaf src-zone {
         type string;
         description
           "The source security zone of the packet";
       }
       leaf dst-zone {
         type string;
         description
           "The destination security zone of the packet";
       }
       leaf src-region {
         type string;
         description
           "Source region of the traffic";
       }
       leaf dst-region{
         type string;
         description
           "Destination region of the traffic";
       }
       leaf policy-id {
         type uint8;
         description
           "The ID of the policy being triggered";
       }
       leaf policy-name {
         type string;
         description
           "The name of the policy being triggered";
       }
       leaf src-user{
         type string;
         description
           "User who generates traffic";
       }
       uses protocol;
       uses traffic-rates;
     }

     notification system-detection-alarm {
       description
         "TBD";
       leaf alarm-catagory {
         type identityref {
           base alarm-type;
         }
         description
           "TBD";
```

```
       }
       uses i2nsf-system-alarm-type-content;
       uses common-notification-content;
     }
     notification system-detection-access-violation {
       description
         "This notification is sent, when a security-sensitive
          authentication action fails.";
       uses i2nsf-system-event-type-content;
       uses common-notification-content;
     }
     notification system-detection-config-change {
       description
         "This notification is sent,
         when an unauthorized confinguration
         change action is detected.";
       uses i2nsf-system-event-type-content;
       uses common-notification-content;
     }
     notification nsf-detection-flood {
       description
         "This notification is sent,
         when a specific flood type is detected";
       uses i2nsf-nsf-event-type-content;
       leaf flood-catagory {
         type identityref {
           base flood-type;
         }
         description
           "TBD";
       }
       leaf start-time {
         type yang:date-and-time;
         mandatory true;
         description
           "The time stamp indicating when the attack started";
       }
       leaf end-time {
         type yang:date-and-time;
         mandatory true;
         description
           "The time stamp indicating when the attack ended";
       }
       leaf attack-rate {
         type uint32;
         description
           "The PPS rate of attack traffic";
       }
```

```
       leaf attack-speed {
         type uint32;
         description
           "The BPS speed of attack traffic";
       }
       uses common-nsf-notification-content;
       uses common-notification-content;
     }
     notification nsf-detection-session-table {
       description
         "This notification is sent, when an a session table event
          is deteced";
       leaf current-session {
         type uint8;
         description
           "The number of concurrent sessions";
       }
       leaf maximum-session {
         type uint8;
         description
           "The maximum number of sessions that the session
            table can support";
       }
       leaf threshold {
         type uint8;
         description
           "The threshold triggering the event";
       }
       leaf table-indentifier {
         type string;
         description
           "The number of session table exceeded the threshold";
       }
       uses common-nsf-notification-content;
       uses common-notification-content;
     }
     notification nsf-detection-virus {
       description
         "This notification is sent, when a virus is detected";
       uses i2nsf-nsf-event-type-content;
       leaf virus {
         type identityref {
           base virus-type;
         }
         description
           "TBD";
       }
       leaf virus-name {
```

```
          type string;
          description
          "The name of the detected virus";
        }
        leaf file-type {
          type string;
          description
            "The type of file virus code is found in (if appicable).";
        }
        leaf file-name {
          type string;
          description
             "The name of file virus code is found in (if appicable).";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification nsf-detection-intrusion {
        description
          "This notification is send, when an intrusion event
           is detected.";
        uses i2nsf-nsf-event-type-content;
        uses protocol;
        leaf intrusion {
          type identityref {
            base intrusion-attack-type;
          }
          description
            "TBD";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification nsf-detection-botnet {
        description
          "This notification is send, when a botnet event is
           detected";
        uses i2nsf-nsf-event-type-content;
        leaf attack-type {
          type identityref {
            base botnet-attack-type;
          }
          description
            "TBD";
        }
        uses protocol;
        leaf botnet-name {
          type string;
```

```
          description
            "The name of the detected botnet";
        }
        leaf role {
          type string;
          description
            "The role of the communicating
             parties within the botnet";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification nsf-detection-web-attack {
        description
          "This notification is send, when an attack event is
           detected";
        uses i2nsf-nsf-event-type-content;
        leaf web-attack {
          type identityref {
            base web-attack-type;
          }
          description
            "TBD";
        }
        uses protocol;
        leaf request {
          type identityref {
            base req-method;
          }
          description
            "TBD";
        }
        leaf req-uri {
          type string;
          description
          "Requested URI";
        }
        leaf uri-category {
          type string;
          description
          "Matched URI category";
        }
        leaf-list filter {
          type identityref {
            base filter-type;
          }
          description
            "TBD";
```

```
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification system-log-access-event {
        description
          "The notification is send, if there is
          a new system log entry about
           a system access event";
        leaf login-ip {
          type inet:ipv4-address;
          mandatory true;
          description
            "Login IP address of a user";
        }
        leaf administrator {
          type string;
          description
            "Administrator that maintains the device";
        }
        leaf login-mode {
          type login-mode;
          description
            "Specifies the administrator log-in mode";
        }
        leaf operation-type {
          type operation-type;
          description
            "The operation type that the administrator execute";
        }
        leaf result {
          type string;
          description
            "Command execution result";
        }
        leaf content {
          type string;
          description
            "The Operation performed by an administrator after login";
        }
        uses common-nsf-notification-content;
      }
      notification system-log-res-util-report {
        description
          "This notification is send, if there is
          a new log entry representing ressource
           utiliztation updates.";
        leaf system-status {
```

```
          type string;
          description
            "The current systems
            running status";
       }
       leaf cpu-usage {
         type uint8;
         description
           "Specifies the relative amount of
           cpu usage wrt plattform ressources";
       }
       leaf memory-usage {
          type uint8;
          description
            "Specifies the amount of memory usage";
       }
       leaf disk-usage {
         type uint8;
         description
           "Specifies the amount of disk usage";
       }
       leaf disk-left {
          type uint8;
          description
            "Specifies the amount of disk left";
       }
       leaf session-num {
         type uint8;
         description
           "The total number of sessions";
       }
       leaf process-num {
         type uint8;
         description
           "The total number of process";
       }
       leaf in-traffic-rate {
         type uint32;
         description
           "The total inbound traffic rate in pps";
       }
       leaf out-traffic-rate {
         type uint32;
         description
            "The total outbount traffic rate in pps";
       }
       leaf in-traffic-speed {
         type uint32;
```

```
         description
           "The total inbound traffic speed in bps";
       }
       leaf out-traffic-speed {
         type uint32;
         description
           "The total outbound traffic speed in bps";
       }
       uses common-nsf-notification-content;
     }
     notification system-log-user-activity-event {
       description
         "This notification is send, if there is
         a new user activity log entry";
       leaf user {
         type string;
         mandatory true;
         description
           "Name of a user";
       }
       leaf group {
         type string;
         mandatory true;
         description
           "Group to which a user belongs.";
       }
       leaf login-ip {
         type inet:ipv4-address;
         mandatory true;
         description
           "Login IP address of a user.";
       }
       leaf authentication {
         type identityref {
           base authentication-mode;
         }
         description
           "TBD";
       }
       leaf accese {
         type identityref {
           base access-mode;
         }
         description
           "TBD";
       }
       leaf online-duration {
         type string;
```

```
         description
           "Online duration";
       }
       leaf logout-duration {
         type string;
         description
           "Lockout duration";
       }
       leaf addtional-info {
         type string;
         description
           "User activities. e.g., Successful
           User Login, Failed Login attempts,
           User Logout, Successful User
           Password Change, Failed User
           Password Change, User Lockout,
           User Unlocking, Unknown";
       }
       uses common-nsf-notification-content;
     }
     notification nsf-log-ddos {
       description
         "This notification is send, if there is
         a new DDoS event log entry in the nsf log";
       leaf attack-type {
         type identityref {
           base ddos-attack-type;
         }
         description
           "TBD";
       }
       leaf attack-ave-rate {
         type uint32;
         description
           "The ave PPS of attack traffic";
       }
       leaf attack-ave-speed {
         type uint32;
         description
           "the ave bps of attack traffic";
       }
       leaf attack-pkt-num {
         type uint32;
         description
           "the number of attack packets";
       }
       leaf attack-src-ip {
         type inet:ipv4-address;
```

```
          description
            "The source IP addresses of attack
             traffics. If there are a large
             amount of IP addresses, then
             pick a certain number of resources
             according to different rules.";
        }
        leaf action {
          type log-action;
          description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
        }
        leaf os {
          type string;
          description
            "simple os information";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification nsf-log-virus {
        description
          "This notification is send, If there is
          a new virus event log enry in the nsf log";
        leaf attack-type {
          type identityref {
            base virus-type;
          }
          description
            "TBD";
        }
        leaf action {
          type log-action;
          description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
        }
        leaf os{
          type string;
          description
            "simple os information";
        }
        leaf time {
          type yang:date-and-time;
          mandatory true;
```

```
          description
            "Indicate the time when the message is generated";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification nsf-log-intrusion {
        description
          "This notification is send, if there is
          a new intrusion event log entry in the nsf log";
        leaf attack-type {
          type identityref {
            base intrusion-attack-type;
          }
          description
            "TBD";
        }
        leaf action {
          type log-action;
          description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
        }
        leaf time {
          type yang:date-and-time;
          mandatory true;
          description
            "Indicate the time when the message is generated";
        }
        leaf attack-rate {
          type uint32;
          description
            "The PPS of attack traffic";
        }
        leaf attack-speed {
          type uint32;
          description
            "the bps of attack traffic";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      notification nsf-log-botnet {
        description
          "This noticiation is send, if there is
          a new botnet event log in the nsf log";
        leaf attack-type {
```

```
         type identityref {
           base botnet-attack-type;
         }
         description
           "TBD";
       }
       leaf action {
         type log-action;
         description
           "Action type: allow, alert,
           block, discard, declare,
           block-ip, block-service";
       }
       leaf botnet-pkt-num{
         type uint8;
         description
           "The number of the packets sent to
           or from the detected botnet";
       }
       leaf os{
         type string;
         description
           "simple os information";
       }
       uses common-nsf-notification-content;
       uses common-notification-content;
     }
     notification nsf-log-dpi {
       description
         "This notification is send, if there is
         a new dpi event in the nsf log";
       leaf attack-type {
         type dpi-type;
         description
           "The type of the dpi";
       }
       leaf src-ip {
         type inet:ipv4-address;
         description
           "The source IP address of the packet";
       }
       leaf dst-ip {
         type inet:ipv4-address;
         description
           "The destination IP address of the packet";
       }
       leaf src-port {
         type inet:port-number;
```

```
          description
            "The source port of the packet";
        }
        leaf dst-port {
          type inet:port-number;
          description
            "The destination port of the packet";
        }
        leaf src-zone {
          type string;
          description
            "The source security zone of the packet";
        }
        leaf dst-zone {
          type string;
          description
            "The destination security zone of the packet";
        }
        leaf src-region {
          type string;
          description
            "Source region of the traffic";
        }
        leaf dst-region{
          type string;
          description
            "Destination region of the traffic";
        }
        leaf policy-id {
          type uint8;
          mandatory true;
          description
            "The ID of the policy being triggered";
        }
        leaf policy-name {
          type string;
          mandatory true;
          description
            "The name of the policy being triggered";
        }
        leaf src-user{
          type string;
          description
            "User who generates traffic";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
```

```
     notification nsf-log-vuln-scan {
       description
         "This notification is send, if there is
         a new vulnerability-scan report in the nsf log";
       leaf vulnerability-id {
         type uint8;
         description
           "The vulnerability id";
       }
       leaf victim-ip {
         type inet:ipv4-address;
         description
           "IP address of the victim host which has vulnerabilities";
       }
       uses protocol;
       leaf port-num {
         type inet:port-number;
           description
             "The port number";
       }
       leaf level {
         type severity;
         description
           "The vulnerability severity";
       }
       leaf os {
         type string;
         description
           "simple os information";
       }
       leaf addtional-info {
         type string;
         description
           "TBD";
       }
       uses common-nsf-notification-content;
       uses common-notification-content;
     }
     notification nsf-log-web-attack {
       description
         "This notificatio is send, if there is
         a new web-attack event in the nsf log";
       leaf attack-type {
         type identityref {
           base web-attack-type;
         }
         description
           "TBD";
```

```
        }
        leaf rsp-code {
          type string;
          description
            "Response code";
        }
        leaf req-clientapp {
          type string;
          description
            "The client application";
        }
        leaf req-cookies {
          type string;
          description
            "Cookies";
        }
        leaf req-host {
          type string;
          description
            "The domain name of the requested host";
        }
        leaf raw-info {
          type string;
          description
            "The information describing
            the packet triggering the event.";
        }
        uses common-nsf-notification-content;
        uses common-notification-content;
      }
      container counters {
        description
          "This is probably better covered by an import
          as this will not be notifications.
          Counter are not very suitable as telemetry, maybe
          via periodic subscriptions, which would still
          violate principle of least surprise.";
        container system-interface {
          description
            "The system counter type is interface counter";
          uses i2nsf-system-counter-type-content;
        }
        container nsf-firewall {
          description
            "The nsf counter type is firewall counter";
          uses i2nsf-nsf-counters-type-content;
          container diretcions {
            description
```

```
            "Inbound or Outbound";
          leaf in-interface {
            type boolean;
            description
              "If the bound is inbound";
          }
          leaf out-interface {
            type boolean;
            description
              "If the bound is outbound";
          }
        }
      }
      container nsf-policy-hits {
        description
          "The counters of policy hit";
        uses i2nsf-nsf-counters-type-content;
        leaf hit-times {
          type uint32;
          description
            "The hit times for policy";
        }
      }
    }
  }
  <CODE ENDS>
```

Figure 2: Data Model of Monitoring

6. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This document has greatly benefited from inputs by Daeyoung Hyun.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

7.2. Informative References

[i2nsf-framework]
Lopez,, D., Lopez,, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-10 (work in progress), November 2017.

[i2nsf-monitoring-im]
Xia,, L., Zhang,, D., Wu, Y., Kumar, R., Lohiya, A., and H. Birkholz, "An Information Model for the Monitoring of Network Security Functions (NSF)", draft-zhang-i2nsf-info-model-monitoring-05 (work in progress), October 2017.

[i2nsf-terminology]
Hares,, S., Strassner,, J., Lopez,, D., Xia,, L., and H. Birkholz,, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), October 2017.

[i2rs-rib-data-model]
Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-10 (work in progress), February 2018.

Appendix A. draft-hong-i2nsf-nsf-monitoring-data-model-02

The following changes are made from draft-hong-i2nsf-nsf-monitoring-data-model-01:

1. The YANG data model is defined in more detail based on the information model for monitoring NSFs.

2. Some of descriptions for YANG data model are revised.

3. Typos and grammatical errors are corrected.

Authors' Addresses

Dongjin Hong
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 7630 5473
EMail: dong.jin@skku.edu

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Jinyong Tim Kim
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 8273 0930
EMail: timkim@skku.edu

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
EMail: shares@ndzh.com

Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu
China

EMail: Frank.xialiang@huawei.com

Henk Birkholz
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

EMail: henk.birkholz@sit.fraunhofer.de

Network Working Group S. Hyun
Internet-Draft J. Jeong
Intended status: Standards Track Sungkyunkwan University
Expires: September 6, 2018 J. Park
ETRI
S. Hares
Huawei
March 5, 2018

Service Function Chaining-Enabled I2NSF Architecture
draft-hyun-i2nsf-nsf-triggered-steering-05

Abstract

This document describes an architecture of the I2NSF framework using security function chaining for security policy enforcement. Security function chaining enables composite inspection of network traffic by steering the traffic through multiple types of network security functions according to the information model for NSFs capabilities in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve security function chaining. It also describes representative use cases to address major benefits from the proposed architecture.

Table of Contents

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various security functions cooperatively analyze network traffic [RFC7498][i2nsf-problem][nsf-capability-im]. In addition, depending on the characteristics of network traffic and their suspiciousness level, the different types of network traffic need to be analyzed through the different sets of security functions.

In order to meet such requirements, besides security policy rules for individual security functions, we need an additional policy about service function chaining (SFC) for network security which determines a set of security functions through which network traffic packets should pass for inspection. In addition, [nsf-capability-im] proposes an information model for NSFs capabilities that enables a security function to trigger further inspection by executing additional security functions based on its own analysis results [RFC8329]. However, the current design of the I2NSF framework does not consider network traffic steering fully in order to enable such chaining between security functions.

In this document, we propose an architecture that integrates additional components from Service Function Chaining (SFC) into the I2NSF framework to support security function chaining. We extend the security controller’s functionalities such that it can interpret a high-level policy of security function chaining into a low-level policy and manage them. It also keeps the track of the available service function (SF) instances for security functions and their information (e.g., network information and workload), and makes a decision on which SF instances to use for a given security function chain/path. Based on the forwarding information provided by the security controller, the service function forwarder (SFF) performs network traffic steering through various required security functions. A classifier is deployed for the enforcement of SFC policies given by the security controller. It performs traffic classification based on the polices so that the traffic passes through the required security function chain/path by the SFF.

2. Objective

o Policy configuration for security function chaining: SFC-enabled I2NSF architecture allows policy configuration and management of security function chaining. Based on the chaining policy, relevant network traffic can be analyzed through various security functions in a composite, cooperative manner.

o Network traffic steering for security function chaining: SFC-enabled I2NSF architecture allows network traffic to be steered through multiple required security functions based on the SFC policy. Moreover, the I2NSF information model for NSFs capabilities [nsf-capability-im] requires a security function to call another security function for further inspection based on its own inspection result. To meet this requirement, SFC-enabled I2NSF architecture also enables traffic forwarding from one security function to another security function.

o Load balancing over security function instances: SFC-enabled I2NSF architecture provides load balancing of incoming traffic over available security function instances by leveraging the flexible traffic steering mechanism. For this objective, it also performs dynamic instantiation of a security function when there are an excessive amount of requests for that security function.

3. Terminology

This document uses the following terminology described in [RFC7665], [RFC7665][i2nsf-terminology][ONF-SFC-Architecture].

o Service Function/Security Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers) [RFC7665]. In this document, SF is used to represent both Service Function and Security Function. Sample Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.

o Classifier: An element that performs Classification. It uses a given policy from SFC Policy Manager.

o Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification [RFC7665].

o Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, an SFF is responsible for delivering traffic to a classifier when needed and supported, transporting traffic to another SFF (in the same or the different type of overlay), and terminating the Service Function Path (SFP) [RFC7665].

o Service Function Path (SFP): The service function path is a constrained specification of where packets assigned to a certain service function path must be forwarded. While it may be so constrained as to identify the exact locations for packet processing, it can also be less specific for such locations [RFC7665].

- o SFC Policy Manager: It is responsible for translating a high-level policy into a low-level policy, and performing the configuration for SFC-aware nodes, passing the translated policy and configuration to SFC-aware nodes, and maintaining a stabilized network.

- o SFC Catalog Manager: It is responsible for keeping the track of the information of available SF instances. For example, the information includes the supported transport protocols, IP addresses, and locations for the SF instances.

- o Control Nodes: It collectively refer to SFC Policy Manager, SFC Catalog Manager, SFF, and Classifier.

- o Service Path Identifier (SPI): It identifies a service path. The classifier MUST use this identifier for path selection and the Control Nodes MUST use this identifier to find the next hop [RFC8300].

- o Service Index (SI): It provides a location within the service path. SI MUST be decremented by service functions or proxy nodes after performing the required services [RFC8300].

- o Network Service Header (NSH): The header is used to carry SFC related information. Basically, SPI and SI should be conveyed to the Control Nodes of SFC via this header.

- o SF Forwarding Table: SFC Policy Manager maintains this table. It contains all the forwarding information on SFC-enabled I2NSF architecture. Each entry includes SFF identifier, SPI, SI, and next hop information. For example, an entry ("SFF: 1", "SPI: 1", "SI: 1", "IP: 192.168.xx.xx") is interpreted as follows: "SFF 1" should forword the traffic containing "SPI 1" and "SI 1" to "IP=192.168.xx.xx".

4. Architecture

This section describes an SFC-enabled I2NSF architecture and the basic operations of service chaining. It also includes details about each component of the architecture.

Figure 1 describes the components of SFC-enabled I2NSF architecture. Our architecture is designed to support a composite inspection of traffic packets in transit. According to the inspection result of each SF, the traffic packets could be steered to another SF for futher detailed analysis. It is also possible to reflect a high-level SFC-related policy and a configuration from I2NSF Client on the components of the original I2NSF framwork. Moreover, the proposed architecture provides load balancing, auto supplementary SF generation, and the elimination of unused SFs. In order to achieve these design purposes, we integrate several components to the original I2NSF framwork. In the following sections, we explain the details of each component.

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | I2NSF User                                                        |
   |              +-+-+-+-+-+-+-+-+-+                                    |
   |              |I2NSF User      |                                    |
   |              |                |                                    |
   |              +-+-+-+^+-+-+-+-+                                    |
   |                     |                                             |
   |                     |                                             |
   +-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                         | Consumer-Facing Interface
                         |
   +-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Security Management System                                        |
   |                     |                                             |
   |       +-+-+-+-+-+-+-v-+-+-+-+-+                                   |
   |       |Security Controller    |                                   |
   |       | +-+-+-+-+  +-+-+-+-+  | Registration                      |
   |       | |SFC    |  |SFC    |  |   Interface +-+-+-+-++-+-+-+-+    |
   |       | |Policy |  |Catalog|  |<------------>| Developer's    |    |
   |       | |Manager|  |Manager|  |             | Mgnt System    |    |
   |       | +-+-+-+-+  +-+-+-+-+  |             +-+-+-+-++-+-+-+-+    |
   |       +-+-+-+-+-+-^-+-+-+-+-+-+                                   |
   +-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                       | NSF-Facing Interface
                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Security Network   |                                               |
   |       +----------------------------------------------+            |
   |       |                    |                         |            |
   |       |              +-+-+-v-+-+-+           +-+-+-+-+v+-+-+      |
   | +-+-+-v-++-+         |  +-+-+-+  |           |    +-+-+-+  |      |
   | |          |         |  |SFF1 |<-|----------------->| SF1 |  |      |
   | |Classifier|<------> |  +-+-+-+< |           |    +-+-+-+  |      |
   | |          |         |         \|            |    +-+-+-+  |      |
   | +-+-+-+-++-+         |  +-+-+-+  \----------------->| SF2 |  |      |
   |                      |  |SFF2 |<-|---------------/ +-+-+-+  |      |
   |                      |  +-+-+-+<-|-------------\   +-+-+-+  |      |
   |                      +-+-+-+-+-+-+              \->| SF3 |  |      |
   |                                                |    +-+-+-+  |      |
   |                                                +-+-+-+-+-+-+-+      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: SFC-enabled I2NSF

4.1. SFC Policy Manager

SFC Policy Manager is a core component in our system. It is responsible for the following two things: (1) Interpreting a high-level SFC policy (or configuration) into a low-level SFC policy (or configuration), which is given by I2NSF Client, and delivering the interpreted policy to Classifiers for security function chaining. (2) Generating an SF forwarding table and distributing the fowarding information to SFF(s) by consulting with SFC Catalog Manager. As Figure 1 describes, SFC Policy Manager performs these additional functionalities through Consumer-Facing Interface and NSF-Facing Interface.

Given a high-level SFC policy/configuration from I2NSF Client via Consumer-Facing Interface, SFC Policy Manager interprets it into a low-level policy/configuration comprehensible to Classifier(s), and then delivers the resulting low-level policy to them. Moreover, SFC Policy Manager possibly generates new policies for the flexible change of traffic steering to rapidly react to the current status of SFs. For instance, it could generate new rules to forward all subsequent packets to "Firewall Instance 2" instead of "Firewall Instance 1" in the case where "Firewall Instance 1" is under congestion.

SFC Policy Manager gets information about SFs from SFC Catalog Manager to generate SF forwarding table. In the table generation process, SFC Policy Manager considers various criteria such as SFC policies, SF load status, SF physical location, and supported transport protocols. An entry of the SF forwarding table consists of SFF Identifier, SFP, SI, and next hop information. The examples of next hop information includes the IP address and supported transport protocols (e.g., VxLAN and GRE). These forwarding table updates are distributed to SFFs with either push or pull methods.

4.2. SFC Catalog Manager

In Figure 1, SFC Catalog Manager is a component integrated into Security Controller. It is responsible for the following three things: (1) Maintaining the information of every available SF instance such as IP address, supported transport protocol, service name, and load status. (2) Responding to the queries of available SF instances from SFC Policy Manager so as to help to generate a forwarding table entry relevant to a given SFP. (3) Requesting Developer’s Management System to dynamically instantiate supplementary SF instances to avoid service congestion or the elimination of an existing SF instance to avoid resource waste.

Whenever a new SF instance is registered, Developer's Management System passes the information of the registered SF instance to SFC Catalog Manager, so SFC Catalog Manager maintains a list of the information of every available SF instance. Once receiving a query of a certain SFP from SFC Policy Manager, SFC Catalog Manager searches for all the available SF instances applicable for that SFP and then returns the search result to SFC Policy Manager.

In our system, each SF instance periodically reports its load status to SFC Catalog Manager. Based on such reports, SFC Catalog Manager updates the information of the SF instances and manages the pool of SF instances by requesting Developer's Management System for the additional instantiation or elimination of the SF instances. Consequently, SFC Catalog Manager enables efficient resource utilization by avoiding congestion and resource waste.

4.3. Developer's Management System

We extend Developer's Management System for additional functionalities as follows. As mentioned above, the SFC Catalog Manager requests the Developer's Management System to create additional SF instances when the existing instances of that service function are congested. On the other hand, when there are an excessive number of instances for a certain service function, the SFC Policy Manager requests the Developer's Management System to eliminate some of the SF instances. As a response to such requests, the Developer's Management System creates and/or removes SF instances. Once it creates a new SF instance or removes an existing SF instance, the changes must be notified to the SFC Catalog Manager.

4.4. Classifier

Classifier is a logical component that may exist as a standalone component or a submodule of another component. In our system, the initial classifier is typically located at an entry point like a border router of the network domain, and performs the initial classification of all incoming packets according to the SFC policies, which are given by SFC policy manager. The classification means determining the SFP through which a given packet should pass. Once the SFP is decided, the classifier constructs an NSH that specifies the corresponding SPI and SI, and attaches it to the packet. The packet will then be forwarded through the determined SFP on the basis of the NSH information.

4.5. Service Function Forwarder (SFF)

It is responsible for the following two functionalities: (1) Forwarding the packets to the next SFF/SF. (2) Handling re-classification request from SF.

An SFF basically takes forwarding functionality, so it needs to find the next SF/SFF for the incoming traffic. It will search its forwarding table to find the next hop information that corresponds to the given traffic. In the case where the SFF finds a target entry on its forwarding table, it just forwards the traffic to the next SF/SFF specified in the next hop information. If an SFF does not have an entry for a given packet, it will request the next hop information to SFC Policy Manager with SFF identifier, SPI, and SI information. The SFC Policy Manager will respond to the SFF with next hop information, and then the SFF updates its forwarding table with the response, forwarding the traffic to the next hop.

Sometimes an SF may want to forward a packet, which is highly suspicious, to another SF for futher security inspection. This is referred to as advanced security action in I2NSF. In this situation, if the next SF may not be the one on the current SFP of the packet, re-classification is required to change the SFP of the packet. If the current SF is capable of re-classifying the packet by itself, the SF updates the SPI field in the NSH in the packet to serve the advanced secuity action. Otherwise, if the classifier exists as a standalone, the SF appends the inspection result of the packet to the MetaData field of the NSH and delivers it to the source SFF. The attached MetaData includes a re-classification request to change the SFP of the packet to another SFP for stronger inspection. When the SFF receives the traffic requiring re-classification, it forwards the traffic to the Classifier where re-classification will be eventually performed.

SFC defines Rendered Service Path (RSP), which represents the sequence of actual visits by a packet to SFFs and SFs [RFC7665]. If the RSP information of a packet is available, the SFF could check this RSP information to detect whether undesired looping happened on the packet. If the SFF detects looping, it could notify the Security Controller of this looping, and the Security Controller could modify relevant security policy rules to resolve this looping.

5. Use Cases

This section introduces three use cases for the SFC-enabled I2NSF architecture : (1) Dynamic Path Alternation, (2) Enforcing Different SFPs Depending on Trust Levels, and (3) Effective Load Balancing with Dynamic SF Instantiation.

5.1. Dynamic Path Alternation

In SFC-enabled I2NSF architecture, a Classifier determines the initial SFP of incoming traffic according to the SFC policies. The classifier then attaches an NSH specifying the determined SFP of the packets, and they are analyzed through the SFs of the initial SFP. However, SFP is not a static property, so it could be changed dynamically through re-classification. A typical example is for a certain SF in the initial SFP to detect that the traffic is highly suspicious (likely to be malicious). In this case, the traffic needs to take stronger inspection through a different SFP which consists of more sophisticated SFs.

Figure 2 illustrates an example of such dynamic SFP alternation in a DDoS attack scenario. SFP-1 represents the default Service Function Path that the traffic initially follows, and SFP-1 consists of AVC, Firewall, and IDS/IPS. If the IDS/IPS suspects that the traffic is attempting DDoS attacks, it will change the SFP of the traffic from the default to SFP-2 so that the DDoS attack mitigator can execute a proper countermeasure against the attack.

Such SFP alternation is possible in the proposed architecture with re-classification. In Figure 1, to initiate re-classification, the IDS/IPS appends its own inspection result to the MetaData field of NSH and deliver it to the SFF from which it has originally received the traffic. The SFF then forwards the received traffic including the inspection result from the IDS/IPS to Classifier for re-classification. Classifier checks the inspection result and determines the new SFP (SFP-2) associated with the inspection result in the SFC policy, and updates the NSH with the SPI of SFP-2. The traffic is forwarded to the DDoS attack mitigator.

```
                   SFP-1. AVC:Firewall:IDS/IPS
   ------------------------------------------------------------------>
   +-+-+-+-+   +-+-+-+   +-+-+-+-+-+     +-+-+-+-+-+   +-+-+-+-+-+-+-+
   | Source|---| AVC |---| Firewall|-----| IDS/IPS |---| Destination |
   +-+-+-+-+   +-+-+-+   +-+-+-+-+-+     +-+-+-+-+-+   +-+-+-+-+-+-+-+
   ----------------------------------------,                  ,------>
                                            \    +-+-+-+-+   /
                                             \   |  DDoS |  /
                                              \  +-+-+-+-+ /
                                               '----------'
                                     SFP-2. AVC:Firewall:DDoS:IDS/IPS
```

Figure 2: Dynamic SFP Alternation Example

5.2. Enforcing Different SFPs Depending on Trust Levels

Because the traffic coming from a trusted source is highly likely to be harmless, it does not need to be inspected excessively. On the other hand, the traffic coming from an untrusted source requires an in-depth inspection. By applying minimum required security functions to the traffic from a trusted source, it is possible to prevent the unnecessary waste of resources. In addition, we can concentrate more resources on potential malicious traffic. In the SFC-enabled I2NSF architecture, by configuring an SFC Policy to take into account the levels of trust of traffic sources, we can apply different SFPs to the traffic coming from different sources.

Figure 3(a) and Figure 3(b) represent SFPs applicable to traffic from trusted and untrusted sources, respectively. In Figure 3(a), we assume a lightweight IDS/IPS which is configured to perform packet header inspection only. In this scenario, when receiving the traffic from a trusted source, the classifier determines the SFP in Figure 3(a) such that the traffic passes through just a simple analysis by the lightweight IDS/IPS. On the other hand, traffic from an untrusted source passes more thorough examination through the SFP in Figure 3(b) which consists of three different types of SFs.

```
+-+-+-+-+-+          +-+-+-+-+-+          +-+-+-+-+-+-+-+
| Source  |----------->| IDS/IPS |----------->| Destination |
+-+-+-+-+-+          +-+-+-+-+-+          +-+-+-+-+-+-+-+

          (a) Traffic flow of trusted source


+-+-+-+-+-+          +-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+
| Source  |          | Anti-Spoofing |      | Destination |
+-+-+-+-+-+          | function      |      +-+-+-+^+-+-+-+
    |                +-+^+-+-+-+-+-+-+             |
    |                   |         |                |
    |       +-+-+-+-+-+-+ |         |    +-+-+-+-+-+ |
    ------->| Firewall  |--         ---->|   DPI   |--
            +-+-+-+-+-+-+              +-+-+-+-+-+

          (b) Traffic flow of untrusted source
```

Figure 3: Different path allocation depending on source of traffic

5.3. Effective Load Balancing with Dynamic SF Instantiation

In a large-scale network domain, there typically exist a large number of SF instances that provide various security services. It is possible that a specific SF instance experiences an excessive amount of traffic beyond its capacity. In this case, it is required to allocate some of the traffic to another available instance of the same security function. If there are no additional instances of the same security function available, we need to create a new SF instance and then direct the subsequent traffic to the new instance. In this way, we can avoid service congestion and achieve more efficient resource utilization. This process is commonly called load balancing.

In the SFC-enabled I2NSF architecture, SFC Catalog Manager performs periodic monitoring of the load status of available SF instances. In addition, it is possible to dynamically generate a new SF instance through Developer's Management System. With these functionalities along with the flexible traffic steering mechanism, we can eventually provide load balancing service.

The following describes the detailed process of load balancing when congestion occurs at the firewall instance:

1. SFC Catalog Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.

2. SFC Catalog Manager requests Developer's Management System to create a new firewall instance.

3. Developer's Management System creates a new firewall instance and then registers the information of the new firewall instance to SFC Catalog Manager.

4. SFC Catalog Manager updates the SFC Information Table to reflect the new firewall instance, and notifies SFC Policy Manager of this update.

5. Based on the received information, SFC Policy Manager updates the forwarding information for traffic steering and sends the new forwarding information to the SFF.

6. According to the new forwarding information, the SFF forwards the subsequent traffic to the new firewall instance. As a result, we can effecively alleviate the burden of the existing firewall instance.

6. Discussion

The information model and data model of security policy rules in the I2NSF framework defines an advanced security action as a type of action to be taken on a packet [nsf-capability-im][nsf-facing-inf-dm]. Through the advanced security action, a basic NSF (e.g., firewall) can call a different type of NSF for more in-depth security analysis of a packet. If an NSF triggers an advanced security action on a given packet, the packet should be forwarded to the NSF dedicated to the advanced action. That is, the advanced action dynamically determines the next NSF where the packet should go through. So if a forwarding component is configured with the network access information (e.g., IP address, port number) of the next required NSF, it can forward the packet to the NSF. With this advanced security action, it is possible to avoid the overhead for configuring and managing the information of security function chains and paths.

In SFC, re-classification is required to support the situation where the security function path of a packet changes dynamically, and the classifier is responsible for re-classification tasks to change the security function path of a packet. But if the classifier exists as a separate component from an NSF, the packet should be first delivered from the NSF to the classifier for re-classification, and this introduces an additional delay. As already mentioned, the advanced security action in the i2nsf framework can omit the requirement of pre-defined security function chain configuration. If there exists no security function chain/path configurations, there is no need of re-classification as well. That is, the forwarder can simply forward the packet to the next required NSF according to the advanced action determiend by the predesessor NSF, without re-classification through the classifier.

7. Implementation Considerations

7.1. SFC Policy Configuration and Management

In the SFC-enabled I2NSF architecture, SFC policy configuration and management should be considered to realize NSF chaining for packets. According to the given SFC policy, a classifier is configured with the corresponding NSF chain/path information, and also an SFF is configured with a forwarding information table.

The following three interfaces need to be considered for SFC policy configuration and management. First of all, the network administrator, typically an I2NSF user, needs to send SFC policy configuration information that should be enforced in the system to the security controller. Thus an interface between the network administrator and the security controller should be set up for this objective. By analyzing the given SFC policy configuration information, the security controller generates NSF chain/path information required for classifiers and forwarding information table of NSFs that SFFs require for packet forwarding. An interface between the security controller and classifier is required to deliver NSF chain/path information to the classifier. In addition, an interface between the security controller and SFF is also required to deliver forwarding information table of NSFs to SFFs.

When there are multiple instances of classifiers and SFFs, synchronizing the configuration information over them is important for them to have a consistent view of the configuration information. Therefore it should be considered how to synchronize the configuration information over the classifiers and SFFs.

7.2. Placement of Classifiers

To implement the SFC-enabled I2NSF architecture, it needs to be considered where to place the classifier. There are three possible options: NSF, SFF, and a separate component. The first option is integrating a classifier into every NSF. This approach is good for re-classification, because each NSF itself can perform re-classification without introducing any additional network overhead. On the other hand, configuring every NSF with NSF chain/path information and maintaining their consistency introduce an extra overhead. The second option is integrating a classifier into a SFF. In general, since the number of SFFs is much smaller than the number of NSFs, the overhead for configuring and managing NSF chain/path information would be smaller than the first option. In this case, re-classification of a packet should be done at a SFF rather than an NSF. The third option is that a classifier operates as a standalone entity. In this case, whenever re-classification is required for a packet, the packet should first stop by the classifier before going through a SFF, and this is likely to increase packet delivery latency.

7.3. Implementation of Network Tunneling

Tunneling protocols can be utilized to support packet forwarding between SFF and NSF or SFC proxy [RFC7665] . In this case, it needs to be considered which tunneling protocol to use, and both SFF and NSF/SFC proxy should support the same tunneling protocols. If an NSF itself should handle the tunneling protocol, it is required to modify the NSF implementation to make it understand the tunneling protocol. When there are diverse NSFs developed by different vendors, how to modify efficiently those NSFs to support the tunneling protocol is an critical issue to reduce the maintenance cost of NSFs after deployment.

We implemented network tunnleing based on GRE (Generic Routing Encapsulation) protocol to support packet forwarding between SFF and SFC proxy. For the NSH encapsulation with GRE protocol in layer 3, we referred to the header format defined in [RFC8300]. Figure 4 shows the format of an entire packet in our implementation, and Figure 5 shows the mapping table of service path identifiers used in our implementation.

```
+----------+----------------------+-------------+
|L2 header | L3 header(Outer IP), | GRE header, |
|          | protocol=47          | PT=0x894F   |
|          |                      |             |
+----------+----------------------+-------------+
     -----------+----------------+
     NSH, NP=0x1|                |
          SPI=1 |original packet |
          SI=1  |                |
     -----------+----------------+
```

Figure 4: Entire packet format for network tunneling based on GRE protocol

```
+------+------------------------------+
| SPI  | Network security function    |
+------+------------------------------+
| 1    | Firewall                     |
+------+------------------------------+
| 2    | Firewall->DPI                |
+------+------------------------------+
| 3    | Firewall->DPI->DDoS metigation|
+------+------------------------------+
```

Figure 5: Mapping table of service path identifiers

7.4. Implementation of SFC using Opendaylight Controller

Traffic steering in I2NSF framework can be implemented by using Opendaylight that supports service function chaining. In such a system where Opendaylight is integrated into I2NSF framework, traffic steering can be performed with three functions as follows. 1) I2NSF Security Controller Function 2) SDN Switch Controller Function 3) SDN Switch Traffic Steering Function. The following describes each of these functions.

What service function chains (SFC) are needed can be determined according to security policy rules of NSFs in I2NSF framework. I2NSF Security Controller Function identifies NSF chains that are required in the system by comprehensively analyzing security policy rules of NSFs. I2NSF Security Controller Function then generates the policies of the identified NSF chains, and requests SDN Switch Controller to enforce the policies of NSF chains.

SDN Switch Controller Function is responsible for creating, updating, and deleting SFCs, while Switch Controller operates to support service function chaining. Opendaylight Switch Controller is able to create key elements for SFC such as SF, SFF, SFC, SFP, and RSP. Once receiving the SFC policies from I2NSF Security Controller Function, SDN Switch Controller Function generates traffic forwarding rules that need to be configured on SFFs and switches, in order for the requested SFC policies to be enforced to relevant traffic. The generated traffic forwarding rules are delivered to relevant SFFs and switches using a data plane protocol ([OpenFlow], [RFC7047],[RFC6241]).

SFFs and switches perform forwarding traffic based on the traffic forwarding rules received from SDN Switch Controller. SDN Switch Traffic Steering Function refers to the Data Plane Elements processes running on SFFs and switches for steering traffic to the destination according to the traffic forwarding rules. To steer packets over NSFs, the packets are encapsulated with Network Service Header (NSH) [RFC8300].

8. Security Considerations

To enable security function chaining in the I2NSF framework, we adopt the additional components in the SFC architecture. Thus, this document shares the security considerations of the SFC architecture that are specified in [RFC7665] for the purpose of achieving secure communication among components in the proposed architecture.

9. Acknowledgments

This work was supported by Institute for Information and communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This document has greatly benefited from inputs by Sanguk Woo, Yunsuk Yeo, Taekyun Roh, and Sarang Wee.

10. References

10.1. Normative References

[OpenFlow]
Open Networking Foundation, "The OpenFlow Switch Specification, Version 1.4.0", OpenFlow https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, October 2013.

[RFC6241] Enns, R. and M. Bjorklund, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7047] Pfaff, B. and B. Davie, "The Open vSwitch Database Management Protocol", RFC 7047, December 2013.

[RFC7665] Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015.

[RFC8300] Quinn, Ed., P., Elzur, Ed., U., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, January 2018.

10.2. Informative References

[i2nsf-problem]
Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, July 2017.

[i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[nsf-capability-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-00 (work in progress), September 2017.

[nsf-facing-inf-dm]
Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-data-model-00 (work in progress), March 2018.

[ONF-SFC-Architecture]
ONF, "L4-L7 Service Function Chaining Solution Architecture", June 2015.

[RFC7498] Quinn, P. and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, April 2015.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

Appendix A. Changes from draft-hyun-i2nsf-nsf-triggered-steering-04

The following changes have been made from draft-hyun-i2nsf-nsf-triggered-steering-04:

- Section 7.4 has been added to discuss an implementation of traffic steering using Opendaylight controller that supports SFC.

- The references have been updated to reflect the latest documents.

Authors' Addresses

Sangwon Hyun
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: http://imtl.skku.ac.kr/

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1 734 604 0332
EMail: shares@ndzh.com

Network Working Group S. Hyun
Internet-Draft J. Jeong
Intended status: Standards Track T. Roh
Expires: September 6, 2018 S. Wi
Sungkyunkwan University
J. Park
ETRI
March 5, 2018

I2NSF Registration Interface YANG Data Model
draft-hyun-i2nsf-registration-interface-dm-03

Abstract

This document describes an YANG data model for I2NSF registration interface between Security Controller and Developer's Management System. The data model is required for NSF instance registration and dynamic life cycle management of NSF instances.

Table of Contents

1. Introduction

This document provides a YANG [RFC6020] data model that defines the required data for the registration interface between Security Controller and Developer's Management System to dynamically manage a pool of NSF instances. This document defines a YANG data model based on the [i2nsf-reg-inf-im]. The terms used in this document are defined in [i2nsf-terminology].

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 terminology described in [i2nsf-terminology], [capability-im], [RFC8329], [nsf-triggered-steering], [supa-policy-data-model], and [supa-policy-info-model].

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/ Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy. [nsf-triggered-steering]

- o Advanced Inspection/Action: As like the I2NSF information model for NSF facing interface [capability-im], Advanced Inspection/ Action means that a security function calls another security function for further inspection based on its own inspection result. [nsf-triggered-steering]

- o Network Security Function Profile (NSF Capability Information): NSF Capability Information specifies the inspection capabilities of the associated NSF instance. Each NSF instance has its own NSF Capability Information to specify the type of security service it provides and its resource capacity etc. [nsf-triggered-steering]

- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol. [supa-policy-info-model]

- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol. [supa-policy-info-model]

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] is as follows:

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

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

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

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

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

4. High-Level YANG

This section provides an overview of the high level YANG.

4.1. Registration Interface

```
module : ietf-i2nsf-regs-interface-model
  +--rw regs-req
  |  uses i2nsf-regs-req
  +--rw instance-mgnt-req
  |  uses i2nsf-instance-mgnt-req
```

Figure 1: High-Level YANG of I2NSF Registration Interface

Each of these sections mirror sections of [i2nsf-reg-inf-im].

4.2. Registration Request

This section expands the i2nsf-regs-req in Figure 1.

```
Registration Request
  +--rw i2nsf-regs-req
    +--rw nsf-capability-information
    |  uses i2nsf-nsf-capability-information
    +--rw nsf-access-info
    |  uses i2nsf-nsf-access-info
```

Figure 2: High-Level YANG of I2NSF Registration Request

Registration Request contains the capability information of newly created NSF to notify its capability to Security Controller. The request also contains Network Access Information so that the Security Controller can access the NSF.

4.3. Instance Management Request

This section expands the i2nsf-instance-mgnt-req in Figure 1.

```
Instance Management Request
  +--rw i2nsf-instance-mgnt-req
    +--rw req-level uint16
    +--rw req-id uint64
    +--rw (req-type)?
      +--rw (instanciation-request)
        +--rw nsf-capability-information
        |  uses i2nsf-nsf-capability-information
      +--rw (deinstanciation-request)
        +--rw nsf-access-info
        |  uses i2nsf-nsf-access-info
```

Figure 3: High-Level YANG of I2NSF Instance Mgnt Request

Instance management request consists of two types: instanciation-request and deinstanciation-request. The instanciation-request is used to request generation of a new NSF instance with NSF Capability Information which specifies required NSF capability information. The deinstanciation-request is used to remove an existing NSF with NSF Access Information.

4.4. NSF Capability Information

This section expands the i2nsf-nsf-capability-information in Figure 2 and Figure 3.

```
NSF Capability Information
  +--rw i2nsf-nsf-capability-information
    +--rw i2nsf-capability
    |  uses ietf-i2nsf-capability
    +--rw performance-capability
    |  uses i2nsf-nsf-performance-caps
```

Figure 4: High-Level YANG of I2NSF NSF Capability Information

In Figure 4, ietf-i2nsf-capability refers module ietf-i2nsf-capability in [i2nsf-capability-dm]. We add the performance capability because it is absent in [i2nsf-capability-dm] and [netmod-acl-model]

4.5. NSF Access Information

This section expands the i2nsf-nsf-access-info in Figure 2 and Figure 3.

```
NSF Access Information
  +--rw i2nsf-nsf-access-info
    +--rw nsf-address  inet:ipv4-address
    +--rw nsf-port-address inet:port-number
```

Figure 5: High-Level YANG of I2NSF NSF Access Informantion

This information is used by other components to access an NSF.

4.6. NSF Performance Capability

This section expands the i2nsf-nsf-performance-caps in Figure 4.

```
NSF Performance Capability
  +--rw i2nsf-nsf-performance-caps
   +--rw processing
   |   +--rw processing-average uint16
   |   +--rw processing-peak uint16
   +--rw bandwidth
   |   +--rw outbound
   |   |  +--rw outbound-average uint16
   |   |  +--rw outbound-peak uint16
   |   +--rw inbound
   |   |  +--rw inbound-average uint16
   |   |  +--rw inbound-peak uint16
```

Figure 6: High-Level YANG of I2NSF NSF Performance Capability

When the Security Controller requests the Developer Management System to create a new NSF instance, the performance capability is used to specify the performance requirements of the new instance.

4.7. Role-Based ACL(Access Control List)

This section expands the ietf-netmod-acl-model in [netmod-acl-model].

```
Role-Based ACL
  +--rw role-based-acl
       uses ietf-netmod-acl-model
```

Figure 7: Role-Based ACL

In [netmod-acl-model], ietf-netmod-acl-model refers module ietf-netmod-acl-model in [netmod-acl-model]. We add the role-based ACL because it is absent in [i2nsf-capability-dm].

5. YANG Modules

This section introduces a YANG module for the information model of the required data for the registration interface between Security Controller and Developer’s Management System, as defined in the [i2nsf-reg-inf-im].

```
<CODE BEGINS> file "ietf-i2nsf-regs-interface@2018-03-05.yang"
module ietf-i2nsf-regs-interface {
      namespace
       "urn:ietf:params:xml:ns:yang:ietf-i2nsf-regs-interface";
      prefix
        regs-interface;

      import ietf-inet-types{
       prefix inet;
      }

      organization
        "IETF I2NSF (Interface to Network Security Functions)
        Working Group";

      contact
       "WG Web: <http://tools.ietf.org/wg/i2nsf>
       WG List: <mailto:i2nsf@ietf.org>

       WG Chair: Adrian Farrel
       <mailto:Adrain@olddog.co.uk>

       WG Chair: Linda Dunbar
       <mailto:Linda.duhbar@huawei.com>

       Editor: Sangwon Hyun
       <mailto:swhyun77@skku.edu>

       Editor: Taekyun Roh
       <mailto:tkroh0198@skku.edu>

       Editor: Sarang Wi
       <mailto:dnl9795@skku.edu>

       Editor: Jaehoon Paul Jeong
       <mailto:pauljeong@skku.edu>
```

```
          Editor: Jung-Soo Park
          <mailto:pjs@etri.re.kr>";

         description
           "It defines a YANG data module for Registration Interface.";
         revision "2018-03-05"{
           description "The second revision";
           reference
           "draft-hares-i2nsf-capability-data-model-03
           draft-hyun-i2nsf-registration-interface-im-04";
         }

       grouping i2nsf-nsf-performance-caps {
           description
           "NSF performance capailities";
           container processing{
             description
             "processing info";
             leaf processing-average{
              type uint16;
              description
              "processing-average";
             }
             leaf processing-peak{
              type uint16;
              description
              "processing peak";
             }
           }

           container bandwidth{
             description
             "bandwidth info";
             container inbound{
              description
              "inbound";
              leaf inbound-average{
               type uint16;
               description
               "inbound-average";
              }
              leaf inbound-peak{
               type uint16;
               description
               "inbound-peak";
              }
             }
```

```
             container outbound{
              description
              "outbound";
              leaf outbound-average{
               type uint16;
               description
               "outbound-average";
              }
              leaf outbound-peak{
               type uint16;
               description
               "outbound-peak";
              }
             }
           }
       }

       grouping i2nsf-nsf-capability-information {
         description
         "Detail information of an NSF";
         container performance-capability {
          uses i2nsf-nsf-performance-caps;
          description
          "performance-capability";
         }

         container i2nsf-capability {
          description
          "It refers draft-hares-i2nsf-capability-data-model-04.txt
          later";
         }
       }

       grouping ietf-netmod-acl-model {
         description
         "Detail information";
         container role-based-acl {
          description
          "It refers draft-ietf-netmod-acl-model-15.txt
          later";
         }
       }

       grouping i2nsf-nsf-access-info {
         description
         "NSF access information";
         leaf nsf-address {
          type inet:ipv4-address;
```

```
           mandatory true;
           description
           "nsf-address";
          }

          leaf nsf-port-address {
           type inet:port-number;
           description
           "nsf-port-address";
          }
        }

        grouping i2nsf-regs-req {
          description
          "The capability information of newly
          created NSF to notify its
          capability to Security Controller";
          container nsf-capability-information {
           description
           "nsf-capability-information";
           uses i2nsf-nsf-capability-information;
          }

          container nsf-access-info {
           description
           "nsf-access-info";
           uses i2nsf-nsf-access-info;
          }
        }

        grouping i2nsf-instance-mgnt-req {
          description
          "Required information for instanceiation-request and
          deinstanciation-request";
          leaf req-level {
           type uint16;
           description
           "req-level";
          }

          leaf req-id {
           type uint64;
           mandatory true;
           description
           "req-id";
          }

          choice req-type {
```

```
          description
          "req-type";
          case instanciation-request {
           description
           "instanciation-request";
           container nsf-capability-information {
             description
             "nsf-capability-information";
             uses i2nsf-nsf-capability-information;
           }
          }

          case deinstanciation-request {
           description
           "deinstanciation-request";
           container nsf-access-info {
             description
             "nsf-access-info";
             uses i2nsf-nsf-access-info;
           }
         }
        }
       }
   }

   <CODE ENDS>
```

Figure 8: Data Model of I2NSF Registration Interface

5.1. XML Example of Registration Interface Data Model

Requirement: Registering the IDS NSF with VoIP/VoLTE security capability using Registration interface.

Here is the configuration xml for this Registration Interface:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
    <i2nsf-regs-req>
      <i2nsf-nsf-capability-information>
        <ietf-i2nsf-capability>
          <nsf-capabilities>
           <nsf-capabilities-id>1</nsf-capabilities-id>
```

```
               <con-sec-control-capabilities>
                <content-security-control>
                 <ids>
                  <ids-support>true</ids-support>
                  <ids-fcn nc:operation="create">
                   <ids-fcn-name>ids-service</ids-fcn-name>
                  </ids-fcn>
                 </ids>
                 <voip-volte>
                  <voip-volte-support>true</voip-volte-support>
                  <voip-volte-fcn nc:operation="create">
                   <voip-volte-fcn-name>
                    ips-service
                   </voip-volte-fcn-name>
                  </voip-volte-fcn>
                 </voip-volte>
                </content-security-control>
              </con-sec-control-capabilities>
             </nsf-capabilities>
           </ietf-i2nsf-capability>
           <i2nsf-nsf-performance-caps>
             <processing>
              <processing-average>1000</processing-average>
              <processing-peak>5000</processing-peak>
             </processing>
             <bandwidth>
              <outbound>
               <outbound-average>1000</outbound-average>
               <outbound-peak>5000</outbound-peak>
              </outbound>
              <inbound>
               <inbound-average>1000</inbound-average>
               <inbound-peak>5000</inbound-peak>
              </inbound>
             </bandwidth>
           </i2nsf-nsf-performance-caps>
         </i2nsf-nsf-capability-information>
         <nsf-access-info>
          <nsf-address>10.0.0.1</nsf-address>
          <nsf-port-address>145</nsf-port-address>
         </nsf-access-info>
       </i2nsf-regs-req>
       </config>
     </edit-config>
   </rpc>
```


Figure 9: Registration Interface example

6. Security Considerations

The information model of the registration interface is based on the I2NSF framework without any architectural changes. Thus, this document shares the security considerations of the I2NSF framwork architecture that are specified in [RFC8329] for the purpose of achieving secure communication among components in the proposed architecture.

7. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs toIndicate Requirement Levels", RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

8.2. Informative References

[capability-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-i2nsf-capability-00 (work in progress), September 2017.

[i2nsf-capability-dm]
Hares, S., Jeong, J., Kim, J., Moskowitz, R., and Q. Lin, "I2NSF Capability YANG Data Model", draft-hares-i2nsf-capability-data-model-06 (work in progress), March 2018.

[i2nsf-reg-inf-im]
Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface Information Model", draft-hyun-i2nsf-registration-interface-im-04 (work in progress), October 2017.

[i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[i2rs-rib-data-model]
Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-10 (work in progress), February 2018.

[netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-16 (work in progress), February 2018.

[nsf-triggered-steering]
Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-05 (work in progress), March 2018.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

[supa-policy-data-model]
Halpern, J., Strassner, J., and S. van der Meer, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-data-model-04 (work in progress), June 2017.

[supa-policy-info-model]
Strassner, J., Halpern, J., and S. van der Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-03 (work in progress), May 2017.

Appendix A. Changes from draft-hyun-i2nsf-registration-interface-dm-02

The following changes are made from draft-hyun-i2nsf-registration-interface-dm-02:

- We updated the name of NSF profile to NSF capability information and contents.
- We added role-based ACL.
- The description of a YANG data module for Registration Interface was updated.
- The references were updated to reflect the latest documents.

Authors' Addresses

Sangwon Hyun
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: http://imtl.skku.ac.kr/

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Taekyun Roh
Electrical Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: tkroh0198@skku.edu
URI: http://imtl.skku.ac.kr/xe/index.php?mid=board_YoKq57

Sarang Wi
Electrical Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: dnl9795@skku.edu
URI: http://imtl.skku.ac.kr/xe/index.php?mid=board_YoKq57

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

S. Hyun
J. Jeong
T. Roh
S. Wi
Sungkyunkwan University
J. Park
ETRI

March 5, 2018

I2NSF Registration Interface Information Model
draft-hyun-i2nsf-registration-interface-im-04

Abstract

This document describes an information model for Interface to Network Security Functions (I2NSF) Registration Interface between Security Controller and Developer's Management System (DMS). The information model is required to support NSF instance creation, registration, and deletion request via the registration interface. This document explains the procedures over I2NSF registration interface for these functionalities. It also describes the detailed information which should be exchanged via I2NSF registration interface.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A number of virtual network security function instances typically exist in Interface to Network Security Functions (I2NSF) framework [RFC8329]. Since these NSF instances may have different security capabilities, it is important to register the security capabilities of each NSF instance into the security controller after they have been created. In addition, it is required to instantiate NSFs of some required security capabilities on demand. As an example, if additional security capabilities are required to meet the new security requirements that an I2NSF user requests, the security controller should be able to request the DMS to instantiate NSFs that have the required security capabilities.

This document describes the information model which is required for the registration interface between security controller and developer's management system to support registration and instantiation of NSFs. It further describes the procedure based on the information model which should be performed by the security controller and the developer's management system via the registration interface.

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 terminology described in [i2nsf-terminology][capability-im][RFC8329] [nsf-triggered-steering].

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/ Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy [nsf-triggered-steering].

- o Advanced Inspection/Action: As like the I2NSF information model for NSF-facing interface [capability-im], Advanced Inspection/ Action means that a security function calls another security function for further inspection based on its own inspection result [nsf-triggered-steering].

- o Network Security Function Profile (NSF Profile): NSF Profile specifies the security and performance capability of an NSF instance. Each NSF instance has its own NSF Profile which describes the type of security service it can provide and its performance capability. [nsf-triggered-steering].

4. Objectives

- o Registering NSF instances from Developer's Management System: Depending on system's security requirements, it may require some NSFs by default. In this case, DMS creates these default NSF instances without the need of receiving requests from Security Controller. After creating them, DMS notifies Security Controller of those NSF instances via registration interface.

- o Creating an NSF instance to serve advanced inspection/action triggered by another NSF: In I2NSF framework, an NSF can trigger another type of NSF(s) for more advanced security inspection of the traffic. In this case, the next NSF is determined by the current NSF's inspection result and I2NSF user's policy. However, if there is no available NSF instance to serve the advanced inspection triggered by the former NSF, we should create an NSF instance by requesting Developer's Management System (DMS) through registration interface.

- o Creating NSF instances required to enforce security policy rules from I2NSF user: In I2NSF framework, I2NSF users decide which security service is necessary in the system. If there is no NSF instances to enforce the security policy requested by I2NSF users, then we should also create the required NSF instances by requesting DMS via registration interface.

- o Deleting an NSF instance that is no longer required: Various types of NSF instances are running in I2NSF framework, and according to dynamic changes on the security policy to be enfored in the system, some types of NSF instances may be no longer required. In this case, Security Controller may request DMS to destroy those NSF instances via registration interface.

5. Information Model

The I2NSF registration interface was only used for registering new NSF instances to Security Controller. In this document, however, we extend its utilization to support on demand NSF instantiation/de-instantiation and describe the information that should be exchanged via the registration interface for the functionality. Moreover, we also define the information model of NSF Profile because, for registration interface, NSF Profile (i.e., capabilities of an NSF) needs to be clarified so that the components of I2NSF framework can exchange the set of capabilities in a standardized manner. This is typically done through the following process:

1) Security Controller first recognizes the set of capabilities (i.e., NSF Profile) or the signature of a specific NSF required or wasted in the current system.

2) Developer's Management System (DMS) matches the recognized information to an NSF based on the information model definition.

3) Developer's Management System creates or eliminates NSFs matching with the above information.

4) Security Controller can then add/remove the corresponding NSF instance to/from its list of available NSF instances in the system.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Registration Interface Information Model                          |
|                                                                    |
|       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+       +-+-+-+-+-+-+-+-+-+-+-+   |
|       |    Instance Management  |       |    Registration    |   |
|       |         Sub-Model       |       |     Sub-Model      |   |
|       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+       +-+-+-+-+-+-+-+-+-+-+-+   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Registration Interface Information Model

As illustrated in Figure 1, the information model for Registration Interface consists of two sub-models: instance management, registration sub-models. The instance management functionality and the registration functionality use NSF Profile to achieve their goals. In this context, NSF Profile is the capability objects that describe and/or prescribe inspection capability an NSF instance can provide.

5.1. NSF Instance Managment Mechanism

For the instance management of NSFs, Security Controller in I2NSF framework requires two types of requests: Instantiation Request and Deinstantiation Request. Security Controller sends the request messages to DMS when required. Once receiving the request, DMS conducts creating/eliminating the corresponding NSF instance and responds Security Controller with the results.

```
+-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+-+-+-+
|  Instantiation  |        | De-instantiation  |
|     Request     |        |      Request      |
+-+-+-+-+-^-+-+-+-+-+        +-+-+-+-+-+-^-+-+-+-+-+
          |                            |
          |                            |
          |                            |
          |                            |
+-+-+-+-+-+-+-+-+-++            +-+-+-+-+-+-+-+-+-+
| NSF Capability |            |  NSF Access   |
|  Information   |            |  Information  |
+-+-+-+-+-+-+-+-+-++            +-+-+-+-+-+-+-+-+-+
```

Figure 2: Instance Management Sub-Model

5.2. NSF Registration Mechanism

In order to register a new NSF instance, DMS should generate a Registration Message to Security Controller. A Registration Message consists of an NSF Profile and an NSF Access Information. The former describes the inspection capability of the new NSF instance and the latter is for enabling network access to the new instance from other components. After this registration process, as explained in [capability-im], the I2NSF capability interface can conduct controlling and monitoring the new registered NSF instance.

```
                        +-+-+-+-+-+-+-+-+-+
                        |       NSF       |
                        |  Registration   |
                        +-+-+-+-+-^-+-+-+-+
                                  |
                  +-----------------------------+
                  |                             |
                  |                             |
           +-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+
           |  NSF Capability   |        | NSF Access    |
           |   Information     |        | Information   |
           +-+-+-+-+-+-+-+-+-+-+        +-+-+-+-+-+-+-+-+
```

Figure 3: Registration Sub-Model

5.3. NSF Access Information

NSF Access Information contains the followings that are required to communicate with an NSF: IPv4 address, IPv6 address, port number, and supported transport protocol(s) (e.g., Virtual Extensible LAN (VXLAN) [RFC 7348], Generic Protocol Extension for VXLAN (VXLAN-GPE) [nvo3-vxlan-gpe], Generic Route Encapsulation (GRE), Ethernet etc.). In this document, NSF Access Information is used to identify a specific NSF instance (i.e. NSF Access Information is the signature(unique identifier) of an NSF instance in the overall system).

5.4. NSF Capability Information

NSF Capability Information describes the security capabilities of an NSF instance. In Figure 4, we show capability objects of an NSF instance. Following the information model of NSF capabilities defiend in [capability-im], we share the same security capabilities: Network-Security Capabilities, Content-Security Capabilities, and Attack Mitigation Capabilities. Also, NSF Capability Information additionally contains the performance capabilities and role-Based access control list (ACL) as shown in Figure 4.

```
                         +-+-+-+-+-+-+-+-+-+
                         |  Capability   |
                         |    Objects    |
                         +-+-+-+-+^-+-+-+-+
                                 |
         +-----------------------+----------------------+----------+
         |                       |                      |          |
         |                       |                      |          |
+-+-+-+-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+-+-+-+ |
|Network-Security |    |Content-Security |    |Attack Mitigation| |
|   Capabilities  |    |   Capabilities  |    |   Capabilities  | |
+-+-+-+-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+-+-+-+ |
                                                                   |
                     +-----------------------+---------------------+
                     |                       |
                     |                       |
            +-+-+-+-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+-+-+-+
            |   Performance   |    |    Role-Based   |
            |   Capabilities  |    |       ACL       |
            +-+-+-+-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+-+-+-+
```

Figure 4: NSF Profile Overview

5.4.1. Performance Capabilities

This information represents the processing capability of an NSF. This information can be used to determine whether the NSF is in congestion by comparing this with the workload that the NSF currently undergoes. Moreover, this information can specify an available amount of each type of resources such as processing power which are available on the NSF. (The registration interface can control the usages and limitations of the created instance and make the appropriate request according to the status.) As illustrated in Figure 5, this information consists of two items: Processing and Bandwidth. Processing information describes the NSF’s available processing power. Bandwidth describes the information about available network amount in two cases, outbound, inbound. This two information can be used for the NSF’s instance request.

```
                  +-+-+-+-+-+-+-+-+-+-+
                  |   Performance    |
                  |   Capabilities   |
                  +-+-+-+-+-^-+-+-+-+-+
                            |
            +-----------------------------+
            |                             |
            |                             |
    +-+-+-+-+-+-+-+-+-+         +-+-+-+-+-+-+-+-+
    |  Processing   |           |  Bandwidth  |
    +-+-+-+-+-+-+-+-+-+         +-+-+-+-+-+-+-+-+
```

Figure 5: Performance Capability Overview

5.4.2. Role-based Access Control List

This information specifies access policies of an NSF to determine whether to permit or deny the access of an entity to the NSF based on the role given to the entity. Each NSF is associated with a role-based access control list (ACL) so that it can determine whether to permit or deny the access request from an entity. Figure 6 and Figure 7 show the structure of the role-based ACL, which is composed of role-id, access-type, and permit/deny. The role-id identifies roles of entities (e.g., administrator, developer etc.). The access-type identifies the specific type of access requests such as NSF rule configuration/update and NSF monitoring. Consequently, the role-based ACL in Figure 6 and Figure 7 specifies a set of access-types to be permitted and to be denied by each role-id.

```
                    +-+-+-+-+-+-+-+-+-+
                    |   Role-based   |
                    |      ACL       |
                    +-+-+-+-+-+-+-+-+-+
                            |
          +------------------------------------+
          |                                    |
    +-+-+-+-+-+-+-+                      +-+-+-+-+-+-+-+
    | Role-id 1 |          ...           | Role-id N |
    +-+-+-+-+-+-+-+                      +-+-+-+-+-+-+-+
```

Figure 6: Role-based Access Control List

```
                              +-+-+-+-+-+-+-+-+-+
                              |   Role-id i    |
                              +-+-+-+-+-+-+-+-+-+
                                       |
                      +----------------------------------+
                      |                                  |
                +-+-+-+-+-+-+-+                   +-+-+-+-+-+-+-+
                |   Permit  |                     |   Deny    |
                +-+-+-+-+-+-+-+                   +-+-+-+-+-+-+-+
                      |                                  |
            +-------------------+              +-------------------+
            |                   |              |                   |
      +-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+
      |access-type| ...  |access-type|  |access-type| ...  |access-type|
      |     p1    |      |     pn    |  |     d1    |      |     dn    |
      +-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+    +-+-+-+-+-+-+-+
```

Figure 7: Role-id Subtree

6. Security Considerations

The information model of the registration interface is based on the I2NSF framework without any architectural changes. Thus, this document shares the security considerations of the I2NSF framwork that are specified in [RFC8329] for the purpose of achieving secure communication between components in the proposed architecture.

7. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This document has greatly benefited from inputs by SangUk Woo and YunSuk Yeo.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[capability-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-00 (work in progress), September 2017.

[i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[nsf-triggered-steering]
Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-05 (work in progress), March 2018.

[nvo3-vxlan-gpe]
Maino, Ed., F., Kreeger, Ed., L., and U. Elzur, Ed., "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-05 (work in progress), October 2017.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

Appendix A. Changes from draft-hyun-i2nsf-registration-interface-im-03

The following changes have been made from draft-hyun-i2nsf-registration-interface-im-03:

- o Section 4 has been revised to discuss about destructing an NSF instance no longer required via registration interface.
- o We changed "NSF profile" into "NSF capability information."
- o Figures 1, 2 and 3 have been updated.

Authors' Addresses

Sangwon Hyun
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: http://imtl.skku.ac.kr/

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

TaeKyun Roh
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: tkroh0198@skku.edu
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

SaRang Wi
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: dnl9795@skku.edu
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

I2NSF Working Group J. Jeong
Internet-Draft Sungkyunkwan University
Intended status: Informational S. Hyun
Expires: March 18, 2020 Myongji University
T. Ahn
Korea Telecom
S. Hares
Huawei
D. Lopez
Telefonica I+D
September 15, 2019

Applicability of Interfaces to Network Security Functions to Network-Based Security Services
draft-ietf-i2nsf-applicability-18

Abstract

This document describes the applicability of Interface to Network Security Functions (I2NSF) to network-based security services in Network Functions Virtualization (NFV) environments, such as firewall, deep packet inspection, or attack mitigation engines.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Interface to Network Security Functions (I2NSF) defines a framework and interfaces for interacting with Network Security Functions (NSFs). Note that an NSF is defined as software that provides a set of security-related services, such as (i) detecting unwanted activity, (ii) blocking or mitigating the effect of such unwanted activity in order to fulfill service requirements, and (iii) supporting communication stream integrity and confidentiality [i2nsf-terminology].

The I2NSF framework allows heterogeneous NSFs developed by different security solution vendors to be used in the Network Functions Virtualization (NFV) environment [ETSI-NFV] by utilizing the capabilities of such NSFs through I2NSF interfaces such as Customer-Facing Interface [consumer-facing-inf-dm] and NSF-Facing Interface [nsf-facing-inf-dm]. In the I2NSF framework, each NSF initially registers the profile of its own capabilities with the Security Controller (i.e., network operator management system [RFC8329]) of the I2NSF system via the Registration Interface [registration-inf-dm]. This registration enables an I2NSF User (i.e., network security administrator) to select and use the NSF to enforce a given security policy. Note that Developer’s Management System (DMS) is management software that provides a vendor’s security service software as a Virtual Network Function (VNF) in an NFV environment (or middlebox in the legacy network) as an NSF, and registers the capabilities of an NSF into Security Controller via Registration Interface for a security service [RFC8329].

Security Controller maintains the mapping between a capability and an NSF, so it can perform to translate a high-level security policy received from I2NSF User to a low-level security policy configured and enforced in an NSF [policy-translation]. Security Controller can monitor the states and security attacks in NSFs through NSF monitoring [nsf-monitoring-dm].

This document illustrates the applicability of the I2NSF framework with five different scenarios:

1. The enforcement of time-dependent web access control.

2. The support of intent-based security services through I2NSF and Security Policy Translator [policy-translation].

3. The application of I2NSF to a Service Function Chaining (SFC) environment [RFC7665].

4. The integration of the I2NSF framework with Software-Defined Networking (SDN) [RFC7149] to provide different security functionality such as firewalls [opsawg-firewalls], Deep Packet Inspection (DPI), and Distributed Denial of Service (DDoS) attack mitigation.

5. The use of Network Functions Virtualization (NFV) [ETSI-NFV] as a supporting technology.

The implementation of I2NSF in these scenarios has allowed us to verify the applicability and effectiveness of the I2NSF framework for a variety of use cases.

2. Terminology

This document uses the terminology described in [RFC7665], [RFC7149], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ITU-T.X.800], [NFV-Terminology], [RFC8329], and [i2nsf-terminology]. In addition, the following terms are defined below:

- Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation.

- Centralized Firewall System: A centralized firewall that can establish and distribute policy rules into network resources for efficient firewall management.

- Centralized VoIP Security System: A centralized security system that handles the security functions required for VoIP and VoLTE services.

- Firewall: A service function at the junction of two network segments that inspects some suspicious packets that attempt to cross the boundary. It also rejects any packet that does not satisfy certain criteria for, for example, disallowed port numbers or IP addresses.

- Network Function: A functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behavior [NFV-Terminology].

- Network Functions Virtualization (NFV): A principle of separating network functions (or network security functions) from the hardware they run on by using virtual hardware abstraction [NFV-Terminology].

- Network Security Function (NSF): Software that provides a set of security-related services. Examples include detecting unwanted activity and blocking or mitigating the effect of such unwanted activity in order to fulfill service requirements. The NSF can also help in supporting communication stream integrity and confidentiality [i2nsf-terminology].

- Security Policy Translator (SPT): Software that translates a high-level security policy for the Consumer-Facing Interface into a low-level security policy for the NSF-Facing Interface [policy-translation]. The SPT is a core part of the Security Controller in the I2NSF system.

- o Service Function Chaining (SFC): The execution of an ordered set of abstract service functions (i.e., network functions) according to ordering constraints that must be applied to packets, frames, and flows selected as a result of classification. The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied [RFC7665].

- o Software-Defined Networking (SDN): A set of techniques that enables to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

```
   +------------+
   | I2NSF User |
   +------------+
          ^
          | Consumer-Facing Interface
          v
+-------------------+      Registration     +-----------------------+
|Security Controller|<--------------------->|Developer's Mgmt System|
+-------------------+       Interface       +-----------------------+
          ^
          | NSF-Facing Interface
          v
   +----------------+ +---------------+   +-----------------------+
   |     NSF-1      |-|     NSF-2     |...|         NSF-n         |
   |   (Firewall)   | | (Web Filter)  |   |(DDoS-Attack Mitigator)|
   +----------------+ +---------------+   +-----------------------+
```

Figure 1: I2NSF Framework

3. I2NSF Framework

This section summarizes the I2NSF framework as defined in [RFC8329]. As shown in Figure 1, an I2NSF User can use security functions by delivering high-level security policies, which specify security requirements that the I2NSF user wants to enforce, to the Security Controller via the Consumer-Facing Interface (CFI) [consumer-facing-inf-dm].

The Security Controller receives and analyzes the high-level security policies from an I2NSF User, and identifies what types of security capabilities are required to meet these high-level security policies. The Security Controller then identifies NSFs that have the required security capabilities, and generates low-level security policies for each of the NSFs so that the high-level security policies are eventually enforced by those NSFs [policy-translation]. Finally, the Security Controller sends the generated low-level security policies to the NSFs via the NSF-Facing Interface (NFI) [nsf-facing-inf-dm].

As shown in Figure 1, with a Developer's Management System (called DMS), developers (or vendors) inform the Security Controller of the capabilities of the NSFs through the Registration Interface (RI) [registration-inf-dm] for registering (or deregistering) the corresponding NSFs. Note that the lifecycle management of NSF code from DMS (e.g., downloading of NSF modules and testing of NSF code) is out of scope for I2NSF.

The Consumer-Facing Interface can be implemented with the Consumer-Facing Interface YANG data model [consumer-facing-inf-dm] using RESTCONF [RFC8040] which befits a web-based user interface for an I2NSF User to send a Security Controller a high-level security policy. Data models specified by YANG [RFC6020] describe high-level security policies to be specified by an I2NSF User. The data model defined in [consumer-facing-inf-dm] can be used for the I2NSF Consumer-Facing Interface. Note that an inside attacker at the I2NSF User can misuse the I2NSF system so that the network system under the I2NSF system is vulnerable to security attacks. To handle this type of threat, the Security Controller needs to monitor the activities of all the I2NSF Users as well as the NSFs through the I2NSF NSF monitoring functionality [nsf-monitoring-dm]. Note that the monitoring of the I2NSF Users is out of scope for I2NSF.

The NSF-Facing Interface can be implemented with the NSF-Facing Interface YANG data model [nsf-facing-inf-dm] using NETCONF [RFC6241] which befits a command-line-based remote-procedure call for a Security Controller to configure an NSF with a low-level security policy. Data models specified by YANG [RFC6020] describe low-level security policies for the sake of NSFs, which are translated from the high-level security policies by the Security Controller. The data model defined in [nsf-facing-inf-dm] can be used for the I2NSF NSF-Facing Interface.

The Registration Interface can be implemented with the Registration Interface YANG data model [registration-inf-dm] using NETCONF [RFC6241] which befits a command-line-based remote-procedure call for a DMS to send a Security Controller an NSF's capability information. Data models specified by YANG [RFC6020] describe the registration of an NSF's capabilities to enforce security services at the NSF. The data model defined in [registration-inf-dm] can be used for the I2NSF Registration Interface.

The I2NSF framework can chain multiple NSFs to implement low-level security policies with the SFC architecture [RFC7665].

The following sections describe different security service scenarios illustrating the applicability of the I2NSF framework.

```
<?xml version="1.0" encoding="UTF-8" ?>
<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy">
  <policy-name>block_website</policy-name>
  <rule>
    <rule-name>block_website_during_working_hours</rule-name>
    <event>
      <time-information>
        <begin-time>09:00</begin-time>
        <end-time>18:00</end-time>
      </time-information>
    </event>
    <condition>
      <firewall-condition>
        <source-target>
          <src-target>Staff_Members'_PCs</src-target>
        </source-target>
      </firewall-condition>
      <custom-condition>
        <destination-target>
          <dest-target>SNS_Websites</dest-target>
        </destination-target>
      </custom-condition>
    </condition>
    <action>
      <primary-action>drop</primary-action>
    </action>
  </rule>
</policy>
```

Figure 2: A High-level Security Policy XML File for Time-based Web Filter

```
<?xml version="1.0" encoding="UTF-8" ?>
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
  <system-policy>
    <system-policy-name>block_website</system-policy-name>
    <rules>
      <rule-name>block_website_during_working_hours</rule-name>
      <time-intervals>
        <absolute-time-interval>
          <begin-time>09:00</begin-time>
          <end-time>18:00</end-time>
        </absolute-time-interval>
      </time-intervals>
      <condition-clause-container>
        <packet-security-ipv6-condition>
          <pkt-sec-ipv6-src>
            <ipv6-address>
              <ipv6>2001:DB8:10:1::10</ipv6>
              <ipv6>2001:DB8:10:1::20</ipv6>
              <ipv6>2001:DB8:10:1::30</ipv6>
            </ipv6-address>
          </pkt-sec-ipv6-src>
        </packet-security-ipv6-condition>
        <packet-security-url-category-condition>
          <user-defined-category>example1.com</user-defined-category>
          <user-defined-category>example2.com</user-defined-category>
          <user-defined-category>example3.com</user-defined-category>
          <user-defined-category>example4.com</user-defined-category>
        </packet-security-url-category-condition>
      </condition-clause-container>
      <action-clause-container>
        <packet-action>
          <egress-action>drop</egress-action>
        </packet-action>
      </action-clause-container>
    </rules>
  </system-policy>
</i2nsf-security-policy>
```


Figure 3: A Low-level Security Policy XML File for Time-based Web Filter

4. Time-dependent Web Access Control Service

This service scenario assumes that an enterprise network administrator wants to control the staff members' access to a particular Internet service (e.g., social networking service (SNS)) during business hours. The following is an example high-level security policy rule for a web filter that the administrator requests: Block the staff members' access to SNS websites from 9 AM (i.e., 09:00) to 6 PM (i.e., 18:00) by dropping their packets. Figure 2 is a high-level security policy XML code for the web filter that is sent from the I2NSF User to the Security Controller via the Consumer-Facing Interface [consumer-facing-inf-dm].

The security policy name is "block_website" with the tag "policy-name", and the security policy rule name is "block_website_during_working_hours" with the tag "rule-name". The filtering event has the time span where the filtering begin time is the time "09:00" (i.e., 9:00AM) with the tag "begin-time", and the filtering end time is the time "18:00" (i.e., 6:00PM) with the tag "end-time". The filtering condition has the source target of "Staff_Members'_PCs" with the tag "src-target", and the destination target of "SNS_Websites" with the tag "dest-target".

Assume that "Staff_Members'_PCs" are 2001:DB8:10:1::10, 2001:DB8:10:1::20, and 2001:DB8:10:1::30, and that "SNS_Websites" are example1.com, example2.com, example3.com, and example4.com, as shown in Figure 3. Note that Figure 3 is a low-level security policy XML code for the web filter that is sent from the Security Controller to an NSF via the NSF-Facing Interface [nsf-facing-inf-dm].

The source target can by translated by the Security Policy Translator (SPT) in the Security Controller to the IP addresses of computers (or mobile devices) used by the staff members. Refer to Section 5 for the detailed description of the SPT. The destination target can also be translated by the SPT to the actual websites corresponding to the symbolic website name "SNS_Websites", and then either each website's URL or the corresponding IP address(es) can be used by both firewall and web filter. The action is to "drop" the packets satisfying the above event and condition with the tag "primary-action".

After receiving the high-level security policy, the Security Controller identifies required security capabilities, e.g., IP address and port number inspection capabilities and URL inspection capability. In this scenario, it is assumed that the IP address and port number inspection capabilities are required to check whether a received packet is an HTTP-session packet from a staff member, which is part of an HTTP session generated by the staff member. The URL inspection capability is required to check whether the target URL of a received packet is one of the target websites (i.e., example1.com, example2.com, example3.com, and example4.com) or not.

The Security Controller maintains the security capabilities of each active NSF in the I2NSF system, which have been reported by the Developer's Management System via the Registration interface. Based on this information, the Security Controller identifies NSFs that can perform the IP address and port number inspection and URL inspection through the security policy translation in Section 5. In this scenario, it is assumed that a firewall NSF has the IP address and port number inspection capabilities and a web filter NSF has URL inspection capability.

The Security Controller generates a low-level security policy for the NSFs to perform IP address and port number inspection, URL inspection, and time checking, which is shown in Figure 3. Specifically, the Security Controller may interoperate with an access control server in the enterprise network in order to retrieve the information (e.g., IP address in use, company identifier (ID), and role) of each employee that is currently using the network. Based on the retrieved information, the Security Controller generates a low-level security policy to check whether the source IP address of a received packet matches any one being used by a staff member.

In addition, the low-level security policy’s rule (shortly, low-level security rule) should be able to determine that a received packet uses either the HTTP protocol without Transport Layer Security (TLS) [RFC8446] or the HTTP protocol with TLS as HTTPS. The low-level security rule for web filter checks that the target URL field of a received packet is equal to one of the target SNS websites (i.e., example1.com, example2.com, example3.com, and example4.com), or that the destination IP address of a received packet is an IP address corresponding to one of the SNS websites. Note that if HTTPS is used for an HTTP-session packet, the HTTP protocol header is encrypted, so the URL information may not be seen from the packet for the web filtering. Thus, the IP address(es) corresponding to the target URL needs to be obtained from the certificate in TLS versions prior to 1.3 [RFC8446] or the Server Name Indication (SNI) in a TCP-session packet in TLS versions without the encrypted SNI [tls-esni]. Also, to obtain IP address(es) corresponding to a target URL, the DNS name resolution process can be observed through a packet capturing tool because the DNS name resolution will translate the target URL into IP address(es). The IP addresses obtained through either TLS or DNS can be used by both firewall and web filter for whitelisting or blacklisting the TCP five-tuples of HTTP sessions.

Finally, the Security Controller sends the low-level security policy of the IP address and port number inspection to the firewall NSF and the low-level security policy for URL inspection to the web filter NSF.

The following describes how the time-dependent web access control service is enforced by the NSFs of firewall and web filter.

1. A staff member tries to access one of the target SNS websites (i.e., example1.com, example2.com, example3.com, and example4.com) during business hours, e.g., 10 AM.

2. The packet is forwarded from the staff member's device to the firewall, and the firewall checks the source IP address and port number. Now the firewall identifies the received packet is an HTTP-session packet from the staff member.

3. The firewall triggers the web filter to further inspect the packet, and the packet is forwarded from the firewall to the web filter. The SFC architecture [RFC7665] can be utilized to support such packet forwarding in the I2NSF framework.

4. The web filter checks the received packet's target URL field or its destination IP address corresponding to the target URL, and detects that the packet is being sent to the server for example1.com. The web filter then checks that the current time is within business hours. If so, the web filter drops the packet, and consequently the staff member's access to one of the SNS websites (i.e., example1.com, example2.com, example3.com, and example4.com) during business hours is blocked.

```
            +------------------------+-------------------------+
            |                                                  |
            |                    I2NSF User                    |
            |                                                  |
            +------------------------+-------------------------+
                                     | Consumer-Facing Interface
                                     |
                         High-level Security Policy
            Security                 |
            Controller               V
            +------------------------+-------------------------+
            |  Security Policy       |                         |
            |  Translator            |                         |
            |  +---------------------+----------------------+  |
            |  |                     |                      |  |
            |  |             +-------+--------+             |  |
            |  |             | Data Extractor |             |  |
            |  |             +-------+--------+             |  |
            |  |                     |  Extracted Data from |  |
            |  |                     V  High-level Policy   |  |
            |  |             +-------+--------+    +------+ |  |
            |  |             | Data Converter |<-->|NSF DB| |  |
            |  |             +-------+--------+    +------+ |  |
            |  |                     |  Required Data for   |  |
            |  |                     V  Target NSFs         |  |
            |  |             +-------+--------+             |  |
            |  |             |Policy Generator|             |  |
            |  |             +-------+--------+             |  |
            |  |                     |                      |  |
            |  +---------------------+----------------------+  |
            |                        |                         |
            +------------------------+-------------------------+
                                     |  NSF-Facing Interface
                                     |
                          Low-level Security Policy
                                     |
                                     V
            +------------------------+-------------------------+
            |                                                  |
            |                      NSF(s)                      |
            |                                                  |
            +------------------------+-------------------------+
```

Figure 4: Security Policy Translation and Enforcement in I2NSF System

5. Intent-based Security Services

I2NSF aims at providing intent-based security services to configure specific security policies into NSFs with customer-friendly secuirty policies at a high level. For example, when an I2NSF User submits a high-level security policy (e.g., web filtering as shown in Figure 2) to the Security Controller, the Security Policy Tranlator (SPT) in the Security Controller will translate it into the correspondong low-level security policy as shown in Figure 3 [policy-translation]. A security administrator using the I2NSF User can describe a security policy without the knowledge of the detailed information about subjects (e.g., source and destination) and objects (e.g., web traffic) of the security policy's rule(s).

Figure 4 shows the security policy translation and enforcement in the I2NSF system [policy-translation]. As shown in Figure 4, an I2NSF User delivers a high-level security policy to the Security Controller using the Consumer-Facing Interface (denoted as CFI). The high-level security policy is translated by the SPT in the Security Controller into the corresponding low-level security policy which is understandable by target NSF(s). The Security Controller delivers the low-level security policy to the appropriate NSF(s) to enforce the policy's rules.

The SPT consists of three modules for security policy translations such as Data Extractor, Data Converter, and Policy Generator, as shown in Figure 4. The Data Extractor extracts data from a high-level security policy delivered by the I2NSF User. The data correspond to the leaf nodes in the YANG data model for the Consumer-Facing Interface. In the high-level policy in Figure 2, the data are the tag values of policy-name, rule-name, begin-time, end-time, src-target, dest-target, and primary-action. That is, the tag values are "block_website", "block_website_during_working_hours", "09:00", "18:00", "Staff_Members'_PCs", "SNS_Websites", and "drop."

The Data Converter converts the extracted high-level policy data received from the Data Extractor into the corresponding low-level policy data. The low-level policy data have the capability information of NSFs to be selected as target NSFs for the required security service enforcement specified by the high-level security policy. The tag values in the extracted high-level policy data are replaced with the tag values in the low-level policy data, which are the leaf nodes of the YANG data model for the NSF-Facing Interface (denoted as NFI). The value of each leaf node in CFI is translated into the value of the corresponding leaf node in NFI. For example, "block_website" of policy-name in CFI (in Figure 2) is translated into "block_website" of system-policy-name in NFI (in Figure 3). The tag values of rule-name, begin-time, end-time, and primary-action in

CFI are mapped into the same values of rule-name, begin-time, end-time, and egress-action in NFI. However, the tag values of src-target and dest-target in CFI are translated into IP addresses and URLs, respectively, for the sake of NFI. That is, "Staff_Members'_PCs" of CFI is translated into three IPv6 addresses such as "2001:DB8:10:1::10", "2001:DB8:10:1::20", and "2001:DB8:10:1::30" for the sake of NFI. Also, "SNS_Websites" of CFI is translated into four URLs such as "example1.com", "example2.com", "example3.com", and "example4.com" for the sake of NFI. In addition to the data conversion, the Data Converter searches for appropriate NSFs having capabilities corresponding to the leaf nodes of the YANG data model for NFI. For the data conversion and NSF search, an NSF database (denoted as NSF DB) can be consulted, as shown in Figure 4, because the NSF DB has the capability information of NSFs that the DMS(s) registered with the Security Controller using the Registration Interface.

The Policy Generator generates a low-level security policy corresponding to the low-level policy data made by the Data Converter per a target NSF. That is, the Policy Generator can build such a low-level security policy XML file like Figure 3 with the NSF DB because the NSF DB has the mapping information between the CFI YANG data model and the NFI YANG data model.

Therefore, by allowing the I2NSF User to express its security policy without knowing the detailed information of entities for security policies, the I2NSF can efficiently support the intent-based security services with the help of the security policy translator along with the NSF DB.

```
      +------------+
      | I2NSF User |
      +------------+
             ^
             | Consumer-Facing Interface
             v
   +-------------------+     Registration     +-----------------------+
   |Security Controller|<-------------------->|Developer's Mgmt System|
   +-------------------+      Interface       +-----------------------+
         ^       ^
         |       | NSF-Facing Interface
         |       |-------------------------
         |                                |
         | NSF-Facing Interface           |
   +-----v-----------+             +------v-------+
   |  +-----------+  |      ------>|    NSF-1     |
   |  |Classifier |  |      |      |  (Firewall)  |
   |  +-----------+  |      |      +--------------+
   |     +-----+     |<-----|      +--------------+
   |     | SFF |     |      |----->|    NSF-2     |
   |     +-----+     |      |      |    (DPI)     |
   +-----------------+      |      +--------------+
                            |             .
                            |             .
                            |             .
                            |      +-----------------------+
                            ------>|        NSF-n          |
                                   |(DDoS-Attack Mitigator)|
                                   +-----------------------+
```

Figure 5: An I2NSF Framework with SFC

6. I2NSF Framework with SFC

In the I2NSF architecture, an NSF can trigger an advanced security action (e.g., DPI or DDoS attack mitigation) on a packet based on the result of its own security inspection of the packet. For example, a firewall triggers further inspection of a suspicious packet with DPI. For this advanced security action to be fulfilled, the suspicious packet should be forwarded from the current NSF to the successor NSF. SFC [RFC7665] is a technology that enables this advanced security action by steering a packet with multiple service functions (e.g., NSFs), and this technology can be utilized by the I2NSF architecture to support the advanced security action.

Figure 5 shows an I2NSF framework with the support of SFC. As shown in the figure, SFC generally requires classifiers and service function forwarders (SFFs); classifiers are responsible for determining which service function path (SFP) (i.e., an ordered sequence of service functions) a given packet should pass through, according to pre-configured classification rules, and SFFs perform forwarding the given packet to the next service function (e.g., NSF) on the SFP of the packet by referring to their forwarding tables. In the I2NSF architecture with SFC, the Security Controller can take responsibilities of generating classification rules for classifiers and forwarding tables for SFFs. By analyzing high-level security policies from I2NSF users, the Security Controller can construct SFPs that are required to meet the high-level security policies, generates classification rules of the SFPs, and then configures classifiers with the classification rules over NSF-Facing Interface so that relevant traffic packets can follow the SFPs. Also, based on the global view of NSF instances available in the system, the Security Controller constructs forwarding tables, which are required for SFFs to forward a given packet to the next NSF over the SFP, and configures SFFs with those forwarding tables over NSF-Facing Interface.

To trigger an advanced security action in the I2NSF architecture, the current NSF appends metadata describing the security capability required to the suspicious packet via a network service header (NSH) [RFC8300]. It then sends the packet to the classifier. Based on the metadata information, the classifier searches an SFP which includes an NSF with the required security capability, changes the SFP-related information (e.g., service path identifier and service index [RFC8300]) of the packet with the new SFP that has been found, and then forwards the packet to the SFF. When receiving the packet, the SFF checks the SFP-related information such as the service path identifier and service index contained in the packet and forwards the packet to the next NSF on the SFP of the packet, according to its forwarding table.

```
      +------------+
      | I2NSF User |
      +------------+
             ^
             | Consumer-Facing Interface
             v
   +-------------------+     Registration     +-----------------------+
   |Security Controller|<-------------------->|Developer's Mgmt System|
   +-------------------+      Interface       +-----------------------+
      ^     ^
      |     | NSF-Facing Interface
      |     v
      | +----------------+ +---------------+   +-----------------------+
      | |     NSF-1      |-|     NSF-2     |...|         NSF-n         |
      | |   (Firewall)   | |     (DPI)     |   |(DDoS-Attack Mitigator)|
      | +----------------+ +---------------+   +-----------------------+
      |
      |
      |                                                     SDN Network
   +--|---------------------------------------------------------------+
   |  V NSF-Facing Interface                                          |
   |  +-----------------+                                             |
   |  | SDN Controller  |                                             |
   |  +-----------------+                                             |
   |           ^                                                      |
   |           | SDN Southbound Interface                             |
   |           v                                                      |
   |      +---------+ +-------------+ +---------+       +---------+   |
   |      |Switch-1 |-|  Switch-2   |-|Switch-3 |.......|Switch-m |   |
   |      |         | |(Classifier) | |  (SFF)  |       |         |   |
   |      +---------+ +-------------+ +---------+       +---------+   |
   +------------------------------------------------------------------+
```

Figure 6: An I2NSF Framework with SDN Network

7. I2NSF Framework with SDN

This section describes an I2NSF framework with SDN for I2NSF applicability and use cases, such as firewall, deep packet inspection, and DDoS-attack mitigation functions. SDN enables some packet filtering rules to be enforced in network forwarding elements (e.g., switch) by controlling their packet forwarding rules. By taking advantage of this capability of SDN, it is possible to optimize the process of security service enforcement in the I2NSF system. For example, for efficient firewall services, simple packet filtering can be performed by SDN forwarding elements (e.g., switches), and complicated packet filtering based on packet payloads can be performed by a firewall NSF. This optimized firewall using both SDN forwarding elements and a firewall NSF is more efficient than a firewall where SDN forwarding elements forward all the packets to a firewall NSF for packet filtering. This is because packets to be filtered out can be early dropped by SDN forwarding elements without consuming further network bandwidth due to the forwarding of the packets to the firewall NSF.

Figure 6 shows an I2NSF framework [RFC8329] with SDN networks to support network-based security services. In this system, the enforcement of security policy rules is divided into the SDN forwarding elements (e.g., a switch running as either a hardware middle box or a software virtual switch) and NSFs (e.g., a firewall running in a form of a VNF [ETSI-NFV]). Note that NSFs are created or removed by the NFV Management and Orchestration (MANO) [ETSI-NFV-MANO], performing the lifecycle management of NSFs as VNFs. Refer to Section 8 for the detailed discussion of the NSF lifecycle management in the NFV MANO for I2NSF. For security policy enforcement (e.g., packet filtering), the Security Controller instructs the SDN Controller via NSF-Facing Interface so that SDN forwarding elements can perform the required security services with flow tables under the supervision of the SDN Controller.

As an example, let us consider two different types of security rules: Rule A is a simple packet filtering rule that checks only the IP address and port number of a given packet, whereas rule B is a time-consuming packet inspection rule for analyzing whether an attached file being transmitted over a flow of packets contains malware. Rule A can be translated into packet forwarding rules of SDN forwarding elements and thus be enforced by these elements. In contrast, rule B cannot be enforced by forwarding elements, but it has to be enforced by NSFs with anti-malware capability. Specifically, a flow of packets is forwarded to and reassembled by an NSF to reconstruct the attached file stored in the flow of packets. The NSF then analyzes the file to check the existence of malware. If the file contains malware, the NSF drops the packets.

In an I2NSF framework with SDN, the Security Controller can analyze given security policy rules and automatically determine which of the given security policy rules should be enforced by SDN forwarding elements and which should be enforced by NSFs. If some of the given rules requires security capabilities that can be provided by SDN forwarding elements, then the Security Controller instructs the SDN Controller via NSF-Facing Interface so that SDN forwarding elements can enforce those security policy rules with flow tables under the supervision of the SDN Controller. Or if some rules require security capabilities that cannot be provided by SDN forwarding elements but by NSFs, then the Security Controller instructs relevant NSFs to enforce those rules.

The distinction between software-based SDN forwarding elements and NSFs, which can both run as VNFs, may be necessary for some management purposes in this system. Note that an SDN forwarding element (i.e., switch) is a specific type of VNF rather than an NSF because an NSF is for security services rather than for packet forwarding. For this distinction, we can take advantage of the NFV MANO where there is a subsystem that maintains the descriptions of the capabilities each VNF can offer [ETSI-NFV-MANO]. This subsystem can determine whether a given software element (VNF instance) is an NSF or a virtualized SDN switch. For example, if a VNF instance has anti-malware capability according to the description of the VNF, it could be considered as an NSF. A VNF onboarding system [VNF-ONBOARDING] can be used as such a subsystem that maintains the descriptions of each VNF to tell whether a VNF instance is for an NSF or for a virtualized SDN switch.

For the support of SFC in the I2NSF framework with SDN, as shown in Figure 6, network forwarding elements (e.g., switch) can play the role of either SFC Classifier or SFF, which are explained in Section 6. Classifier and SFF have an NSF-Facing Interface with Security Controller. This interface is used to update security service function chaining information for traffic flows. For example, when it needs to update an SFP for a traffic flow in an SDN network, as shown in Figure 6, SFF (denoted as Switch-3) asks Security Controller to update the SFP for the traffic flow (needing another security service as an NSF) via NSF-Facing Interface. This update lets Security Controller ask Classifier (denoted as Switch-2) to update the mapping between the traffic flow and SFP in Classifier via NSF-Facing Interface.

The following subsections introduce three use cases from [RFC8192] for cloud-based security services: (i) firewall system, (ii) deep packet inspection system, and (iii) attack mitigation system.

7.1. Firewall: Centralized Firewall System

A centralized network firewall can manage each network resource and apply common rules to individual network elements (e.g., switch). The centralized network firewall controls each forwarding element, and firewall rules can be added or deleted dynamically.

A time-based firewall can be enforced with packet filtering rules and a time span (e.g., work hours). With this time-based firewall, a time-based security policy can be enforced, as explained in Section 4. For example, employees at a company are allowed to access social networking service websites during lunch time or after work hours.

7.2. Deep Packet Inspection: Centralized VoIP/VoLTE Security System

A centralized VoIP/VoLTE security system can monitor each VoIP/VoLTE flow and manage VoIP/VoLTE security rules, according to the configuration of a VoIP/VoLTE security service called VoIP Intrusion Prevention System (IPS). This centralized VoIP/VoLTE security system controls each switch for the VoIP/VoLTE call flow management by manipulating the rules that can be added, deleted or modified dynamically.

The centralized VoIP/VoLTE security system can cooperate with a network firewall to realize VoIP/VoLTE security service. Specifically, a network firewall performs the basic security check of an unknown flow's packet observed by a switch. If the network firewall detects that the packet is an unknown VoIP call flow's packet that exhibits some suspicious patterns, then it triggers the VoIP/VoLTE security system for more specialized security analysis of the suspicious VoIP call packet.

7.3. Attack Mitigation: Centralized DDoS-attack Mitigation System

A centralized DDoS-attack mitigation can manage each network resource and configure rules to each switch for DDoS-attack mitigation (called DDoS-attack Mitigator) on a common server. The centralized DDoS-attack mitigation system defends servers against DDoS attacks outside the private network, that is, from public networks [RFC8612][dots-architecture].

Servers are categorized into stateless servers (e.g., DNS servers) and stateful servers (e.g., web servers). For DDoS-attack mitigation, the forwarding of traffic flows in switches can be dynamically configured such that malicious traffic flows are handled by the paths separated from normal traffic flows in order to minimize the impact of those malicious traffic on the servers. This flow path separation can be done by a flow forwarding path management scheme [dots-architecture][AVANT-GUARD]. This management should consider the load balance among the switches for the defense against DDoS attacks.

So far this section has described the three use cases for network-based security services using the I2NSF framework with SDN networks. To support these use cases in the proposed data-driven security service framework, YANG data models described in [consumer-facing-inf-dm], [nsf-facing-inf-dm], and [registration-inf-dm] can be used as Consumer-Facing Interface, NSF-Facing Interface, and Registration Interface, respectively, along with RESTCONF [RFC8040] and NETCONF [RFC6241].

```
                                                  +--------------------+
   +-------------------------------------------+  | ----------------   |
   |            I2NSF User (OSS/BSS)           |  | | NFV          |   |
   +------+------------------------------------+  | | Orchestrator +-+ |
          |  Consumer-Facing Interface            |  -----+---------- | |
   +------|------------------------------------+  |      |           | |
   | -----+----------  (a)  -----------------  |  |  ----+-----      | |
   | |  Security    +-------+  Developer's  |  |  |  |        |      | |
   | |Controller(EM)|       |Mgmt System(EM)|  +-(b)-+ VNFM(s)|      | |
   | -----+----------       -----------------  |  |  |        |      | |
   |      |  NSF-Facing Interface              |  |  ----+-----      | |
   |  ----+-----    ----+-----    ----+-----   |  |      |           | |
   |  |NSF(VNF)|    |NSF(VNF)|    |NSF(VNF)|   |  |      |           | |
   |  ----+-----    ----+-----    ----+-----   |  |      |           | |
   |      |             |             |        |  |      |           | |
   +------|-------------|-------------|--------+  |      |           | |
          |             |             |           |      |           | |
   +------+-------------+-------------+--------+  |      |           | |
   |         NFV Infrastructure (NFVI)         |  |      |           | |
   | -----------    -----------    ----------- |  |      |           | |
   | | Virtual |    | Virtual |    | Virtual | |  |      |           | |
   | | Compute |    | Storage |    | Network | |  |      |           | |
   | -----------    -----------    ----------- |  |  ----+-----      | |
   | +---------------------------------------+ |  |  |        |      | |
   | |         Virtualization Layer          | +-----+ VIM(s) +------+ |
   | +---------------------------------------+ |  |  |        |        |
   | +---------------------------------------+ |  |  ----------        |
   | | -----------  -----------  ----------- | |  |                    |
   | | | Compute |  | Storage |  | Network | | |  |                    |
   | | | Hardware|  | Hardware|  | Hardware| | |  |                    |
   | | -----------  -----------  ----------- | |  |                    |
   | |          Hardware Resources           | |  |   NFV Management   |
   | +---------------------------------------+ |  | and Orchestration  |
   |                                           |  |       (MANO)       |
   +-------------------------------------------+  +--------------------+
   (a) = Registration Interface
   (b) = Ve-Vnfm Interface
```

Figure 7: I2NSF Framework Implementation with respect to the NFV Reference Architectural Framework

8. I2NSF Framework with NFV

This section discusses the implementation of the I2NSF framework using Network Functions Virtualization (NFV).

NFV is a promising technology for improving the elasticity and efficiency of network resource utilization. In NFV environments, NSFs can be deployed in the forms of software-based virtual instances rather than physical appliances. Virtualizing NSFs makes it possible to rapidly and flexibly respond to the amount of service requests by dynamically increasing or decreasing the number of NSF instances. Moreover, NFV technology facilitates flexibly including or excluding NSFs from multiple security solution vendors according to the changes on security requirements. In order to take advantages of the NFV technology, the I2NSF framework can be implemented on top of an NFV infrastructure as show in Figure 7.

Figure 7 shows an I2NSF framework implementation based on the NFV reference architecture that the European Telecommunications Standards Institute (ETSI) defines [ETSI-NFV]. The NSFs are deployed as VNFs in Figure 7. The Developer's Management System (DMS) in the I2NSF framework is responsible for registering capability information of NSFs into the Security Controller. However, those NSFs are created or removed by a virtual network function manager (VNFM) in the NFV MANO that performs the lifecycle management of VNFs. Note that the lifecycle management of VNFs is out of scope for I2NSF. The Security Controller controls and monitors the configurations (e.g., function parameters and security policy rules) of VNFs via the NSF-Facing Interface along with the NSF monitoring capability [nsf-facing-inf-dm][nsf-monitoring-dm]. Both the DMS and Security Controller can be implemented as the Element Managements (EMs) in the NFV architecture. Finally, the I2NSF User can be implemented as OSS/BSS (Operational Support Systems/Business Support Systems) in the NFV architecture that provides interfaces for users in the NFV system.

The operation procedure in the I2NSF framework based on the NFV architecture is as follows:

1. The VNFM has a set of virtual machine (VM) images of NSFs, and each VM image can be used to create an NSF instance that provides a set of security capabilities. The DMS initially registers a mapping table of the ID of each VM image and the set of capabilities that can be provided by an NSF instance created from the VM image into the Security Controller.

2. If the Security Controller does not have any instantiated NSF that has the set of capabilities required to meet the security requirements from users, it searches the mapping table (registered by the DMS) for the VM image ID corresponding to the required set of capabilities.

3. The Security Controller requests the DMS to instantiate an NSF with the VM image ID via VNFM.

4. When receiving the instantiation request, the VNFM first asks the NFV orchestrator for the permission required to create the NSF instance, requests the VIM to allocate resources for the NSF instance, and finally creates the NSF instance based on the allocated resources.

5. Once the NSF instance has been created by the VNFM, the DMS performs the initial configurations of the NSF instance and then notifies the Security Controller of the NSF instance.

6. After being notified of the created NSF instance, the Security Controller delivers low-level security policy rules to the NSF instance for policy enforcement.

We can conclude that the I2NSF framework can be implemented based on the NFV architecture framework. Note that the registration of the capabilities of NSFs is performed through the Registration Interface and the lifecycle management for NSFs (VNFs) is performed through the Ve-Vnfm interface between the DMS and VNFM, as shown in Figure 7.

9. Security Considerations

The same security considerations for the I2NSF framework [RFC8329] are applicable to this document.

This document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300].

The role of the DMS is to provide an I2NSF system with the software packages or images for NSF execution. The DMS must not access NSFs in activated status. An inside attacker or a supply chain attacker at the DMS can seriously weaken the I2NSF system's security. A malicious DMS is relevant to an insider attack, and a compromised DMS is relevant to a supply chain attack. A malicious (or compromised) DMS could register an NSF of its choice in response to a capability request by the Security Controller. As a result, a malicious DMS can attack the I2NSF system by providing malicious NSFs with arbitrary capabilities to include potentially controlling those NSFs in real time. An unwitting DMS could be compromised and the infrastructure of the DMS could be coerced into distributing modified NSFs as well.

To deal with these types of threats, an I2NSF system should not use NSFs from an untrusted DMS or without prior testing. The practices by which these packages are downloaded and loaded into the system are out of scope for I2NSF.

I2NSF system operators should audit and monitor interactions with DMSs. Additionally, the operators should monitor the running NSFs through the I2NSF NSF Monitoring Interface [nsf-monitoring-dm] as part of the I2NSF NSF-Facing Interface. Note that the mechanics for monitoring the DMSs are out of scope for I2NSF.

10. Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea MSIT (Ministry of Science and ICT) (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This work has been partially supported by the European Commission under Horizon 2020 grant agreement no. 700199 "Securing against intruders and other threats through a NFV-enabled environment (SHIELD)". This support does not imply endorsement.

11. Contributors

I2NSF is a group effort. I2NSF has had a number of contributing authors. The following are considered co-authors:

- Hyoungshick Kim (Sungkyunkwan University)
- Jinyong Tim Kim (Sungkyunkwan University)
- Hyunsik Yang (Soongsil University)
- Younghan Kim (Soongsil University)
- Jung-Soo Park (ETRI)
- Se-Hui Lee (Korea Telecom)
- Mohamed Boucadair (Orange)

12. References

12.1. Normative References

[AVANT-GUARD]
Shin, S., Yegneswaran, V., Porras, P., and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks", ACM CCS, November 2013.

[consumer-facing-inf-dm]
Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", draft-ietf-i2nsf-consumer-facing-interface-dm-06 (work in progress), July 2019.

[dots-architecture]
Mortensen, A., Reddy, T., Andreasen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-14 (work in progress), May 2019.

[ETSI-NFV]
"Network Functions Virtualisation (NFV); Architectural Framework", Available: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf, October 2013.

[ITU-T.Y.3300]
"Framework of Software-Defined Networking", Available: https://www.itu.int/rec/T-REC-Y.3300-201406-I, June 2014.

[NFV-Terminology]
"Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV", Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf, December 2014.

[nsf-facing-inf-dm]
Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-07 (work in progress), July 2019.

[nsf-monitoring-dm]
Jeong, J., Chung, C., Hares, S., Xia, L., and H. Birkholz, "I2NSF NSF Monitoring YANG Data Model", draft-ietf-i2nsf-nsf-monitoring-data-model-01 (work in progress), July 2019.

[ONF-SDN-Architecture]
"SDN Architecture (Issue 1.1)", Available: https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf, June 2016.

[registration-inf-dm]
Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-ietf-i2nsf-registration-interface-dm-05 (work in progress), July 2019.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.

[RFC7665] Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, July 2017.

[RFC8300] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", RFC 8300, January 2018.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

[RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open Threat Signaling (DOTS) Requirements", RFC 8612, May 2019.

12.2. Informative References

[ETSI-NFV-MANO]
"Network Functions Virtualisation (NFV); Management and Orchestration", Available: https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf, December 2014.

[i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-08 (work in progress), July 2019.

[ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[opsawg-firewalls]
Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.

[policy-translation]
Jeong, J., Yang, J., Chung, C., and J. Kim, "Security Policy Translation in Interface to Network Security Functions", draft-yang-i2nsf-security-policy-translation-04 (work in progress), July 2019.

[tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-04 (work in progress), July 2019.

[VNF-ONBOARDING]
"VNF Onboarding", Available: https://wiki.opnfv.org/display/mano/VNF+Onboarding, November 2016.

Appendix A. Changes from draft-ietf-i2nsf-applicability-17

The following changes have been made from draft-ietf-i2nsf-applicability-17:

- In Section 4, a high-level security policy XML file in Figure 2 and the corresponding low-level security policy XML file Figure 3 are constructed using the Consumer-Facing Interface data model and the NSF-Facing data model, respectively.

- For the applicability of I2NSF to the real world, Section 5 is added to support the Intent-based Security Services using I2NSF. This section explains the security policy translation based on an I2NSF User's intents on the required security services. Figure 4 shows the archiecture and procedure of the I2NSF security policy translator.

Authors' Addresses

Jaehoon Paul Jeong
Department of Computer Science and Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Sangwon Hyun
Department of Computer Engineering
Myongji University
116 Myongji-ro, Cheoin-gu
Yongin 17058
Republic of Korea

Phone: +82 62 230 7473
EMail: shyun@chosun.ac.kr

Tae-Jin Ahn
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8409
EMail: taejin.ahn@kt.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
EMail: shares@ndzh.com

Diego R. Lopez
Telefonica I+D
Jose Manuel Lara, 9
Seville 41013
Spain

Phone: +34 682 051 091
EMail: diego.r.lopez@telefonica.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

J. Jeong
E. Kim
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei

March 5, 2018

I2NSF Consumer-Facing Interface YANG Data Model
draft-ietf-i2nsf-consumer-facing-interface-dm-00

Abstract

This document describes a YANG data model for the Consumer-Facing Interface between an Interface to Network Security Functions (I2NSF) User and Security Controller in an I2NSF system in a Network Functions Virtualization (NFV) environment. The data model is required for enabling different users of a given I2NSF system to define, manage, and monitor security policies for specific flows within an administrative domain.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document provides a YANG [RFC6020] data model that defines the required data for the Consumer-Facing Interface between an Interface to Network Security Functions (I2NSF) User and Security Controller in an I2NSF system [i2nsf-framework] in a Network Functions Virtualization (NFV) environment. The data model is required for enabling different users of a given I2NSF system to define, manage and monitor security policies for specific flows within an administrative domain. This document defines a YANG data model based on the information model of I2NSF Consumer-Facing Interface [client-facing-inf-im].

Data models are defined at a lower level of abstraction and provide many details. They provide details about the implementation of a protocol's specification, e.g., rules that explain how to map managed objects onto lower-level protocol constructs. Since conceptual models can be implemented in different ways, multiple data models can be derived by a single information model.

The efficient and flexible provisioning of network functions by NFV leads to a rapid advance in the network industry. As practical applications, network security functions (NSFs), such as firewall, intrusion detection system (IDS)/intrusion protection system (IPS), and attack mitigation, can also be provided as virtual network functions (VNF) in the NFV system. By the efficient virtual technology, these VNFs might be automatically provisioned and dynamically migrated based on real-time security requirements. This document presents a YANG data model to implement security functions based on NFV.

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 RFC 2119 [RFC3444].

3. Terminology

This document uses the terminology described in [i2nsf-terminology][client-facing-inf-im][client-facing-inf-req].

4. Data Modeling for Security Policies for Consumer-Facing Interface

The main objective of this data model is to fully transform the information model [client-facing-inf-im] into a YANG data model that can be used for delivering control and management messages via the Consumer-Facing Interface between an I2NSF User and Security Controller for the I2NSF User's high-level security policies.

The semantics of the data model must be aligned with the information model of the Consumer-Facing Interface. The transformation of the information model was performed so that this YANG data model can facilitate the efficient delivery of the control or management messages.

This data model is designed to support the I2NSF framework that can be extended according to the security needs. In other words, the model design is independent of the content and meaning of specific policies as well as the implementation approach. This document suggests a VoIP/VoLTE security service as a use case for policy rule generation.

Multi-tenancy in this document enables multiple administrative domains in order to manage application resources. An Enterprise organization may have multiple tenants or departments such as HR, finance, and legal. Thus, we need an object which defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. You can think of it as a way to assign policy users to a job function or a set of permissions within the organization. The policy-role object SHALL have Name, Date and access-profile to grant or deny permissions for the perpose of security policy management.

```
module: policy-general
    +--rw policy
    |  +--rw rule* [rule-id]
    |     +--rw rule-id                      uint16
    |     +--rw name?                        string
    |     +--rw date?                        yang:date-and-time
    |     +--rw case?                        string
    |     +--rw event* [event-id]
    |     |  +--rw event-id                  string
    |     |  +--rw name?                     string
    |     |  +--rw date?                     yang:date-and-time
    |     |  +--rw event-type?               string
    |     |  +--rw time-information?         string
    |     |  +--rw event-map-group?    -> /threat-feed/event-map-group
    |     |                                 /event-map-group-id
    |     |  +--rw enable?                   boolean
    |     +--rw condition* [condition-id]
    |     |  +--rw condition-id              string
    |     |  +--rw source?                   string
    |     |  +--rw destination?              string
    |     |  +--rw match?                    boolean
    |     |  +--rw match-direction?          string
    |     |  +--rw exception?                string
    |     +--rw policy-action* [policy-action-id]
    |        +--rw policy-action-id          string
    |        +--rw name?                     string
    |        +--rw date?                     yang:date-and-time
    |        +--rw primary-action?           string
    |        +--rw secondary-action?         string
    |        +--rw owner?                    string
    +--rw multi-tenancy
    |  +--rw policy-domain* [policy-domain-id]
    |  |  +--rw policy-domain-id             uint16
    |  |  +--rw name                         string
```

```
   |  |  +--rw address?                     string
   |  |  +--rw contact                      string
   |  |  +--rw date                         yang:date-and-time
   |  |  +--rw policy-tenant* [policy-tenant-id]
   |  |  |  +--rw policy-tenant-id         uint16
   |  |  |  +--rw name                     string
   |  |  |  +--rw date                     yang:date-and-time
   |  |  |  +--rw domain?              ->  /multi-tenancy
   |  |  |                                 /policy-domain
   |  |  |                                 /policy-domain-id
   |  |  +--rw authentication-method?   ->  /multi-tenancy
   |  |                                    /policy-mgnt-auth-method
   |  |                                    /policy-mgnt-auth-method-id
   |  +--rw policy-role* [policy-role-id]
   |  |  +--rw policy-role-id               uint16
   |  |  +--rw name                         string
   |  |  +--rw date                         yang:date-and-time
   |  |  +--rw access-profile               string
   |  +--rw policy-user* [policy-user-id]
   |  |  +--rw policy-user-id               uint16
   |  |  +--rw name                         string
   |  |  +--rw date                         yang:date-and-time
   |  |  +--rw password                     string
   |  |  +--rw email                        string
   |  |  +--rw scope-type?                  string
   |  |  +--rw scope-reference?             string
   |  |  +--rw role                         string
   |  +--rw policy-mgnt-auth-method* [policy-mgnt-auth-method-id]
   |     +--rw policy-mgnt-auth-method-id   uint16
   |     +--rw name                         string
   |     +--rw date                         yang:date-and-time
   |     +--rw authentication-method        enumeration
   |     +--rw mutual-authentication        boolean
   |     +--rw token-server                 inet:ipv4-address
   |     +--rw certificate-server           inet:ipv4-address
   |     +--rw single-sing-on-server        inet:ipv4-address
   +--rw end-group
   |  +--rw meta-data-source* [meta-data-source-id]
   |  |  +--rw meta-data-source-id          uint16
   |  |  +--rw name                         string
   |  |  +--rw date                         yang:date-and-time
   |  |  +--rw tag-type?                    boolean
   |  |  +--rw tag-server-information?      inet:ipv4-address
   |  |  +--rw tag-application-protocol?    string
   |  |  +--rw tag-server-credential?       string
   |  +--rw user-group* [user-group-id]
   |  |  +--rw user-group-id                uint16
   |  |  +--rw name?                        string
```

```
   |  |  +--rw date?                        yang:date-and-time
   |  |  +--rw group-type?                  enumeration
   |  |  +--rw meta-data-server?            inet:ipv4-address
   |  |  +--rw group-member?                string
   |  |  +--rw risk-level?                  uint16
   |  +--rw device-group* [device-group-id]
   |  |  +--rw device-group-id              uint16
   |  |  +--rw name?                        string
   |  |  +--rw date?                        yang:date-and-time
   |  |  +--rw group-type?                  enumeration
   |  |  +--rw meta-data-server?            inet:ipv4-address
   |  |  +--rw group-member?                string
   |  |  +--rw risk-level?                  uint16
   |  +--rw application-group* [application-group-id]
   |  |  +--rw application-group-id         uint16
   |  |  +--rw name?                        string
   |  |  +--rw date?                        yang:date-and-time
   |  |  +--rw group-type?                  enumeration
   |  |  +--rw meta-data-server?            inet:ipv4-address
   |  |  +--rw group-member?                string
   |  |  +--rw risk-level?                  uint16
   |  +--rw location-group* [location-group-id]
   |     +--rw location-group-id            uint16
   |     +--rw name?                        string
   |     +--rw date?                        yang:date-and-time
   |     +--rw group-type?                  enumeration
   |     +--rw meta-data-server?            inet:ipv4-address
   |     +--rw group-member?                string
   |     +--rw risk-level?                  uint16
   +--rw threat-feed
   |  +--rw threat-feed* [threat-feed-id]
   |  |  +--rw threat-feed-id               uint16
   |  |  +--rw name?                        string
   |  |  +--rw date?                        yang:date-and-time
   |  |  +--rw feed-type                    enumeration
   |  |  +--rw feed-server?                 inet:ipv4-address
   |  |  +--rw feed-priority?               uint16
   |  +--rw custom-list* [custom-list-id]
   |  |  +--rw custom-list-id               uint16
   |  |  +--rw name?                        string
   |  |  +--rw date?                        yang:date-and-time
   |  |  +--rw list-type                    enumeration
   |  |  +--rw list-property                enumeration
   |  |  +--rw list-content?                string
   |  +--rw malware-scan-group* [malware-scan-group-id]
   |  |  +--rw malware-scan-group-id        uint16
   |  |  +--rw name?                        string
   |  |  +--rw date?                        yang:date-and-time
```

```
|  |  +--rw signature-server?            inet:ipv4-address
|  |  +--rw file-types?                  string
|  |  +--rw malware-signatures?          string
|  +--rw event-map-group* [event-map-group-id]
|     +--rw event-map-group-id           uint16
|     +--rw name?                        string
|     +--rw date?                        yang:date-and-time
|     +--rw security-events?             string
|     +--rw threat-map?                  string
+--rw telemetry-data
   +--rw telemetry-data* [telemetry-data-id]
   |  +--rw telemetry-data-id            uint16
   |  +--rw name?                        string
   |  +--rw date?                        yang:date-and-time
   |  +--rw logs?                        boolean
   |  +--rw syslogs?                     boolean
   |  +--rw snmp?                        boolean
   |  +--rw sflow?                       boolean
   |  +--rw netflow?                     boolean
   |  +--rw interface-stats?             boolean
   +--rw telemetry-source* [telemetry-source-id]
   |  +--rw telemetry-source-id          uint16
   |  +--rw name?                        string
   |  +--rw date?                        yang:date-and-time
   |  +--rw source-type?                 enumeration
   |  +--rw nsf-source?                  inet:ipv4-address
   |  +--rw nsf-credentials?             string
   |  +--rw collection-interval?         uint16
   |  +--rw collection-method?           enumeration
   |  +--rw heartbeat-interval?          uint16
   |  +--rw qos-marking?                 uint16
   +--rw telemetry-destination* [telemetry-destination-id]
      +--rw telemetry-destination-id     uint16
      +--rw name?                        string
      +--rw date?                        yang:date-and-time
      +--rw collector-source?            inet:ipv4-address
      +--rw collector-credentials?       string
      +--rw data-encoding?               string
      +--rw data-transport?              enumeration
```

Figure 1: Generic Data Model for Security Policies for cf Interface

5. YANG Data Model for Security Policies for Consumer-Facing Interface

This section describes a YANG data model for Consumer-Facing Interface, based on the information model of Consumer-Facing Interface to security controller [client-facing-inf-im].

```
<CODE BEGINS> file "policy-general.yang"
module ietf-policy-general {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-policy-general";
  prefix
    cf-interface;

  import ietf-yang-types{
    prefix yang;
  }

  import ietf-inet-types{
    prefix inet;
  }

  organization
    "IETF I2NSF (Interface to Network Security Functions)
     Working Group";

  contact
    "WG Web: <http://tools.ietf.org/wg/i2nsf>
     WG List: <mailto:i2nsf@ietf.org>

     WG Chair: Adrian Farrel
     <mailto:Adrain@olddog.co.uk>

     WG Chair: Linda Dunbar
     <mailto:Linda.duhbar@huawei.com>

     Editor: Jaehoon Paul Jeong
     <mailto:pauljeong@skku.edu>";

  description
    "This module defines a YANG data module for consumer-facing
     interface to security controller.";

  revision "2018-03-05"{
    description "fourth revision";
    reference
      "draft-kumar-i2nsf-client-facing-interface-im-04";
  }
```

```
  //Groupings
  container policy {
    description
    "This object is a policy instance to have
    complete information such as where and when
    a policy need to be applied.";

    list rule {
     key "rule-id";
     leaf rule-id {
     type uint16;
     description
     "This is ID for rules.";
    }
      description
      "This is a container for rules.";
      leaf name {
       type string;
        description
         "This field idenfifies the name of this object.";
      }

      leaf date {
       type yang:date-and-time;
        description
         "Date this object was created or last
          modified";
      }

      leaf case {
       type string;
        description
         "to identify whether the rule belongs to
         web filter or enterprise mode.";
      }

      list event {
       key "event-id";
       description
       "This represents the security event of a
       policy-rule.";

       leaf event-id {
       type string;
       mandatory true;
       description
       "This represents the event-id.";
       }
```

```
       leaf name {
        type string;
         description
          "This field idenfifies the name of this object.";
       }

       leaf date {
        type yang:date-and-time;
        description
         "Date this object was created or last
          modified";
       }

       leaf event-type {
        type string;
         description
          "This field identifies the event of
          policy enforcement trigger type.";
       }

       leaf time-information {
        type string;
        description
         "This field contains time calendar such as
         BEGIN-TIME and END-TIME for one time
         enforcement or recurring time calendar for
         periodic enforcement.";
       }

       leaf event-map-group {
        type leafref {
        path "/threat-feed/event-map-group/event-map-group-id";
        }
        description
        "This field contains security events or threat
        map in order to determine when a policy need
        to be activated. This is a reference to
        Evnet-Map-Group.";
       }

       leaf enable {
        type boolean;
         description
          "This determines whether the condition
          matches the security event or not.";
       }
     }
```

```
     list condition {
      key "condition-id";
      description
      "This represents the condition of a
      policy-rule.";

      leaf condition-id {
       type string;
       description
       "This represents the condition-id.";
      }

      leaf source {
       type string;
        description
        "This field identifies the source of
        the traffic. This could be reference to
        either 'Policy Endpoint Group' or
        'Threat-Feed' or 'Custom-List' if Security
        Admin wants to specify the source; otherwise,
        the default is to match all traffic.";
      }

      leaf destination {
       type string;
        description
         "This field identifies the source of
          the traffic. This could be reference to
          either 'Policy Endpoint Group' or
          'Threat-Feed' or 'Custom-List' if Security
          Admin wants to specify the source; otherwise,
          the default is to match all traffic.";
      }

      leaf match {
       type boolean;
        description
         "This field identifies the match criteria used to
         evaluate whether the specified action need to be
         taken or not.  This could be either a Policy-
         Endpoint-Group identifying a Application set or a
         set of traffic rules.";
      }

      leaf match-direction {
       type string;
        description
         "This field identifies if the match criteria is
```

```
          to evaluated for both direction of the traffic or
          only in one direction with default of allowing in
          the other direction for stateful match conditions.
          This is optional and by default rule should apply
          in both directions.";
       }

       leaf exception {
        type string;
         description
          "This field identifies the exception
          consideration when a rule is evaluated for a
          given communication.  This could be reference to
          Policy-Endpoint-Group object or set of traffic
          matching criteria.";
       }
      }

      list policy-action {
       key "policy-action-id";

       leaf policy-action-id {
       type string;
       mandatory true;
        description
         "this represents the policy-action-id.";
       }
       description
        "This object represents actions that a
        Security Admin wants to perform based on
        a certain traffic class.";

       leaf name {
        type string;
         description
          "The name of the policy-action object.";
       }

       leaf date {
        type yang:date-and-time;
         description
          "When the object was created or last
          modified.";
       }

       leaf primary-action {
        type string;
         description
```

```
         "This field identifies the action when a rule
         is matched by NSF. The action could be one of
         'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
         'AUTHENTICATE-SESSION', 'IPS, 'APP-FIREWALL', etc.";
      }

      leaf secondary-action {
       type string;
        description
         "This field identifies additional actions if
         a rule is matched. This could be one of 'LOG',
         'SYSLOG', 'SESSION-LOG', etc.";
      }


      leaf owner {
       type string;
        description
         "This field defines the owner of this
         policy. Only the owner is authorized to
         modify the contents of the policy.";
      }
     }
    }
 }

     container multi-tenancy {
        description
          "The descriptions of multi-tenancy.";

        list policy-domain {
          key "policy-domain-id";

          leaf policy-domain-id {
            type uint16;
            description
              "This represents the list of domains.";
          }
          description
          "this represent the list of policy domains";
          leaf name {
            type string;
            mandatory true;
            description
              "Name of the organization or customer representing
              this domain.";
          }
```

```
          leaf address {
            type string;
            description
              "address of an organization or customer.";
          }

          leaf contact {
            type string;
            mandatory true;
            description
              "contact information of the organization
               or customer.";
          }

          leaf date {
            type yang:date-and-time;
            mandatory true;
            description
              "The date when this account was created
              or last modified.";
          }
          list policy-tenant {
            key "policy-tenant-id";
            leaf policy-tenant-id {
              type uint16;
              description
                "The policy tenant id.";
            }
            description
            "This represents the list of tenants";

            leaf name {
              type string;
              mandatory true;
              description
                "Name of the Department or Division within
                 an organization.";
            }

            leaf date {
              type yang:date-and-time;
              mandatory true;
              description
                "Date this account was created or last modified.";
            }

            leaf domain {
              type leafref {
```

```
                path "/multi-tenancy/policy-domain/policy-domain-id";
              }
              description
              "This field identifies the domain to which this
              tenant belongs. This should be reference to a
              'Policy-Domain' object.";
            }
          }
          leaf authentication-method {
            type leafref {
              path "/multi-tenancy/policy-mgnt-auth-method/policy-mgnt-auth-meth
od-id";
            }

            description
            "Authentication method to be used for this domain.
            It should be a reference to a 'policy-mgmt-auth-method'
            object.";
          }
        }

        list policy-role {
          key "policy-role-id";

          leaf policy-role-id {
          type uint16;
          mandatory true;
          description
            "This defines a set of permissions assigned
            to a user in an organization that want to manage
            its own Security Policies.";
          }
          description
          "This represents the list of policy roles.";

          leaf name {
            type string;
            mandatory true;
            description
              "This field identifies name of the role.";
          }

          leaf date {
            type yang:date-and-time;
            mandatory true;
            description
              "Date this role was created or last modified.";
          }
```

```
          leaf access-profile {
            type string;
            mandatory true;
            description
              "This field identifies the access profile for the
              role. The profile grants or denies access to policy
              objects.  Multiple access profiles can be
              concatenated together.";
          }
        }

        list policy-user {
          key "policy-user-id";

          leaf policy-user-id {
          type uint16;
          description
            "This represents the policy-user-id.";
          }
          description
          "This represents the list of policy users.";
          leaf name {
            type string;
            mandatory true;
            description
              "The name of a user.";
          }

          leaf date {
            type yang:date-and-time;
            mandatory true;
            description
              "Date this user was created or last modified";
          }

          leaf password {
            type string;
            mandatory true;
            description
              "User password for basic authentication";
          }

          leaf email {
            type string;
            mandatory true;
            description
              "The email account of a user";
          }
```

```
          leaf scope-type {
            type string;
            description
              "identifies whether a user has domain-wide
              or tenant-wide privileges";
          }

          leaf scope-reference {
            type string;
            description
              "This references policy-domain or policy-tenant
              to identify the scope.";
          }

          leaf role {
            type string;
            mandatory true;
            description
              "This references policy-role to define specific
              permissions";
          }
        }

        list policy-mgnt-auth-method {
          key "policy-mgnt-auth-method-id";

          leaf policy-mgnt-auth-method-id {
          type uint16;
          description
            "This represents the authentication method id.";
          }
          description
          "The descriptions of policy management
            authentication methods.";
          leaf name {
            type string;
            mandatory true;
            description
              "name of the authentication method";
          }

          leaf date {
            type yang:date-and-time;
            mandatory true;
            description
              "date when the authentication method
              was created";
          }
```

```
          leaf authentication-method {
            type enumeration{
              enum password{
                description
                  "password-based authentication.";
              }
              enum token{
                description
                  "token-based authentication.";
              }
              enum certificate{
                description
                  "certificate-based authentication.";
              }
            }
            mandatory true;
            description
              "The description of authentication method;
              token-based, password, certificate,
              single-sign-on";
          }

          leaf mutual-authentication {
            type boolean;
            mandatory true;
            description
              "To identify whether the authentication
               is mutual";
          }

          leaf token-server {
            type inet:ipv4-address;
            mandatory true;
            description
              "The token-server information if the
              authentication method is token-based";
          }

          leaf certificate-server {
            type inet:ipv4-address;
            mandatory true;
            description
              "The certificate-server information if
              the authentication method is certificate-based";
          }

          leaf single-sing-on-server {
            type inet:ipv4-address;
```

```
            mandatory true;
            description
              "The single-sign-on-server information
              if the authentication method is
              single-sign-on-based";
          }
        }
     }
container end-group {
        description
          "A logical entity in their business
          environment, where a security policy
          is to be applied.";

        list meta-data-source {
          key "meta-data-source-id";
          leaf meta-data-source-id {
          type uint16;
          mandatory true;
          description
            "This represents the meta-data source id.";
          }
          description
          "This represents the meta-data source.";

          leaf name {
            type string;
            mandatory true;
            description
              "This identifies the name of the
              meta-datas-ource.";
          }

          leaf date {
            type yang:date-and-time;
            mandatory true;
            description
              "This identifies the date this object was
              created or last modified.";
          }

          leaf tag-type {
            type boolean;
            description
              "This identifies the group type; user group,
              app group or device group.";
          }
```

```
          leaf tag-server-information {
            type inet:ipv4-address;
            description
              "The description of suthentication method;
              token-based, password, certificate,
              single-sign-on";
          }

          leaf tag-application-protocol {
            type string;
            description
              "This filed identifies the protocol e.g. LDAP,
              Active Directory, or CMDB";
          }

          leaf tag-server-credential {
            type string;
            description
              "This field identifies the credential
              information needed to access the tag server";
          }
        }

        list user-group{
          key "user-group-id";

          leaf user-group-id {
          type uint16;
          mandatory true;
          description
            "This represents the the user group id.";
          }
          description
          "This represents the user group.";

          leaf name {
            type string;
            description
              "This field identifies the name of user-group.";
          }

          leaf date {
            type yang:date-and-time;
            description
              "when this user-group was created or last modified.";
          }

          leaf group-type {
```

```
            type enumeration{
              enum user-tag{
                description
                  "The user group is based on user-tag.";
              }
              enum user-name{
                description
                  "The user group is based on user-name.";
              }
              enum ip-address{
                description
                  "The user group is based on ip-address.";
              }
            }

            description
              "This describes the group type; User-tag,
              User-name or IP-address.";
          }

          leaf meta-data-server {
            type inet:ipv4-address;
            description
              "This references metadata source";
          }

          leaf group-member {
            type string;
            description
              "This describes the user-tag information";
          }

          leaf risk-level {
            type uint16;
            description
              "This represents the threat level; valid range
               may be 0 to 9.";
          }
        }

        list device-group {
          key "device-group-id";
          leaf device-group-id {
          type uint16;
           description
            "This represents a device group id.";
          }
          description
```

```
          "This represents a device group.";
        leaf name {
          type string;
        description
            "This field identifies the name of
            a device-group.";
        }
        leaf date {
          type yang:date-and-time;
          description
          "The date when this group was create or
          last modified.";
        }

        leaf group-type {
          type enumeration{
            enum device-tag{
              description
                "The device group is based on device-tag.";
            }
            enum device-name{
              description
                "The device group is based on device-name.";
            }
            enum ip-address{
              description
                "The device group is based on ip-address.";
            }
          }
          description
            "This describes the group type; device-tag,
            device-name or IP-address.";
        }

        leaf meta-data-server {
          type inet:ipv4-address;
          description
            "This references meta-data-source
            object.";
        }

        leaf group-member {
          type string;
          description
            "This describes the device-tag, device-name or
            IP-address information";
        }
```

```
          leaf risk-level {
            type uint16;
            description
              "This represents the threat level; valid range
              may be 0 to 9.";
          }
        }

        list application-group{
          key "application-group-id";
          leaf application-group-id {
          type uint16;
          description
          "This represents an application group id.";
          }
          description
          "This represents an application group.";
          leaf name {
            type string;
            description
            "This field identifies the name of
            an application group";
          }

          leaf date {
            type yang:date-and-time;
            description
            "The date when this group was created or
            last modified.";
          }

          leaf group-type {
            type enumeration{
              enum application-tag{
                description
                  "The application group is based on application-tag.";
              }
              enum device-name{
                description
                  "The application group is based on application-name.";
              }
              enum ip-address{
                description
                  "The application group is based on ip-address.";
              }
            }
            description
              "This identifies the group type;
```

```
              application-tag, application-name or
              IP-address.";
          }

          leaf meta-data-server {
            type inet:ipv4-address;
            description
              "This references meta-data-source
              object.";
          }

          leaf group-member {
            type string;
            description
              "This describes the application-tag,
              application-name or IP-address information";
          }

          leaf risk-level {
            type uint16;
            description
              "This represents the threat level; valid range
               may be 0 to 9.";
          }
        }

        list location-group{
          key "location-group-id";
          leaf location-group-id {
          type uint16;
          description
          "This represents a location group id.";
          }
          description
          "This represents a location group.";

          leaf name {
            type string;
            description
            "This field identifies the name of
            a location group";

          }

          leaf date {
            type yang:date-and-time;
            description
            "The date when this group was created or
```

```
          last modified.";
        }

        leaf group-type {
          type enumeration{
            enum location-tag{
              description
                "The location group is based on location-tag.";
            }
            enum location-name{
              description
                "The location group is based on location-name.";
            }
            enum ip-address{
              description
                "The location group is based on ip-address.";
            }
          }
          description
            "This identifies the group type;
            location-tag, location-name or
            IP-address.";
        }

        leaf meta-data-server {
          type inet:ipv4-address;
          description
            "This references meta-data-source
            object.";
        }

        leaf group-member {
          type string;
          description
            "This describes the location-tag,
            location-name or IP-address information";
        }

        leaf risk-level {
          type uint16;
          description
            "This represents the threat level; valid range
             may be 0 to 9.";
        }
      }
  }

  container threat-feed {
```

```
      description
      "this describes the list of threat-feed.";

        list threat-feed {
          key "threat-feed-id";
          leaf threat-feed-id {
          type uint16;
          mandatory true;
          description
            "This represents the threat-feed-id.";
          }
          description
            "This represents the threat feed within the
            threat-prevention-list.";
          leaf name {
            type string;
            description
              "Name of the theat feed.";
          }

          leaf date {
            type yang:date-and-time;
            description
              "when the threat-feed was created.";
          }

          leaf feed-type {
            type enumeration {
              enum unknown {
                description
                  "feed-type is unknown.";
              }
              enum ip-address {
                description
                  "feed-type is IP address.";
              }
              enum url {
                description
                  "feed-type is URL.";
              }
            }
            mandatory true;
            description
              "This determined whether the feed-type is IP address
              based or URL based.";
          }

          leaf feed-server {
```

```
            type inet:ipv4-address;
            description
              "this contains threat feed server information.";
          }

          leaf feed-priority {
            type uint16;
            description
              "this describes the priority of the threat from
              0 to 5, where 0 means the threat is minimum and
              5 meaning the maximum.";
          }
      }

      list custom-list {
          key "custom-list-id";
          leaf custom-list-id {
          type uint16;
          description
          "this describes the custom-list-id.";
          }
          description
          "this describes the threat-prevention custom list.";
          leaf name {
            type string;
            description
              "Name of the custom-list.";
          }

          leaf date {
            type yang:date-and-time;
            description
              "when the custom list was created.";
          }

          leaf list-type {
            type enumeration {
              enum unknown {
                description
                  "list-type is unknown.";
              }
              enum ip-address {
                description
                  "list-type is IP address.";
              }
              enum mac-address {
                description
                  "list-type is MAC address.";
```

```
              }
              enum url {
                description
                  "list-type is URL.";
              }
            }
            mandatory true;
            description
              "This determined whether the feed-type is IP address
              based or URL based.";
          }

         leaf list-property {
            type enumeration {
              enum unknown {
                description
                  "list-property is unknown.";
              }
              enum blacklist {
                description
                  "list-property is blacklist.";
              }
              enum whitelist {
                description
                  "list-property is whitelist.";
              }
            }
            mandatory true;
            description
              "This determined whether the list-type is blacklist
              or whitelist.";
          }

          leaf list-content {
            type string;
            description
              "This describes the contents of the custom-list.";
          }
      }

      list malware-scan-group {
          key "malware-scan-group-id";
          leaf malware-scan-group-id {
          type uint16;
          mandatory true;
          description
          "This is the malware-scan-group-id.";
          }
```

```
          description
          "This represents the malware-scan-group.";
          leaf name {
            type string;
            description
              "Name of the malware-scan-group.";
          }

          leaf date {
            type yang:date-and-time;
            description
              "when the malware-scan-group was created.";
          }

          leaf signature-server {
            type inet:ipv4-address;
            description
              "This describes the signature server of the
              malware-scan-group.";
          }

          leaf file-types {
            type string;
            description
              "This contains a list of file types needed to
              be scanned for the virus.";
          }

          leaf malware-signatures {
            type string;
            description
              "This contains a list of malware signatures or hash.";
          }
      }

      list event-map-group {
          key "event-map-group-id";
          leaf event-map-group-id {
          type uint16;
          mandatory true;
          description
          "This is the event-map-group-id.";
          }
          description
          "This represents the event map group.";

          leaf name {
            type string;
```

```
            description
              "Name of the event-map.";
          }

          leaf date {
            type yang:date-and-time;
            description
              "when the event-map was created.";
          }

          leaf security-events {
            type string;
            description
              "This contains a list of security events.";
          }

          leaf threat-map {
            type string;
            description
              "This contains a list of threat levels.";
          }
      }
    }

    container telemetry-data {
      description
      "Telemetry provides visibility into the network
      activities which can be tapped for further
      security analytics, e.g., detecting potential
      vulnerabilities, malicious activities, etc.";

      list telemetry-data {
        key "telemetry-data-id";

        leaf telemetry-data-id {
        type uint16;
        mandatory true;
        description
        "This is ID for telemetry-data-id.";
        }
        description
        "This is ID for telemetry-data.";

        leaf name {
          type string;
          description
            "Name of the telemetry-data object.";
        }
```

```
        leaf date {
          type yang:date-and-time;
          description
            "This field states when the telemery-data
            object was created.";
        }

        leaf logs {
          type boolean;
          description
            "This field identifies whether logs
            need to be collected.";
        }

        leaf syslogs {
          type boolean;
          description
            "This field identifies whether System logs
            need to be collected.";
        }

        leaf snmp {
          type boolean;
          description
            "This field identifies whether 'SNMP traps' and
            'SNMP alarms' need to be collected.";
        }

        leaf sflow {
          type boolean;
          description
            "This field identifies whether 'sFlow' data
            need to be collected.";
        }

        leaf netflow {
          type boolean;
          description
            "This field identifies whether 'NetFlow' data
            need to be collected.";
        }

        leaf interface-stats {
          type boolean;
          description
            "This field identifies whether 'Interface' data
            such as packet bytes and counts need to be
            collected.";
```

```
        }
      }

      list telemetry-source {
        key "telemetry-source-id";

        leaf telemetry-source-id {
        type uint16;
        mandatory true;
        description
        "This is ID for telemetry-source-id.";
        }
        description
        "This is ID for telemetry-source.";

        leaf name {
          type string;
          description
            "This identifies the name of this object.";
        }

        leaf date {
          type yang:date-and-time;
            description
              "Date this object was created or last modified";
        }

        leaf source-type {
          type enumeration {
            enum network-nsf {
              description
                "NSF telemetry source type is network-nsf.";
            }

            enum firewall-nsf {
              description
                "NSF telemetry source type is firewall-nsf.";
            }
            enum ids-nsf {
              description
                "NSF telemetry source type is ids-nsf.";
            }
            enum ips-nsf {
              description
                "NSF telemetry source type is ips-nsf.";
            }
            enum proxy-nsf {
              description
```

```
                "NSF telemetry source type is proxy-nsf.";
            }
            enum other-nsf {
              description
                "NSF telemetry source type is other-nsf.";
            }
          }
          description
            "This should have one of the following type of
            the NSF telemetry source: NETWORK-NSF,
            FIREWALL-NSF, IDS-NSF, IPS-NSF,
            PROXY-NSF, VPN-NSF, DNS, ACTIVE-DIRECTORY,
            IP Reputation Authority, Web Reputation
            Authority, Anti-Malware Sandbox, Honey Pot,
            DHCP, Other Third Party, ENDPOINT";
        }

        leaf nsf-source {
          type inet:ipv4-address;
           description
            "This field contains information such as
            IP address and protocol (UDP or TCP) port
            number of the NSF providing telemetry data.";
        }

        leaf nsf-credentials {
          type string;
          description
            "This field contains username and password
            to authenticate with the NSF.";
        }

        leaf collection-interval {
          type uint16;
          units seconds;
          default 5000;
          description
            "This field contains time in milliseconds
             between each data collection. For example,
             a value of 5000 means data is streamed to
             collector every 5 seconds. Value of 0 means
             data streaming is event-based";
        }

        leaf collection-method {
          type enumeration {
            enum unknown {
              description
```

```
                "collection-method is unknown.";
            }
            enum push-based {
              description
                "collection-method is PUSH-based.";
            }
            enum pull-based {
              description
                "collection-method is PULL-based.";
            }
          }
          description
          "This field contains a method of collection,
          i.e., whether it is PUSH-based or PULL-based.";
        }
        leaf heartbeat-interval {
          type uint16;
          units seconds;
          description
          "time in seconds the source sends telemetry
          heartbeat.";
        }

        leaf qos-marking {
          type uint16;
          description
          "DSCP value must be contained in this field.";
        }
      }

      list telemetry-destination {
        key "telemetry-destination-id";

        leaf telemetry-destination-id {
        type uint16;
        description
        "this represents the telemetry-destination-id";
        }
        description
        "This object contains information related to
        telemetry destination. The destination is
        usually a collector which is either a part of
        Security Controller or external system
        such as Security Information and Event
        Management (SIEM).";

        leaf name {
          type string;
```

```
          description
            "This identifies the name of this object.";
        }

        leaf date {
          type yang:date-and-time;
          description
            "Date this object was created or last
            modified";
        }

          leaf collector-source {
            type inet:ipv4-address;
              description
                "This field contains information such as
                IP address and protocol (UDP or TCP) port
                number for the collector’s destination.";
          }

          leaf collector-credentials {
            type string;
              description
              "This field contains the username and
             password for the collector.";
          }

          leaf data-encoding {
            type string;
             description
              "This field contains the telemetry data encoding
              in the form of schema.";
          }

          leaf data-transport {
            type enumeration{
              enum grpc {
                description
                  "telemetry data protocol is grpc.";
              }
              enum buffer-over-udp{
                description
                  "telemetry data protocol is buffer over UDP.";
              }
            }
            description
            "This field contains streaming telemetry data
            protocols. This could be gRPC, protocol
            buffer over UDP, etc.";
```

```
          }
      }
    }
  }
<CODE ENDS>
```

Figure 2: YANG for policy-general

6. Security Considerations

The data model for the I2NSF Consumer-Facing Interface is derived from the I2NSF Consumer-Facing Interface Information Model [client-facing-inf-im], so the same security considerations with the information model should be included in this document. The data model needs to support a mechanism to protect Consumer-Facing Interface to Security Controller.

7. Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This document has greatly benefited from inputs by Hyoungshick Kim, Mahdi F. Dachmehchi, Seungjin Lee, Jinyong Tim Kim, and Daeyoung Hyun.

8. Contributors

I2NSF is a group effort. The following people actively contributed to the consumer facing interface data model, and are considered co-authors: o Hyoungshick Kim (Sungkyunkwan University) o Seungjin Lee (Sungkyunkwan University)

9. References

9.1. Normative References

[RFC3444] Pras, A., "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

9.2. Informative References

[client-facing-inf-im]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-kumar-i2nsf-client-facing-interface-im-04 (work in progress), July 2017.

[client-facing-inf-req]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-03 (work in progress), July 2017.

[i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-08 (work in progress), October 2017.

[i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Birkholz, H., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-terminology-04 (work in progress), July 2017.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Appendix A. Changes from draft-jeong-i2nsf-consumer-facing-interface-dm-05

The following changes have been made from draft-jeong-i2nsf-consumer-facing-interface-dm-05:

o In Section 4, the YANG has been modified to represent a policy delivered over the consumer facing interface. More specifically, the YANG model has been modified so that a policy-domain object can have multiple tenants, and as a result, the policy-tenant leaf in the tree is added to be the child of policy-domain object. This clarifies the relationship between a domain and tenants.

o The overall organization of the YANG data model and its data types have also been reviewed and corrected, and produced the corresponding data tree as shown in the Section 5. The reviewed data tree model and YANG fully adopted Event-Condition-Action (ECA) scheme as suggested in the most recent draft about the I2NSF Consumer-Facing Interface Information Model [client-facing-inf-im] and I2NSF Framework [i2nsf-framework].

o The data tree model in Appendix B and Yang in Appendix C have also been modified for better adoption of ECA based policy generation.

o A revised version of an example XML format output is as shown in Appendix D for VoIP service policy based on Yang in Appendix C.

o Overall editorial errors have been corrected.

Appendix B. Use Case: Policy Instance Example for VoIP/VoLTE Security Services

A common scenario for VoIP/VoLTE policy enforcement could be that a malicious call is made to a benign user of any telecommunication company. For example, imagine a case wherea company "A" employs a hacker with a malicious attempt to hack a user’s phone with malware. The company "A" is located in a country, such as Africa, and uses the user’s hacked phone to call the company. The hacked user is unaware of the company "A" so complains about the international call that was made to the company "B", which is the user’s telecommunications company. The company "A" charges the company "B" for the international call. The company "B" cannot charge the user for the call, and has no choice but to pay the company "A". The following shows the example data tree model for the VoIP/VoLTE services. Multi-tenancy, endpoint groups, threat prevention, and telemetry data components are general part of the tree model, so we can just modify the policy instance in order to generate and enforce high-level policies. The policy-calendar can act as a scheduler to set the star and end time to block calls which uses suspicious ids, or calls from other countries.

```
module: policy-voip
    +--rw policy-voip
    |  +--rw rule-voip* [rule-voip-id]
    |  |  +--rw rule-voip-id           uint16
    |  |  +--rw name?                  string
    |  |  +--rw date?                  yang:date-and-time
    |  |  +--rw event* [event-id]
    |  |  |  +--rw event-id            string
    |  |  |  +--rw name?               string
    |  |  |  +--rw date?               yang:date-and-time
    |  |  |  +--rw event-type?         string
    |  |  |  +--rw Time-Information?   string
    |  |  |  +--rw event-map-group?    -> /threat-feed/event-map-group
    |  |  |                               /event-map-group-id
    |  |  |  +--rw enable?             boolean
    |  |  +--rw condition* [condition-id]
    |  |  |  +--rw condition-id        string
    |  |  |  +--rw source-caller?      -> /threat-feed/threat-feed
    |  |  |                               /threat-feed-id
    |  |  |  +--rw destination-callee? -> /threat-feed/custom-list
    |  |  |                               /custom-list-id
    |  |  |  +--rw match?              boolean
    |  |  |  +--rw match-direction?    string
    |  |  |  +--rw exception?          string
    |  |  +--rw action* [action-id]
    |  |  |  +--rw action-id           string
    |  |  |  +--rw name?               string
    |  |  |  +--rw date?               yang:date-and-time
    |  |  |  +--rw primary-action?     string
    |  |  |  +--rw secondary-action?   string
    |  |  +--rw precedence?            uint16
    |  +--rw owner* [owner-id]
    |     +--rw owner-id               string
    |     +--rw name?                  string
    |     +--rw date?                  yang:date-and-time
    +--rw threat-feed
       +--rw threat-feed* [threat-feed-id]
       |  +--rw threat-feed-id         uint16
       |  +--rw name?                  string
       |  +--rw date?                  yang:date-and-time
       |  +--rw feed-type              enumeration
       |  +--rw feed-server?           inet:ipv4-address
       |  +--rw feed-priority?         uint16
       +--rw custom-list* [custom-list-id]
       |  +--rw custom-list-id         uint16
```

```
         |  +--rw name?                  string
         |  +--rw date?                  yang:date-and-time
         |  +--rw list-type              enumeration
         |  +--rw list-property          enumeration
         |  +--rw list-content?          string
         +--rw malware-scan-group* [malware-scan-group-id]
         |  +--rw malware-scan-group-id  uint16
         |  +--rw name?                  string
         |  +--rw date?                  yang:date-and-time
         |  +--rw signature-server?      inet:ipv4-address
         |  +--rw file-types?            string
         |  +--rw malware-signatures?    string
         +--rw event-map-group* [event-map-group-id]
            +--rw event-map-group-id     uint16
            +--rw name?                  string
            +--rw date?                  yang:date-and-time
            +--rw security-events?       string
            +--rw threat-map?            string
```

Figure 3: Policy Instance Example for VoIP/VoLTE Security Services

Appendix C. Policy Instance YANG Example for VoIP/VoLTE Security Services

The following YANG data model is a policy instance for VoIP/VoLTE security services. The policy-calendar can act as a scheduler to set the start time and end time to block malicious calls which use suspicious IDs, or calls from other countries.

```
<CODE BEGINS> file "ietf-i2nsf-cf-interface-voip.yang"

  module ietf-policy-voip {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-policy-voip";
  prefix
    "cf-interface";

  import ietf-yang-types{
    prefix yang;
  }

  import ietf-inet-types{
    prefix inet;
  }
  organization
    "IETF I2NSF (Interface to Network Security Functions)
```

```
        Working Group";

     contact
       "WG Web: <http://tools.ietf.org/wg/i2nsf>
        WG List: <mailto:i2nsf@ietf.org>

        WG Chair: Adrian Farrel
        <mailto:Adrain@olddog.co.uk>

        WG Chair: Linda Dunbar
        <mailto:Linda.duhbar@huawei.com>

        Editor: Jaehoon Paul Jeong
        <mailto:pauljeong@skku.edu>";

     description
       "This module defines a YANG data module for consumer-facing
        interface to security controller.";

     revision "2018-03-05"{
       description "sixth revision";
       reference
         "draft-kumar-i2nsf-client-facing-interface-im-04";
     }

     container policy-voip {
             description
               "This object is a policy instance to have
               complete information such as where and when
               a policy need to be applied.";
       list rule-voip {
             key "rule-voip-id";
             leaf rule-voip-id {
             type uint16;
             mandatory true;
             description
             "This is ID for rules.";
             }
             description
             "This is a container for rules.";
             leaf name {
               type string;
               description
                 "This field idenfifies the name of this object.";
             }
             leaf date {
               type yang:date-and-time;
               description
```

```
                 "Date this object was created or last
                 modified";
             }
           list event {
             key "event-id";
             description
             "This represents the security event of a
                 policy-rule.";
             leaf event-id {
               type string;
               mandatory true;
               description
                 "This represents the event-id.";
             }
             leaf name {
               type string;
               description
                 "This field idenfifies the name of this object.";
             }
             leaf date {
               type yang:date-and-time;
               description
                 "Date this object was created or last
                 modified";
             }
             leaf event-type {
               type string;
               description
                 "This field identifies the event event type
                 .";
             }
             leaf Time-Information {
               type string;
               description
                 "This field contains time calendar such as
                BEGIN-TIME and END-TIME for one time
                enforcement or recurring time calendar for
                periodic enforcement.";
             }
             leaf event-map-group {
               type leafref{
                 path "/threat-feed/event-map-group/event-map-group-id";
             }
               description
               "This field contains security events or threat
               map in order to determine when a policy need
               to be activated. This is a reference to
               Evnet-Map-Group.";
```

```
             }
             leaf enable {
               type boolean;
               description
                 "This determines whether the condition
                 matches the security event or not.";
             }
           }
           list condition {
             key "condition-id";
             description
             "This represents the condition of a
                 policy-rule.";
             leaf condition-id {
               type string;
               description
                 "This represents the condition-id.";
             }
             leaf source-caller {
               type leafref {
                 path "/threat-feed/threat-feed/threat-feed-id";
               }
               description
                 "This field identifies the source of
                 the traffic. This could be reference to
                 either 'Policy Endpoint Group' or
                 'Threat-Feed' or 'Custom-List' if Security
                 Admin wants to specify the source; otherwise,
                 the default is to match all traffic.";
             }
             leaf destination-callee {
               type leafref {
                 path "/threat-feed/custom-list/custom-list-id";
               }
               description
                 "This field identifies the source of
                 the traffic. This could be reference to
                 either 'Policy Endpoint Group' or
                 'Threat-Feed' or 'Custom-List' if Security
                 Admin wants to specify the source; otherwise,
                 the default is to match all traffic.";
             }
             leaf match {
               type boolean;
               description
                 "This field identifies the match criteria used to
                evaluate whether the specified action need to be
                taken or not.  This could be either a Policy-
```

```
                Endpoint-Group identifying a Application set or a
                set of traffic rules.";
             }
             leaf match-direction {
               type string;
               description
                 "This field identifies if the match criteria is
                to evaluated for both direction of the traffic or
                only in one direction with default of allowing in
                the other direction for stateful match conditions.
                This is optional and by default rule should apply
                in both directions.";
             }
             leaf exception {
               type string;
               description
                 "This field identifies the exception
                 consideration when a rule is evaluated for a
                 given communication.  This could be reference to
                 Policy-Endpoint-Group object or set of traffic
                 matching criteria.";
             }
           }
           list action {
             key "action-id";
             leaf action-id {
             type string;
             mandatory true;
             description
               "this represents the policy-action-id.";
             }
             description
               "This object represents actions that a
               Security Admin wants to perform based on
               a certain traffic class.";
             leaf name {
               type string;
               description
                 "The name of the policy-action object.";
             }

             leaf date {
               type yang:date-and-time;
               description
                 "When the object was created or last
                 modified.";
             }
             leaf primary-action {
```

```
              type string;
              description
                "This field identifies the action when a rule
                is matched by NSF. The action could be one of
                'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
                'AUTHENTICATE-SESSION', 'IPS, 'APP-FIREWALL', etc.";
            }
            leaf secondary-action {
              type string;
              description
                "This field identifies additional actions if
                a rule is matched. This could be one of 'LOG',
                'SYSLOG', 'SESSION-LOG', etc.";
            }
          }
          leaf precedence {
            type uint16;
              description
                "This field identifies the precedence
                assigned to this rule by Security Admin.
                This is helpful in conflict resolution
                when two or more rules match a given
                traffic class.";
          }
      }
      list owner {
            key "owner-id";
            leaf owner-id {
            type string;
            mandatory true;
            description
              "this represents the owner-id.";
            }
            description
              "This field defines the owner of this policy.
              Only the owner is authorized to modify the
              contents of the policy.";
            leaf name {
              type string;
              description
                "The name of the owner.";
            }
            leaf date {
              type yang:date-and-time;
              description
                "When the object was created or last
                modified.";
            }
```

```
        }
      }
      container threat-feed {
          description
          "this describes the list of threat-feed.";

            list threat-feed {
              key "threat-feed-id";
              leaf threat-feed-id {
              type uint16;
              mandatory true;
              description
                "This represents the threat-feed-id.";
              }
              description
                "This represents the threat feed within the
                threat-prevention-list.";
              leaf name {
                type string;
                description
                  "Name of the theat feed.";
              }

              leaf date {
                type yang:date-and-time;
                description
                  "when the threat-feed was created.";
              }

              leaf feed-type {
                type enumeration {
                  enum unknown {
                    description
                      "feed-type is unknown.";
                  }
                  enum ip-address {
                    description
                      "feed-type is IP address.";
                  }
                  enum url {
                    description
                      "feed-type is URL.";
                  }
                }
                mandatory true;
                description
                  "This determined whether the feed-type is IP address
                  based or URL based.";
```

```
             }

             leaf feed-server {
               type inet:ipv4-address;
               description
                 "this contains threat feed server information.";
             }

             leaf feed-priority {
               type uint16;
               description
                 "this describes the priority of the threat from
                 0 to 5, where 0 means the threat is minimum and
                 5 meaning the maximum.";
             }
         }

         list custom-list {
             key "custom-list-id";
             leaf custom-list-id {
             type uint16;
             description
             "this describes the custom-list-id.";
             }
             description
             "this describes the threat-prevention custom list.";
             leaf name {
               type string;
               description
                 "Name of the custom-list.";
             }

             leaf date {
               type yang:date-and-time;
               description
                 "when the custom list was created.";
             }

             leaf list-type {
               type enumeration {
                 enum unknown {
                   description
                     "list-type is unknown.";
                 }
                 enum ip-address {
                   description
                     "list-type is IP address.";
                 }
```

```
                enum mac-address {
                  description
                    "list-type is MAC address.";
                }
                enum url {
                  description
                    "list-type is URL.";
                }
              }
              mandatory true;
              description
                "This determined whether the feed-type is IP address
                based or URL based.";
            }

           leaf list-property {
              type enumeration {
                enum unknown {
                  description
                    "list-property is unknown.";
                }
                enum blacklist {
                  description
                    "list-property is blacklist.";
                }
                enum whitelist {
                  description
                    "list-property is whitelist.";
                }
              }
              mandatory true;
              description
                "This determined whether the list-type is blacklist
                or whitelist.";
            }

            leaf list-content {
              type string;
              description
                "This describes the contents of the custom-list.";
            }
        }

        list malware-scan-group {
            key "malware-scan-group-id";
            leaf malware-scan-group-id {
            type uint16;
            mandatory true;
```

```
             description
             "This is the malware-scan-group-id.";
             }
             description
             "This represents the malware-scan-group.";
             leaf name {
               type string;
               description
                 "Name of the malware-scan-group.";
             }

             leaf date {
               type yang:date-and-time;
               description
                 "when the malware-scan-group was created.";
             }

             leaf signature-server {
               type inet:ipv4-address;
               description
                 "This describes the signature server of the
                 malware-scan-group.";
             }

             leaf file-types {
               type string;
               description
                 "This contains a list of file types needed to
                 be scanned for the virus.";
             }

             leaf malware-signatures {
               type string;
               description
                 "This contains a list of malware signatures or hash.";
             }
         }

         list event-map-group {
             key "event-map-group-id";
             leaf event-map-group-id {
             type uint16;
             mandatory true;
             description
             "This is the event-map-group-id.";
             }
             description
             "This represents the event map group.";
```

```
             leaf name {
               type string;
               description
                 "Name of the event-map.";
             }

             leaf date {
               type yang:date-and-time;
               description
                 "when the event-map was created.";
             }

             leaf security-events {
               type string;
               description
                 "This contains a list of security events.";
             }

             leaf threat-map {
               type string;
               description
                 "This contains a list of threat levels.";
             }
         }
       }
   }


   <CODE ENDS>
```

Figure 4: Policy Instance YANG Example for VoIP Security Services

Appendix D. Example XML output for VoIP service

In this section, we present an XML example for VoIP service. Here, we are going to drop calls commin from a country with an Ip from South Africa that is classified as malicious.


```
<?xml version="1.1" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:restconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
   <config>
    <i2nsf-cf-interface-voip-req nc:operation="create">
        <policy-voip>
```

```
            <rule-voip>
                <rule-voip-id>01</rule-voip-id>
                <rule-voip-name>voip-policy-example</rule-voip-name>
                <rule-voip-date>2017.10.25/20:30:32</rule-voip-date>
                <event>
                    <event-id>01</event-id>
                    <event-name>voip_call</event-name>
                    <event-date>2017.10.25/20:30:32</event-date>
                    <event-type>malicious</event-type>
                    <time-information>
                        <begin-time>22:00</begin-time>
                        <end-time>08:00</end-time>
                    </time-information>
                    <event-map-group>19</event-map-group>
                    <enable>True</enable>
                </event>
                <condition>
                    <condition-id>01</condition-id>
                    <source-caller>105.176.0.0</source-caller>
                    <destination-callee>192.168.171.35</destination-callee>
                    <match-direction>default</match-direction>
                    <exeption>00</exeption>
                </condition>
                <action>
                    <action-id>01</action-id>
                    <action-name>action-voip</action-name>
                    <action-date>2017.10.25/20:30:32</action-date>
                    <primary-action>DENY</primary-action>
                    <secondary-action>LOG</secondary-action>
                </action>
                <precedence>none</precedence>
              <owner>
                <owner-id>01</owner-id>
                <name>i2nsf-admin</name>
              </owner>
            </rule-voip>
        </policy-voip>
    </i2nsf-cf-interface-voip-req>
   </config>
 </edit-config>
</rpc>
```

Figure 5: An XML example for VoIP service

Authors' Addresses

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Eunsoo Kim
Department of Electrical and Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4104
EMail: eskim86@skku.edu
URI: http://seclab.skku.edu/people/eunsoo-kim/

Tae-Jin Ahn
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8409
EMail: taejin.ahn@kt.com

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

EMail: rkkumar@juniper.net

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
EMail: shares@ndzh.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

J. Kim
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
Q. Lin
Huawei
March 5, 2018

I2NSF Network Security Function-Facing Interface YANG Data Model
draft-ietf-i2nsf-nsf-facing-interface-dm-00

Abstract

This document defines a YANG data model corresponding to the information model for Network Security Functions (NSF) facing interface in Interface to Network Security Functions (I2NSF). It describes a data model for the features provided by generic security functions. This data model provides generic components whose vendors is well understood, so that the generic component can be used even if it has some vendor specific functions. These generic functions represent a point of interoperability, and can be provided by any product that offers the required Capabilities. Also, if vendors need additional features for its network security function, they can add the features by extending the YANG data model.

Status of This Memo

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of security services with the information model for Network Security Functions (NSF) facing interface in Interface to Network Security Functions (I2NSF). It provides a specific information model and the corresponding data models for generic network security functions (i.e., network security functions), as defined in [i2nsf-nsf-cap-im]. With these data model, I2NSF controller can control the capabilities of NSFs.

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules.

The "ietf-i2nsf-nsf-facing-interface" YANG module defined in this document provides the following features:

- o configuration of I2NSF security policy rule for generic network security function policy
- o configuration of event clause for generic network security function policy
- o configuration of condition clause for generic network security function policy
- o configuration of action clause for generic network security function policy

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 terminology described in [i2nsf-nsf-cap-im][i2rs-rib-data-model][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol.
- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is

independent of data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] 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. The Structure and Objective of I2NSF Security Policy

4.1. I2NSF Security Policy Rule

This shows a policy rule for generic network security functions. The object of a policy rule is defined as policy information and rule information. This includes ECA Policy Rule such as Event Clause Objects, Condition Clause Objects, Action Clause Objects, Resolution Strategy, and Default Action.

4.2. Event Clause

This shows an event clause for generic network security functions. An Event is any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of I2NSF Policy Rules, it is used to determine whether the Condition clause of the I2NSF Policy Rule can be evaluated or not. The object of an event clauses is defined as user security event, device security event, system security event, and time security event. The objects of event clauses can be extended according to specific vendor event features.

4.3. Condition Clause

This shows a condition clause for generic network security functions. A condition is defined as a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to determine whether or not the set of Actions in that (imperative) I2NSF Policy Rule can be executed or not. These objects are defined as packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. The objects of action clauses can be extended according to specific vendor condition features.

4.4. Action Clause

This shows an action clause for generic network security functions. An action is used to control and monitor aspects of flow-based NSFs when the event and condition clauses are satisfied. NSFs provide security functions by executing various Actions. The object of an action clause is defined as ingress action, egress action, and apply profile action. The objects of action clauses can be extended according to specific vendor action features.

5. Data Model Structure

This section shows a data model structure tree of generic network security functions that are defined in the [i2nsf-nsf-cap-im].

- o Consideration of ECA Policy Model by Aggregating the Event, Condition, and Action Clauses Objects.
- o Consideration of Capability Algebra.
- o Consideration of NSFs Capability Categories (i.e., Network Security, Content Security, and Attack Mitigation Capabilities).
- o Definitions for Network Security Event Class, Network Security Condition Class, and Network Security Action Class.

5.1. I2NSF Security Policy Rule

The data model for the identification of network security policy has the following structure:

```
module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|  +--rw policy-name             string
|  +--rw eca-policy-rules* [rule-id]
```

```
   |  |  +--rw rule-id             uint8
   |  |  +--rw rule-description?   string
   |  |  +--rw rule-rev?           uint8
   |  |  +--rw rule-priority?      uint8
   |  |  +--rw policy-event-clause-agg-ptr*       instance-identifier
   |  |  +--rw policy-condition-clause-agg-ptr*   instance-identifier
   |  |  +--rw policy-action-clause-agg-ptr*      instance-identifier
   |  |  +--rw time-zone
   |  |     +--rw absolute-time-zone
   |  |     |  +--rw time
   |  |     |  |  +--rw start-time?   yang:date-and-time
   |  |     |  |  +--rw end-time?     yang:date-and-time
   |  |     |  +--rw date
   |  |     |  |  +--rw absolute-date*   yang:date-and-time
   |  |     +--rw periodic-time-zone
   |  |        +--rw day
   |  |        |  +--rw sunday?   boolean
   |  |        |  +--rw monday?   boolean
   |  |        |  +--rw tuesday?   boolean
   |  |        |  +--rw wednesday?   boolean
   |  |        |  +--rw thursday?   boolean
   |  |        |  +--rw friday?   boolean
   |  |        |  +--rw saturday?   boolean
   |  |        +--rw month
   |  |           +--rw january?   boolean
   |  |           +--rw february?   boolean
   |  |           +--rw march?   boolean
   |  |           +--rw april?   boolean
   |  |           +--rw may?   boolean
   |  |           +--rw june?   boolean
   |  |           +--rw july?   boolean
   |  |           +--rw august?   boolean
   |  |           +--rw september?   boolean
   |  |           +--rw october?   boolean
   |  |           +--rw november?   boolean
   |  |           +--rw december?   boolean
   |  +--rw resolution-strategy
   |  |  +--rw (resolution-strategy-type)?
   |  |     +--:(fmr)
   |  |     |  +--rw first-matching-rule?   boolean
   |  |     +--:(lmr)
   |  |        +--rw last-matching-rule?    boolean
   |  +--rw default-action
   |     +--rw default-action-type?   ingress-action
   +--rw event-clause-container
   |  ...
   +--rw condition-clause-container
   |  ...
```

```
   +--rw action-clause-container
      ...
```

Figure 1: Data Model Structure for Network Security Policy Identification

5.2. Event Clause

The data model for event rule has the following structure:

```
   module: ietf-i2nsf-policy-rule-for-nsf
   +--rw i2nsf-security-policy* [policy-name]
   |      ...
   |  +--rw eca-policy-rules* [rule-id]
   |      ...
   |  +--rw resolution-strategy
   |      ...
   |  +--rw default-action
   |      ...
   +--rw event-clause-container
   |  +--rw event-clause-list* [eca-object-id]
   |     +--rw entity-class?        identityref
   |     +--rw eca-object-id        string
   |     +--rw manual?              string
   |     +--rw sec-event-content    string
   |     +--rw sec-event-format     sec-event-format
   |     +--rw sec-event-type       string
   +--rw condition-clause-container
   |  ...
   +--rw action-clause-container
      ...
```

Figure 2: Data Model Structure for Event Rule

These objects are defined as user security event, device security event, system security event, and time security event. These objects can be extended according to specific vendor event features. We will add additional event objects for more generic network security functions.

5.3. Condition Clause

The data model for condition rule has the following structure:

```
  module: ietf-i2nsf-policy-rule-for-nsf
  +--rw i2nsf-security-policy* [policy-name]
  |      ...
```

```
   |  +--rw eca-policy-rules* [rule-id]
   |     ...
   |  +--rw resolution-strategy
   |     ...
   |  +--rw default-action
   |     ...
   +--rw event-clause-container
   |  ...
   +--rw condition-clause-container
   |  +--rw condition-clause-list* [eca-object-id]
   |     +--rw entity-class?                     identityref
   |     +--rw eca-object-id                     string
   |     +--rw packet-security-condition
   |     |  +--rw packet-manual?                    string
   |     |  +--rw packet-security-mac-condition
   |     |  |  +--rw pkt-sec-cond-mac-dest*         yang:phys-address
   |     |  |  +--rw pkt-sec-cond-mac-src*          yang:phys-address
   |     |  |  +--rw pkt-sec-cond-mac-8021q*        string
   |     |  |  +--rw pkt-sec-cond-mac-ether-type*   string
   |     |  |  +--rw pkt-sec-cond-mac-tci*          string
   |     |  +--rw packet-security-ipv4-condition
   |     |  |  +--rw pkt-sec-cond-ipv4-header-length*     uint8
   |     |  |  +--rw pkt-sec-cond-ipv4-tos*               uint8
   |     |  |  +--rw pkt-sec-cond-ipv4-total-length*      uint16
   |     |  |  +--rw pkt-sec-cond-ipv4-id*                uint8
   |     |  |  +--rw pkt-sec-cond-ipv4-fragment*          uint8
   |     |  |  +--rw pkt-sec-cond-ipv4-fragment-offset*   uint16
   |     |  |  +--rw pkt-sec-cond-ipv4-ttl*               uint8
   |     |  |  +--rw pkt-sec-cond-ipv4-protocol*          uint8
   |     |  |  +--rw pkt-sec-cond-ipv4-src*         inet:ipv4-address
   |     |  |  +--rw pkt-sec-cond-ipv4-dest*        inet:ipv4-address
   |     |  |  +--rw pkt-sec-cond-ipv4-ipopts?            string
   |     |  |  +--rw pkt-sec-cond-ipv4-sameip?            boolean
   |     |  |  +--rw pkt-sec-cond-ipv4-geoip*             string
   |     |  +--rw packet-security-ipv6-condition
   |     |  |  +--rw pkt-sec-cond-ipv6-dscp*             string
   |     |  |  +--rw pkt-sec-cond-ipv6-ecn*              string
   |     |  |  +--rw pkt-sec-cond-ipv6-traffic-class*    uint8
   |     |  |  +--rw pkt-sec-cond-ipv6-flow-label*       uint32
   |     |  |  +--rw pkt-sec-cond-ipv6-payload-length*   uint16
   |     |  |  +--rw pkt-sec-cond-ipv6-next-header*      uint8
   |     |  |  +--rw pkt-sec-cond-ipv6-hop-limit*        uint8
   |     |  |  +--rw pkt-sec-cond-ipv6-src*    inet:ipv6-address
   |     |  |  +--rw pkt-sec-cond-ipv6-dest*   inet:ipv6-address
   |     |  +--rw packet-security-tcp-condition
   |     |  |  +--rw pkt-sec-cond-tcp-src-port*         inet:port-number
   |     |  |  +--rw pkt-sec-cond-tcp-dest-port*         inet:port-number
   |     |  |  +--rw pkt-sec-cond-tcp-seq-num*       uint32
```

```
   |     |  |  +--rw pkt-sec-cond-tcp-ack-num*       uint32
   |     |  |  +--rw pkt-sec-cond-tcp-window-size*   uint16
   |     |  |  +--rw pkt-sec-cond-tcp-flags*         uint8
   |     |  +--rw packet-security-udp-condition
   |     |  |  +--rw pkt-sec-cond-udp-src-port*         inet:port-number
   |     |  |  +--rw pkt-sec-cond-udp-dest-port*         inet:port-number
   |     |  |  +--rw pkt-sec-cond-udp-length*   string
   |     |  +--rw packet-security-icmp-condition
   |     |     +--rw pkt-sec-cond-icmp-type*      uint8
   |     |     +--rw pkt-sec-cond-icmp-code*      uint8
   |     |     +--rw pkt-sec-cond-icmp-seg-num*   uint32
   |     +--rw packet-payload-condition
   |     |  +--rw packet-payload-manual?            string
   |     |  +--rw pkt-payload-content*              string
   |     +--rw target-condition
   |     |  +--rw target-manual?                    string
   |     |  +--rw device-sec-context-cond
   |     |     +--rw pc?                  boolean
   |     |     +--rw mobile-phone?        boolean
   |     |     +--rw voip-volte-phone?    boolean
   |     |     +--rw tablet?              boolean
   |     |     +--rw iot?                 boolean
   |     |     +--rw vehicle?             boolean
   |     +--rw users-condition
   |     |  +--rw users-manual?                     string
   |     |  +--rw user
   |     |  |  +--rw (user-name)?
   |     |  |     +--:(tenant)
   |     |  |     |  +--rw tenant    uint8
   |     |  |     +--:(vn-id)
   |     |  |        +--rw vn-id     uint8
   |     |  +--rw group
   |     |     +--rw (group-name)?
   |     |        +--:(tenant)
   |     |        |  +--rw tenant    uint8
   |     |        +--:(vn-id)
   |     |           +--rw vn-id     uint8
   |     +--rw context-condition
   |     |  +--rw context-manual?                   string
   |     +--rw gen-context-condition
   |        +--rw gen-context-manual?               string
   |        +--rw geographic-location
   |           +--rw src-geographic-location*    uint32
   |           +--rw dest-geographic-location*   uint32
   +--rw action-clause-container
      ...
```

Figure 3: Data Model Structure for Condition Rule

These objects are defined as packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. These objects can be extended according to specific vendor condition features. We will add additional condition objects for more generic network security functions.

5.4. Action Clause

The data model for action rule has the following structure:

```
module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|     ...
|  +--rw eca-policy-rules* [rule-id]
|     ...
|  +--rw resolution-strategy
|     ...
|  +--rw default-action
|     ...
+--rw event-clause-container
|  ...
+--rw condition-clause-container
|  ...
+--rw action-clause-container
   +--rw action-clause-list* [eca-object-id]
      +--rw entity-class?                     identityref
      +--rw eca-object-id                     string
         +--rw ingress-action
         |  +--rw ingress-manual?                   string
         |  +--rw ingress-action-type?              ingress-action
         +--rw egress-action
         |  +--rw egress-manual?                    string
         |  +--rw egress-action-type?               egress-action
         +--rw apply-profile
            +--rw profile-manual?                   string
               +--rw content-security-control
               |  +--rw content-security-control-types
               |     +--rw antivirus?             boolean
               |     +--rw ips?                   boolean
               |     +--rw ids?                   boolean
               |     +--rw url-filtering?         boolean
               |     +--rw data-filtering?        boolean
               |     +--rw mail-filtering?        boolean
               |     +--rw file-blocking?         boolean
               |     +--rw file-isolate?          boolean
               |     +--rw pkt-capture?           boolean
               |     +--rw application-control?   boolean
```

```
                  |     +--rw voip-volte?            boolean
                  +--rw attack-mitigation-control
                        +--rw ddos-attack
                        |  +--rw ddos-attack-type
                        |     +--rw network-layer-ddos-attack
                        |     |  +--rw network-layer-ddos-attack-type
                        |     |     +--rw syn-flood?       boolean
                        |     |     +--rw udp-flood?       boolean
                        |     |     +--rw icmp-flood?      boolean
                        |     |     +--rw ip-frag-flood?   boolean
                        |     |     +--rw ipv6-related?    boolean
                        |     +--rw app-layer-ddos-attack
                        |        +--rw app-ddos-attack-types
                        |           +--rw http-flood?      boolean
                        |           +--rw https-flood?     boolean
                        |           +--rw dns-flood?       boolean
                        |           +--rw dns-amp-flood?   boolean
                        |           +--rw ssl-ddos?        boolean
                        +--rw single-packet-attack
                           +--rw single-packet-attack-type
                              +--rw scan-and-sniff-attack
                              |  +--rw scan-and-sniff-attack-types
                              |     +--rw ip-sweep?        boolean
                              |     +--rw port-scanning?   boolean
                              +--rw malformed-packet-attack
                              |  +--rw malformed-packet-attack-types
                              |     +--rw ping-of-death?   boolean
                              |     +--rw teardrop?        boolean
                              +--rw special-packet-attack
                                 +--rw special-packet-attack-types
                                    +--rw oversized-icmp?   boolean
                                    +--rw tracert?          boolean
```

Figure 4: Data Model Structure for Action Rule

These objects are defined as ingress action, egress action, and apply profile action. These objects can be extended according to specific vendor action feature. We will add additional action objects for more generic network security functions.

6. YANG Module

6.1. IETF NSF-Facing Interface YANG Data Module

This section introduces a YANG module for the information model of network security functions, as defined in the [i2nsf-nsf-cap-im].

<CODE BEGINS> file "ietf-i2nsf-policy-rule-for-nsf@2018-03-05.yang"

```
module ietf-i2nsf-policy-rule-for-nsf {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf";
  prefix
    policy-rule-for-nsf;

  import ietf-inet-types{
    prefix inet;
  }
  import ietf-yang-types{
    prefix yang;
  }

  organization
    "IETF I2NSF (Interface to Network Security Functions)
     Working Group";

  contact
    "WG Web: <http://tools.ietf.org/wg/i2nsf>
     WG List: <mailto:i2nsf@ietf.org>

     WG Chair: Adrian Farrel
     <mailto:Adrain@olddog.co.uk>

     WG Chair: Linda Dunbar
     <mailto:Linda.duhbar@huawei.com>

     Editor: Jingyong Tim Kim
     <mailto:timkim@skku.edu>

     Editor: Jaehoon Paul Jeong
     <mailto:pauljeong@skku.edu>

     Editor: Susan Hares
     <mailto:shares@ndzh.com>";

  description
    "This module defines a YANG data module for network security
     functions.";
  revision "2018-03-05"{
    description "The fourth revision";
    reference
      "draft-ietf-i2nsf-capability-00";
  }

  typedef sec-event-format {
      type enumeration {
```

```
        enum unknown {
            description
              "If SecEventFormat is unknown";
        }
        enum guid {
            description
              "If SecEventFormat is GUID
              (Generic Unique IDentifier)";
        }
        enum uuid {
            description
              "If SecEventFormat is UUID
              (Universal Unique IDentifier)";
        }
        enum uri {
            description
              "If SecEventFormat is URI
              (Uniform Resource Identifier)";
        }
        enum fqdn {
            description
              "If SecEventFormat is FQDN
              (Fully Qualified Domain Name)";
        }
        enum fqpn {
            description
              "If SecEventFormat is FQPN
              (Fully Qualified Path Name)";
        }
      }
      description
        "This is used for SecEventFormat.";
  }

  typedef ingress-action {
      type enumeration {
        enum pass {
            description
              "If ingress action is pass";
        }
        enum drop {
            description
              "If ingress action is drop";
        }
        enum reject {
            description
              "If ingress action is reject";
        }
```

```
        enum alert {
            description
              "If ingress action is alert";
        }
        enum mirror {
            description
              "If ingress action is mirror";
        }
      }
      description
        "This is used for ingress action.";
  }

  typedef egress-action {
      type enumeration {
        enum invoke-signaling {
            description
              "If egress action is invoke signaling";
        }
        enum tunnel-encapsulation {
            description
              "If egress action is tunnel encapsulation";
        }
        enum forwarding {
            description
              "If egress action is forwarding";
        }
        enum redirection {
            description
              "If egress action is redirection";
        }
      }
      description
        "This is used for egress action.";
  }

  identity ECA-OBJECT-TYPE {
    description "TBD";
  }

  identity ECA-EVENT-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
  }

  identity ECA-CONDITION-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
```

```
  }

  identity ECA-ACTION-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
  }

   identity EVENT-USER-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }

   identity EVENT-DEV-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }

   identity EVENT-SYS-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }

   identity EVENT-TIME-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }


  grouping i2nsf-eca-object-type {
    leaf entity-class {
      type identityref {
        base ECA-OBJECT-TYPE;
      }
      description "TBD";
    }
    leaf eca-object-id {
        type string;
        description "TBD";
    }
    description "TBD";
  }

  grouping i2nsf-event-type {
      description "TBD";
      leaf manual {
        type string;
        description
          "This is manual for event.
```

```
            Vendors can write instructions for event
            that vendor made";
        }

        leaf sec-event-content {
          type string;
          mandatory true;
          description
           "This is a mandatory string that contains the content
            of the SecurityEvent. The format of the content
            is specified in the SecEventFormat class
            attribute, and the type of event is defined in the
            SecEventType class attribute. An example of the
            SecEventContent attribute is a string hrAdmin,
            with the SecEventFormat set to 1 (GUID) and the
            SecEventType attribute set to 5 (new logon).";
        }

        leaf sec-event-format {
          type sec-event-format;
          mandatory true;
          description
           "This is a mandatory uint 8 enumerated integer, which
            is used to specify the data type of the
            SecEventContent attribute. The content is
            specified in the SecEventContent class attribute,
            and the type of event is defined in the
            SecEventType class attribute. An example of the
            SecEventContent attribute is string hrAdmin,
            with the SecEventFormat attribute set to 1 (GUID)
            and the SecEventType attribute set to 5
            (new logon).";
        }

        leaf sec-event-type {
          type string;
          mandatory true;
          description
           "This is a mandatory uint 8 enumerated integer, which
            is used to specify the type of event that involves
            this user. The content and format are specified in
            the SecEventContent and SecEventFormat class
            attributes, respectively. An example of the
            SecEventContent attribute is string hrAdmin,
            with the SecEventFormat attribute set to 1 (GUID)
            and the SecEventType attribute set to 5
           (new logon).";
        }
```

```
 }


  list i2nsf-security-policy {
    key "policy-name";
    description
      "policy is a list
       including a set of security rules according to certain logic,
       i.e., their similarity or mutual relations, etc. The network
       security policy is able to apply over both the unidirectional
       and bidirectional traffic across the NSF.";

      leaf policy-name {
        type string;
        mandatory true;
        description
          "The name of the policy.
           This must be unique.";
      }

      list eca-policy-rules {
        key "rule-id";
        description
          "This is a rule for network security functions.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The id of the rule.
             This must be unique.";
        }

        leaf rule-description {
          type string;
          description
            "This description gives more information about
             rules.";
        }

        leaf rule-rev {
          type uint8;
          description
            "This shows rule version.";
        }

        leaf rule-priority {
          type uint8;
```

```
          description
            "The priority keyword comes with a mandatory
             numeric value which can range from 1 till 255.";
        }
        leaf-list policy-event-clause-agg-ptr {
            type instance-identifier;
            must 'derived-from-or-self (/event-clause-container/
            event-clause-list/entity-class, "ECA-EVENT-TYPE")';
            description
                "TBD";
        }
        leaf-list policy-condition-clause-agg-ptr {
            type instance-identifier;
            must 'derived-from-or-self (/condition-clause-container/
            condition-clause-list/entity-class, "ECA-CONDITION-TYPE")';
            description
                "TBD";
        }
        leaf-list policy-action-clause-agg-ptr {
            type instance-identifier;
            must 'derived-from-or-self (/action-clause-container/
            action-clause-list/entity-class, "ECA-ACTION-TYPE")';
            description
                "TBD";
        }

        container time-zone {
          description
            "This can be used to apply rules according to time-zone";
          container absolute-time-zone {
            description
            "This can be used to apply rules according to
             absolute-time";
            container time {
              description
                "This can be used to apply rules according to time";
              leaf start-time {
                type yang:date-and-time;
                description
                  "This is start time for time zone";
              }
              leaf end-time {
                type yang:date-and-time;
                description
                  "This is end time for time zone";
              }
            }
            container date {
```

```
              description
                "This can be used to apply rules according to date";
              leaf absolute-date {
                type yang:date-and-time;
                description
                  "This is absolute date for time zone";
              }
            }
          }
          container periodic-time-zone {
            description
             "This can be used to apply rules according to
             periodic-time-zone";
            container day {
              description
               "This can be used to apply rules according
                to periodic day";
              leaf sunday {
                type boolean;
                description
                  "This is sunday for periodic day";
              }
              leaf monday {
                type boolean;
                description
                  "This is monday for periodic day";
              }
              leaf tuesday {
                type boolean;
                description
                  "This is tuesday for periodic day";
              }
              leaf wednesday {
                type boolean;
                description
                  "This is wednesday for periodic day";
              }
              leaf thursday {
                type boolean;
                description
                  "This is thursday for periodic day";
              }
              leaf friday {
                type boolean;
                description
                  "This is friday for periodic day";
              }
              leaf saturday {
```

```
                type boolean;
                description
                  "This is saturday for periodic day";
              }
            }
            container month {
              description
               "This can be used to apply rules according
                to periodic month";
              leaf january {
                type boolean;
                description
                  "This is january for periodic month";
              }
              leaf february {
                type boolean;
                description
                  "This is february for periodic month";
              }
              leaf march {
                type boolean;
                description
                  "This is march for periodic month";
              }
              leaf april {
                type boolean;
                description
                  "This is april for periodic month";
              }
              leaf may {
                type boolean;
                description
                  "This is may for periodic month";
              }
              leaf june {
                type boolean;
                description
                  "This is june for periodic month";
              }
              leaf july {
                type boolean;
                description
                  "This is july for periodic month";
              }
              leaf august {
                type boolean;
                description
                  "This is august for periodic month";
```

```
              }
              leaf september {
                type boolean;
                description
                  "This is september for periodic month";
              }
              leaf october {
                type boolean;
                description
                  "This is october for periodic month";
              }
              leaf november {
                type boolean;
                description
                  "This is november for periodic month";
              }
              leaf december {
                type boolean;
                description
                  "This is december for periodic month";
              }
            }
          }
        }
      }

        container resolution-strategy {
          description
            "The resolution strategies can be used to
            specify how to resolve conflicts that occur between
            the actions of the same or different policy rules that
            are matched and contained in this particular NSF";


          choice resolution-strategy-type {
            description
              "Vendors can use YANG data model to configure rules";

            case fmr {
              leaf first-matching-rule {
                type boolean;
                description
                  "If the resolution strategy is first matching rule";
              }
            }
            case lmr {
              leaf last-matching-rule {
                type boolean;
```

```
                description
                  "If the resolution strategy is last matching rule";
              }
            }

          }
        }

        container default-action {
          description
            "This default action can be used to specify a predefined
            action when no other alternative action was matched
            by the currently executing I2NSF Policy Rule. An analogy
            is the use of a default statement in a C switch statement.";

          leaf default-action-type {
            type ingress-action;
            description
              "Ingress action type: permit, deny, and mirror.";
          }
        }
      }



    container event-clause-container {
      description "TBD";
      list event-clause-list {
      key eca-object-id;
      uses i2nsf-eca-object-type {
        refine entity-class {
            default ECA-EVENT-TYPE;
        }
      }

      description
        " This is abstract. An event is defined as any important
          occurrence in time of a change in the system being
          managed, and/or in the environment of the system being
          managed. When used in the context of policy rules for
          a flow-based NSF, it is used to determine whether the
          Condition clause of the Policy Rule can be evaluated
          or not. Examples of an I2NSF event include time and
          user actions (e.g., logon, logoff, and actions that
          violate any ACL.).";

       uses i2nsf-event-type;
       }
```

```
    }
      container condition-clause-container {
      description "TBD";
      list condition-clause-list {
        key eca-object-id;
        uses i2nsf-eca-object-type {
            refine entity-class {
                default ECA-CONDITION-TYPE;
            }
        }
        description
          " This is abstract.  A condition is defined as a set
          of attributes, features, and/or values that are to be
          compared with a set of known attributes, features,
          and/or values in order to determine whether or not the
          set of Actions in that (imperative) I2NSF Policy Rule
          can be executed or not. Examples of I2NSF Conditions
          include matching attributes of a packet or flow, and
          comparing the internal state of an NSF to a desired
          state.";

          container packet-security-condition {
            description
              "TBD";
            leaf packet-manual {
              type string;
              description
                "This is manual for packet condition.
                Vendors can write instructions for packet condition
                that vendor made";
            }

            container packet-security-mac-condition {
              description
               "The purpose of this Class is to represent packet MAC
               packet header information that can be used as part of
               a test to determine if the set of Policy Actions in
               this ECA Policy Rule should be execute or not.";

              leaf-list pkt-sec-cond-mac-dest {
                type yang:phys-address;
                description
                  "The MAC destination address (6 octets long).";
              }

              leaf-list pkt-sec-cond-mac-src {
                type yang:phys-address;
                description
```

```
                  "The MAC source address (6 octets long).";
              }

              leaf-list pkt-sec-cond-mac-8021q {
                type string;
                description
                  "This is an optional string attribute, and defines
                   The 802.1Q tab value (2 octets long).";
              }

              leaf-list pkt-sec-cond-mac-ether-type {
                type string;
                description
                  "The EtherType field (2 octets long). Values up to
                   and including 1500 indicate the size of the
                   payload in octets; values of 1536 and above
                   define which protocol is encapsulated in the
                   payload of the frame.";
              }

              leaf-list pkt-sec-cond-mac-tci {
                type string;
                description
                 "This is an optional string attribute, and defines
                  the Tag Control Information. This consists of a 3
                  bit user priority field, a drop eligible indicator
                  (1 bit), and a VLAN identifier (12 bits).";
              }
            }

            container packet-security-ipv4-condition {
              description
                "The purpose of this Class is to represent IPv4
                 packet header information that can be used as
                 part of a test to determine if the set of Policy
                 Actions in this ECA Policy Rule should be executed
                 or not.";

              leaf-list pkt-sec-cond-ipv4-header-length {
                type uint8;
                description
                  "The IPv4 packet header consists of 14 fields,
                   of which 13 are required.";
              }

              leaf-list pkt-sec-cond-ipv4-tos {
                type uint8;
                description
```

```
                "The ToS field could specify a datagram’s priority
                 and request a route for low-delay,
                 high-throughput, or highly-reliable service..";
            }

            leaf-list pkt-sec-cond-ipv4-total-length {
              type uint16;
              description
                "This 16-bit field defines the entire packet size,
                 including header and data, in bytes.";
            }

            leaf-list pkt-sec-cond-ipv4-id {
              type uint8;
              description
                "This field is an identification field and is
                 primarily used for uniquely identifying
                 the group of fragments of a single IP datagram.";
            }

            leaf-list pkt-sec-cond-ipv4-fragment {
              type uint8;
              description
                "IP fragmentation is an Internet Protocol (IP)
                 process that breaks datagrams into smaller pieces
                 (fragments), so that packets may be formed that
                 can pass through a link with a smaller maximum
                 transmission unit (MTU) than the original
                 datagram size.";
            }

            leaf-list pkt-sec-cond-ipv4-fragment-offset {
              type uint16;
              description
                "Fragment offset field along with Don’t Fragment
                 and More Fragment flags in the IP protocol
                 header are used for fragmentation and reassembly
                 of IP datagrams.";
            }

            leaf-list pkt-sec-cond-ipv4-ttl {
              type uint8;
              description
                "The ttl keyword is used to check for a specific
                 IP time-to-live value in the header of
                 a packet.";
            }
```

```
            leaf-list pkt-sec-cond-ipv4-protocol {
              type uint8;
              description
                "Internet Protocol version 4(IPv4) is the fourth
                 version of the Internet Protocol (IP).";
            }

            leaf-list pkt-sec-cond-ipv4-src {
              type inet:ipv4-address;
              description
                "Defines the IPv4 Source Address.";
            }

            leaf-list pkt-sec-cond-ipv4-dest {
              type inet:ipv4-address;
              description
                "Defines the IPv4 Destination Address.";
            }

            leaf pkt-sec-cond-ipv4-ipopts {
              type string;
              description
                "With the ipopts keyword you can check if
                 a specific ip option is set. Ipopts has
                 to be used at the beginning of a rule.";
            }

            leaf pkt-sec-cond-ipv4-sameip {
              type boolean;
              description
                "Every packet has a source IP-address and
                 a destination IP-address. It can be that
                 the source IP is the same as
                 the destination IP.";
            }

            leaf-list pkt-sec-cond-ipv4-geoip {
              type string;
              description
                "The geoip keyword enables you to match on
                 the source, destination or source and destination
                 IP addresses of network traffic and to see to
                 which country it belongs. To do this, Suricata
                 uses GeoIP API with MaxMind database format.";
            }
          }

          container packet-security-ipv6-condition {
```

```
              description
                 "The purpose of this Class is to represent packet
                 IPv6 packet header information that can be used as
                 part of a test to determine if the set of Policy
                 Actions in this ECA Policy Rule should be executed
                 or not.";

              leaf-list pkt-sec-cond-ipv6-dscp {
                type string;
                description
                  "Differentiated Services Code Point (DSCP)
                   of ipv6.";
              }

              leaf-list pkt-sec-cond-ipv6-ecn {
                type string;
                description
                  "ECN allows end-to-end notification of network
                   congestion without dropping packets.";
              }

              leaf-list pkt-sec-cond-ipv6-traffic-class {
                type uint8;
                description
                  "The bits of this field hold two values. The 6
                   most-significant bits are used for
                   differentiated services, which is used to
                   classify packets.";
              }

              leaf-list pkt-sec-cond-ipv6-flow-label {
                type uint32;
                description
                  "The flow label when set to a non-zero value
                   serves as a hint to routers and switches
                   with multiple outbound paths that these
                   packets should stay on the same path so that
                   they will not be reordered.";
              }

              leaf-list pkt-sec-cond-ipv6-payload-length {
                type uint16;
                description
                  "The size of the payload in octets,
                   including any extension headers.";
              }

              leaf-list pkt-sec-cond-ipv6-next-header {
```

```
                type uint8;
                description
                  "Specifies the type of the next header.
                   This field usually specifies the transport
                   layer protocol used by a packet's payload.";
              }

              leaf-list pkt-sec-cond-ipv6-hop-limit {
                type uint8;
                description
                  "Replaces the time to live field of IPv4.";
              }

              leaf-list pkt-sec-cond-ipv6-src {
                type inet:ipv6-address;
                description
                  "The IPv6 address of the sending node.";
              }

              leaf-list pkt-sec-cond-ipv6-dest {
                type inet:ipv6-address;
                description
                  "The IPv6 address of the destination node(s).";
              }
            }

            container packet-security-tcp-condition {
              description
                "The purpose of this Class is to represent packet
                 TCP packet header information that can be used as
                 part of a test to determine if the set of Policy
                 Actions in this ECA Policy Rule should be executed
                 or not.";

              leaf-list pkt-sec-cond-tcp-src-port {
                type inet:port-number;
                description
                  "This is a mandatory string attribute, and
                  defines the Source Port number (16 bits).";
              }

              leaf-list pkt-sec-cond-tcp-dest-port {
                type inet:port-number;
                description
                  "This is a mandatory string attribute, and
                  defines the Destination Port number (16 bits).";
              }
```

```
            leaf-list pkt-sec-cond-tcp-seq-num {
              type uint32;
              description
                "If the SYN flag is set (1), then this is the
                 initial sequence number.";
            }

            leaf-list pkt-sec-cond-tcp-ack-num {
              type uint32;
              description
                "If the ACK flag is set then the value of this
                 field is the next sequence number that the sender
                 is expecting.";
            }

            leaf-list pkt-sec-cond-tcp-window-size {
              type uint16;
              description
                "The size of the receive window, which specifies
                 the number of windows size units
                 (by default,bytes) (beyond the segment
                 identified by the sequence number in the
                 acknowledgment field) that the sender of this
                 segment is currently willing to recive.";
            }

            leaf-list pkt-sec-cond-tcp-flags {
              type uint8;
              description
                "This is a mandatory string attribute, and defines
                 the nine Control bit flags (9 bits).";
            }
          }

          container packet-security-udp-condition {
            description
             "The purpose of this Class is to represent packet UDP
              packet header information that can be used as part
              of a test to determine if the set of Policy Actions
              in this ECA Policy Rule should be executed or not.";

            leaf-list pkt-sec-cond-udp-src-port {
              type inet:port-number;
              description
                "This is a mandatory string attribute, and
                defines the UDP Source Port number (16 bits).";
            }
```

```
              leaf-list pkt-sec-cond-udp-dest-port {
                type inet:port-number;
                description
                  "This is a mandatory string attribute, and
                  defines the UDP Destination Port number (16 bits).";
              }

              leaf-list pkt-sec-cond-udp-length {
                type string;
                description
                 "This is a mandatory string attribute, and defines
                  the length in bytes of the UDP header and data
                  (16 bits).";
              }
            }

            container packet-security-icmp-condition {
              description
                "The internet control message protocol condition.";

              leaf-list pkt-sec-cond-icmp-type {
                type uint8;
                description
                  "ICMP type, see Control messages.";
              }

              leaf-list pkt-sec-cond-icmp-code {
                type uint8;
                description
                  "ICMP subtype, see Control messages.";
              }

              leaf-list pkt-sec-cond-icmp-seg-num {
                type uint32;
                description
                  "The icmp Sequence Number.";
              }
            }
          }

          container packet-payload-condition {
            description
              "TBD";
            leaf packet-payload-manual {
              type string;
              description
               "This is manual for payload condition.
               Vendors can write instructions for payload condition
```

```
               that vendor made";
            }
            leaf-list pkt-payload-content {
              type string;
              description
                "The content keyword is very important in
                 signatures. Between the quotation marks you
                 can write on what you would like the
                 signature to match.";
            }
          }
          container target-condition {
            description
              "TBD";
            leaf target-manual {
              type string;
              description
                "This is manual for target condition.
                Vendors can write instructions for target condition
                that vendor made";
            }

            container device-sec-context-cond {
              description
                "The device attribute that can identify a device,
                 including the device type (i.e., router, switch,
                 pc, ios, or android) and the device’s owner as
                 well.";

              leaf pc {
                type boolean;
                description
                  "If type of a device is PC.";
              }

              leaf mobile-phone {
                type boolean;
                description
                  "If type of a device is mobile-phone.";
              }

              leaf voip-volte-phone {
                type boolean;
                description
                  "If type of a device is voip-volte-phone.";
              }

              leaf tablet {
```

```
                type boolean;
                description
                  "If type of a device is tablet.";
              }


              leaf iot {
                type boolean;
                description
                  "If type of a device is Internet of Things.";
              }

              leaf vehicle {
                type boolean;
                description
                  "If type of a device is vehicle.";
              }
            }
          }
          container users-condition {
            description
              "TBD";
            leaf users-manual {
              type string;
              description
                "This is manual for user condition.
                Vendors can write instructions for user condition
                that vendor made";
            }

            container user{
              description
                "The user (or user group) information with which
                 network flow is associated: The user has many
                 attributes such as name, id, password, type,
                 authentication mode and so on. Name/id is often
                 used in the security policy to identify the user.
                 Besides, NSF is aware of the IP address of the
                 user provided by a unified user management system
                 via network. Based on name-address association,
                 NSF is able to enforce the security functions
                 over the given user (or user group)";

              choice user-name {
                description
                  "The name of the user.
                   This must be unique.";
```

```
                case tenant {
                  description
                    "Tenant information.";

                  leaf tenant {
                    type uint8;
                    mandatory true;
                    description
                      "User’s tenant information.";
                  }
                }

                case vn-id {
                  description
                    "VN-ID information.";

                  leaf vn-id {
                    type uint8;
                    mandatory true;
                    description
                      "User’s VN-ID information.";
                  }
                }
              }
            }
            container group {
              description
                "The user (or user group) information with which
                 network flow is associated: The user has many
                 attributes such as name, id, password, type,
                 authentication mode and so on. Name/id is often
                 used in the security policy to identify the user.
                 Besides, NSF is aware of the IP address of the
                 user provided by a unified user management system
                 via network. Based on name-address association,
                 NSF is able to enforce the security functions
                 over the given user (or user group)";

              choice group-name {
                description
                  "The name of the user.
                   This must be unique.";

                case tenant {
                  description
                    "Tenant information.";

                  leaf tenant {
```

```
                    type uint8;
                    mandatory true;
                    description
                      "User’s tenant information.";
                  }
                }

                case vn-id {
                  description
                    "VN-ID information.";

                  leaf vn-id {
                    type uint8;
                    mandatory true;
                    description
                      "User’s VN-ID information.";
                  }
                }
              }
            }

          }
          container context-condition {
            description
              "TBD";
            leaf context-manual {
              type string;
              description
                "This is manual for context condition.
                Vendors can write instructions for context condition
                that vendor made";
            }
          }
          container gen-context-condition {
            description
              "TBD";
            leaf gen-context-manual {
              type string;
              description
                "This is manual for generic context condition.
                Vendors can write instructions for generic context
                condition that vendor made";
            }

            container geographic-location {
              description
                "The location where network traffic is associated
                 with. The region can be the geographic location
```

```
                 such as country, province, and city,
                 as well as the logical network location such as
                 IP address, network section, and network domain.";

              leaf-list src-geographic-location {
                type uint32;
                description
                  "This is mapped to ip address. We can acquire
                   source region through ip address stored the
                   database.";
              }
              leaf-list dest-geographic-location {
                type uint32;
                description
                  "This is mapped to ip address. We can acquire
                   destination region through ip address stored
                   the database.";
              }
            }
          }
       }
      }
      container action-clause-container {
        description "TBD";
        list action-clause-list {
        key eca-object-id;
        uses i2nsf-eca-object-type {
          refine entity-class {
            default ECA-ACTION-TYPE;
          }
        }
        description
          "An action is used to control and monitor aspects of
           flow-based NSFs when the event and condition clauses
           are satisfied. NSFs provide security functions by
           executing various Actions. Examples of I2NSF Actions
           include providing intrusion detection and/or protection,
           web and flow filtering, and deep packet inspection
           for packets and flows.";


          container ingress-action {
            description
              "TBD";
            leaf ingress-manual {
              type string;
              description
                "This is manual for ingress action.
```

```
                Vendors can write instructions for ingress action
                that vendor made";
            }
            leaf ingress-action-type {
              type ingress-action;
              description
                "Ingress action type: permit, deny, and mirror.";
            }
          }
          container egress-action {
            description
              "TBD";
            leaf egress-manual {
              type string;
              description
                "This is manual for egress action.
                Vendors can write instructions for egress action
                that vendor made";
            }
            leaf egress-action-type {
              type egress-action;
              description
                "Egress-action-type: invoke-signaling,
                 tunnel-encapsulation, and forwarding.";
            }
          }
          container apply-profile {
            description
              "TBD";
            leaf profile-manual {
              type string;
              description
                "This is manual for apply profile action.
                Vendors can write instructions for apply
                profile action that vendor made";
            }

              container content-security-control {
                description
                 "Content security control is another category of
                 security capabilities applied to application layer.
                 Through detecting the contents carried over the
                 traffic in application layer, these capabilities
                 can realize various security purposes, such as
                 defending against intrusion, inspecting virus,
                 filtering malicious URL or junk email, and blocking
                 illegal web access or data retrieval.";
```

```
                container content-security-control-types {
                  description
                   "Content Security types: Antivirus, IPS, IDS,
                    url-filtering, data-filtering, mail-filtering,
                    file-blocking, file-isolate, pkt-capture,
                    application-control, and voip-volte.";

                  leaf antivirus {
                      type boolean;
                      description
                        "Additional inspection of antivirus.";
                  }

                  leaf ips {
                      type boolean;
                      description
                        "Additional inspection of IPS.";
                  }

                  leaf ids {
                      type boolean;
                      description
                        "Additional inspection of IDS.";
                  }

                  leaf url-filtering {
                      type boolean;
                      description
                        "Additional inspection of URL filtering.";
                  }

                  leaf data-filtering {
                     type boolean;
                     description
                       "Additional inspection of data filtering.";
                  }

                  leaf mail-filtering {
                    type boolean;
                    description
                      "Additional inspection of mail filtering.";
                  }

                  leaf file-blocking {
                    type boolean;
                    description
                      "Additional inspection of file blocking.";
                  }
```

```
                  leaf file-isolate {
                    type boolean;
                    description
                      "Additional inspection of file isolate.";
                  }

                  leaf pkt-capture {
                    type boolean;
                    description
                      "Additional inspection of packet capture.";
                  }

                  leaf application-control {
                    type boolean;
                    description
                      "Additional inspection of app control.";
                  }

                  leaf voip-volte {
                    type boolean;
                    description
                      "Additional inspection of VoIP/VoLTE.";
                  }
                }
              }

              container attack-mitigation-control {
                description
                  "This category of security capabilities is
                   specially used to detect and mitigate various
                   types of network attacks.";

                container ddos-attack {
                  description
                    "A distributed-denial-of-service (DDoS) is
                     where the attack source is more than one,
                     often thousands of unique IP addresses.";

                  container ddos-attack-type {
                    description
                      "DDoS-attack types: Network Layer
                      DDoS Attacks and Application Layer
                      DDoS Attacks.";

                    container network-layer-ddos-attack {
                      description
                        "Network layer DDoS-attack.";
                      container network-layer-ddos-attack-type {
```

```
                          description
                            "Network layer DDoS attack types:
                             Syn Flood Attack, UDP Flood Attack,
                             ICMP Flood Attack, IP Fragment Flood,
                             IPv6 Related Attacks, and etc";

                          leaf syn-flood {
                            type boolean;
                            description
                              "Additional Inspection of
                               Syn Flood Attack.";
                          }

                          leaf udp-flood {
                            type boolean;
                            description
                              "Additional Inspection of
                               UDP Flood Attack.";
                          }

                          leaf icmp-flood {
                            type boolean;
                            description
                              "Additional Inspection of
                               ICMP Flood Attack.";
                          }

                          leaf ip-frag-flood {
                            type boolean;
                            description
                              "Additional Inspection of
                               IP Fragment Flood.";
                          }

                          leaf ipv6-related {
                            type boolean;
                            description
                              "Additional Inspection of
                               IPv6 Related Attacks.";
                          }
                        }
                      }

                      container app-layer-ddos-attack {
                        description
                          "Application layer DDoS-attack.";

                        container app-ddos-attack-types {
```

```
                description
                  "Application layer DDoS-attack types:
                   Http Flood Attack, Https Flood Attack,
                   DNS Flood Attack, and
                   DNS Amplification Flood Attack,
                   SSL DDoS Attack, and etc.";

                leaf http-flood {
                  type boolean;
                  description
                    "Additional Inspection of
                     Http Flood Attack.";
                }

                leaf https-flood {
                  type boolean;
                  description
                    "Additional Inspection of
                     Https Flood Attack.";
                }

                leaf dns-flood {
                  type boolean;
                  description
                    "Additional Inspection of
                     DNS Flood Attack.";
                }

                leaf dns-amp-flood {
                  type boolean;
                  description
                    "Additional Inspection of
                     DNS Amplification Flood Attack.";
                }

                leaf ssl-ddos {
                  type boolean;
                  description
                    "Additional Inspection of
                     SSL Flood Attack.";
                }
              }
            }
          }
        }

        container single-packet-attack {
          description
```

```
                      "Single Packet Attacks.";
                    container single-packet-attack-type {
                      description
                        "DDoS-attack types: Scanning Attack,
                         Sniffing Attack, Malformed Packet Attack,
                         Special Packet Attack, and etc.";

                      container scan-and-sniff-attack {
                        description
                          "Scanning and Sniffing Attack.";
                        container scan-and-sniff-attack-types {
                          description
                            "Scanning and sniffing attack types:
                             IP Sweep attack, Port Scanning,
                             and etc.";

                          leaf ip-sweep {
                            type boolean;
                            description
                              "Additional Inspection of
                               IP Sweep Attack.";
                          }

                          leaf port-scanning {
                            type boolean;
                            description
                              "Additional Inspection of
                               Port Scanning Attack.";
                          }
                        }
                      }

                      container malformed-packet-attack {
                        description
                          "Malformed Packet Attack.";
                        container malformed-packet-attack-types {
                          description
                            "Malformed packet attack types:
                             Ping of Death Attack, Teardrop Attack,
                             and etc.";

                          leaf ping-of-death {
                            type boolean;
                            description
                              "Additional Inspection of
                               Ping of Death Attack.";
                          }
```

```
                          leaf teardrop {
                            type boolean;
                            description
                              "Additional Inspection of
                               Teardrop Attack.";
                          }
                        }
                      }

                      container special-packet-attack {
                        description
                          "special Packet Attack.";
                        container special-packet-attack-types {
                          description
                            "Special packet attack types:
                             Oversized ICMP Attack, Tracert Attack,
                             and etc.";

                          leaf oversized-icmp {
                            type boolean;
                            description
                              "Additional Inspection of
                               Oversize ICMP Attack.";
                          }

                          leaf tracert {
                            type boolean;
                            description
                              "Additional Inspection of
                               Tracrt Attack.";
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }

        <CODE ENDS>
```

Figure 5: YANG Data Module of I2NSF NSF-Facing-Interface

7. Security Considerations

This document introduces no additional security threats and SHOULD follow the security requirements as stated in [RFC8329].

8. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

9. Contributors

I2NSF is a group effort. I2NSF has had a number of contributing authors. The following are considered co-authors:

- o Hyoungshick Kim (Sungkyunkwan University)
- o Daeyoung Hyun (Sungkyunkwan University)
- o Dongjin Hong (Sungkyunkwan University)
- o Liang Xia (Huawei)
- o Tae-Jin Ahn (Korea Telecom)
- o Se-Hui Lee (Korea Telecom)

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

10.2. Informative References

[i2nsf-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-00 (work in progress), September 2017.

[i2rs-rib-data-model] Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-10 (work in progress), February 2018.

[supa-policy-info-model] Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-03 (work in progress), May 2017.

Appendix A. Changes from draft-kim-i2nsf-nsf-facing-interface-data-model-04

The following changes are made from draft-kim-i2nsf-nsf-facing-interface-data-model-04:

1. We replaced "Objectives" section with "The Structure and Objective of I2NSF Security Policy" in order to convey clearer meaning.

2. We replaced the module name for this YANG data model in order to convey clearer meaning.

3. We modified it to support not only absolute time zone but also periodic time zone.

4. We added port number to the condition clause.

5. We modified the choice-case structure into a container structure to allow for the selection of multiple catalogues for condition and action clauses.

Authors' Addresses

Jinyong Tim Kim
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 8273 0930
EMail: timkim@skku.edu

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
EMail: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 34129
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Phone: +1-734-604-0332
EMail: shares@ndzh.com

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

Phone:
EMail: linqiushi@huawei.com

I2NSF Working Group R. Kumar
Internet-Draft A. Lohiya
Intended status: Informational Juniper Networks
Expires: September 6, 2018 D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei
J. Jeong
Sungkyunkwan University
March 5, 2018

Information Model for Consumer-Facing Interface to Security Controller
draft-kumar-i2nsf-client-facing-interface-im-05

Abstract

This document defines an information model for Consumer-Facing interface to Security Controller based on the requirements identified in [I-D.ietf-i2nsf-client-facing-interface-req]. The information model defines various managed objects and relationship among these objects needed to build the interface. The information model is organized based on the "Event-Condition-Event" (ECA) policy model defined by a capability information model for Interface to Network Security Functions (I2NSF) [I-D.ietf-i2nsf-capability].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Interface to Network Security Functions (I2NSF) defines a Consumer-Facing Interface to deliver high-level security policies to Security Controller [RFC8192][RFC8329] for securiy enforcement in Network Security Functions (NSFs).

The Consumer-Facing interface would be built using a set of objects, with each object capturing a unique set of information from Security Admin (i.e., I2NSF User [RFC8329]) needed to express a Security Policy. An object may have relationship with various other objects to express a complete set of requirement. An information model captures the managed objects and relationship among these objects. The information model proposed in this document is in accordance with interface requirements as defined in [I-D.ietf-i2nsf-client-facing-interface-req].

An NSF Capability model is proposed in [I-D.ietf-i2nsf-capability] as the basic model for both the NSF-Facing interface and Consumer-Facing interface security policy model of this document. The information model proposed in this document is structured in accordance with the "Event-Condition-Event" (ECA) policy model.

[RFC3444] explains differences between an information and data model. This document use the guidelines in [RFC3444] to define an information model for Consumer-Facing interface in this document. A data model, which represents an implementation of the proposed information model in a specific data representation language, will be defined in a separate document.

2. Conventions Used in this Document

BSS: Business Support System

CLI: Command Line Interface

CMDB: Configuration Management Database

Controller: Security Controller or Management System

CRUD: Create, Retrieve, Update, Delete

FW: Firewall

GUI: Graphical User Interface

IDS: Intrusion Detection System

IPS: Intrusion Prevention System

LDAP: Lightweight Directory Access Protocol

NSF: Network Security Function, defined by [I-D.ietf-i2nsf-terminology]

OSS: Operations Support System

RBAC: Role-Based Access Control

SIEM: Security Information and Event Management

URL: Universal Resource Locator

vNSF: NSF being instantiated on Virtual Machines

3. Information Model for Policy

A Policy object represents a mechanism to express a Security Policy by Security Admin (i.e., I2NSF User) using Consumer-Facing interface toward Security Controller; the policy would be enforced on an NSF. The Policy object SHALL have following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Multi-Tenancy: The multi-tenant environment information in which the policy is applied. The Rules in the Policy can refer to sub-objects (e.g., domain, tenant, role, and user) of it. It can be either a reference to a Multi-Tenancy object defined in another place, or a concrete object. See details in Section 4.

End-Group: This field contains a list of logical entities in the business environment where a Security Policy is to be applied. It can be referenced by the Condition objects in a Rule, e.g., Source, Destination, Match, etc. It can be either a reference to an End-Group object defined in other place, or a concrete object. See details in Section 5.

Threat-Feed: This field represents threat feed such as Botnet servers, GeoIP, and Malware signature. This information can be referenced by the Rule Action object directly to execute the threat mitigation. See details in Section 6.

Telemetry-Data: This field represents the telemetry collection related information that the Rule Action object can refer to about how to collect the interested telemetry information, for example, what type of telemetry to collect, where the telemetry source is, where to send the telemetry information. See details in Section 7.

Rules: This field contains a list of rules. If the rule does not have a user-defined precedence, then any conflict must be manually resolved.

Owner: This field defines the owner of this policy. Only the owner is authorized to modify the contents of the policy.

A policy is a container of Rules. In order to express a Rule, a Rule must have complete information such as where and when a policy needs to be applied. This is done by defining a set of managed objects and relationship among them. A Policy Rule may be related segmentation, threat mitigation or telemetry data collection from an NSF in the network, which will be specified as the sub-model of the policy model in the subsequent sections.

The rule object SHALL have the following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event: This field includes the information to determine whether the Rule Condition can be evaluated or not. See details in Section 3.1.

Condition: This field contains all the checking conditions to apply to the objective traffic. See details in Section 3.2.

Action: This field identifies the action taken when a rule is matched. There is always an implicit action to drop traffic if no rule is matched for a traffic type. See details in Section 3.3.

Precedence: This field identifies the precedence assigned to this rule by Security Admin. This is helpful in conflict resolution when two or more rules match a given traffic class.

3.1. Event Sub-Model

The Event Object contains information related to scheduling a Rule. The Rule could be activated based on a time calendar or security event including threat level changes.

Event object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event-Type: This field identifies whether the event of triggering policy enforcement is "ADMIN-ENFORCED", "TIME-ENFORCED" or "EVENT-ENFORCED".

Time-Information: This field contains a time calendar such as "BEGIN-TIME" and "END-TIME" for one time enforcement or recurring time calendar for periodic enforcement.

Event-Map-Group: This field contains security events or threat map in order to determine when a policy needs to be activated. This is a reference to Event-Map-Group defined later.

3.1.1. Event-Map-Group

This object represents an event map containing security events and threat levels used for dynamic policy enforcement. The Event-Map-Group object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Security-Events: This contains a list of security events used for purpose for Security Policy definition.

Threat-Map: This contains a list of threat levels used for purpose for Security Policy definition.

3.2. Condition Sub-Model

This object represents Conditions that Security Admin wants to apply the checking on the traffic in order to determine whether the set of actions in the Rule can be executed or not.

The Condition object SHALL have following information:

Source: This field identifies the source of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry-Source for telemetry collection policy.

Destination: This field identifies the destination of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry- Destination for telemetry collection policy.

Match: This field identifies the match criteria used to evaluate whether the specified action needs to be taken or not. This could be either a Policy-Endpoint-Group identifying an Application set or a set of traffic rules.

Match-Direction: This field identifies whether the match criteria is to be evaluated for both directions or only one direction of the traffic with a default of allowing the other direction for stateful match conditions. This is optional and by default a rule should apply to both directions.

Exception: This field identifies the exception consideration when a rule is evaluated for a given communication. This could be a reference to "Policy-Endpoint-Group" object or set of traffic matching criteria.

3.3. Action Sub-Model

This object represents actions that Security Admin wants to perform based on certain traffic class.

The Action object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Primary-Action: This field identifies the action when a rule is matched by an NSF. The action could be one of "PERMIT", "DENY", "MIRROR", "REDIRECT", "RATE-LIMIT", "TRAFFIC-CLASS", "AUTHENTICATE-SESSION", "IPS", "APP-FIREWALL", or "COLLECT".

Secondary-Action: Security Admin can also specify additional actions if a rule is matched. This could be one of "LOG", "SYSLOG", or "SESSION-LOG".

4. Information Model for Multi Tenancy

Multi-tenancy is an important aspect of any application that enables multiple administrative domains in order to manage application resources. An Enterprise organization may have multiple tenants or departments such as Human Resources (HR), Finance, and Legal, with each tenant having a need to manage their own Security Policies. In a Service Provider, a tenant could represent a Customer that wants to manage its own Security Policies.

There are multiple managed objects that constitute multi-tenancy aspects. This section lists these objects and any relationship among these objects.

4.1. Policy-Domain

This object defines a boundary for the purpose of policy management within a Security Controller. This may vary based on how the Security Controller is deployed and hosted. For example, if an Enterprise hosts a Security Controller in their network; the domain in this case could just be the one that represents that Enterprise. But if a Cloud Service Provider hosts managed services, then a domain could represent a single customer of that Provider. Multi-tenancy model should be able to work in all such environments.

The Policy-Domain object SHALL have following information:

Name: Name of the organization or customer representing this domain.

Address: Address of the organization or customer.

Contact: Contact information of the organization or customer.

Date: Date when this account was created or last modified.

Authentication-Method: Authentication method to be used for this domain. It should be a reference to a 'Policy-Management-Authentication-Method' object.

4.2. Policy-Tenant

This object defines an entity within an organization. The entity could be a department or business unit within an Enterprise organization that would like to manage its own Policies due to regulatory compliance or business reasons.

The Policy-Tenant object SHALL have following information:

Name: Name of the Department or Division within an organization.

Date: Date when this account was created or last modified.

Domain: This field identifies the domain to which this tenant belongs. This should be a reference to a Policy-Domain object.

4.3. Policy-Role

This object defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. It provides a convenient way to assign policy users to a job function or a set of permissions within the organization.

The Policy-Role object SHALL have the following information:

Name: This field identifies the name of the role.

Date: Date when this role was created or last modified.

Access-Profile: This field identifies the access profile for the role. The profile grants or denies the permissions to access Endpoint Groups for the purpose of policy management or may restrict certain operations related to policy managements.

4.4. Policy-User

This object represents a unique identity within an organization. The identity authenticates with Security Controller using credentials such as a password or token in order to perform policy management. A user may be an individual, system, or application requiring access to Security Controller.

The Policy-User object SHALL have the following information:

Name: Name of a user.

Date: Date when this user was created or last modified.

Password: User password for basic authentication.

Email: E-mail address of the user.

Scope-Type: This field identifies whether the user has domain-wide or tenant-wide privileges.

Scope-Reference: This field should be a reference to either a Policy-Domain or a Policy-Tenant object.

Role: This field should be a reference to a Policy-Role object that defines the specific permissions.

4.5. Policy-Management-Authentication-Method

This object represents authentication schemes supported by Security Controller.

This Policy-Management-Authentication-Method object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Authentication-Method: This field identifies the authentication methods. It could be a password-based, token-based, certificate-based or single sign-on authentication.

Mutual-Authentication: This field indicates whether mutual authentication is mandatory or not.

Token-Server: This field stores the information about server that validates the token submitted as credentials.

Certificate-Server: This field stores the information about server that validates certificates submitted as credentials.

Single Sign-on-Server: This field stores the information about server that validates user credentials.

5. Information Model for Policy Endpoint Groups

The Policy Endpoint Group is a very important part of building User-construct based policies. Security Admin would create and use these objects to represent a logical entity in their business environment, where a Security Policy is to be applied.

There are multiple managed objects that constitute a Policy Endpoint Group. This section lists these objects and relationship among these objects.

5.1. Tag-Source

This object represents information source for tag. The tag in a group must be mapped to its corresponding contents to enforce a Security Policy.

Tag-Source object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Tag-Type: This field identifies the Endpoint Group type. It can be a User-Group, App-Group, Device-Group or Location-Group.

Tag-Source-Server: This field identifies information related to the source of the tag such as IP address and UDP/TCP port information.

Tag-Source-Application: This filed identifies the protocol, e.g., LDAP, Active Directory, or CMDB used to communicate with a server.

Tag-Source-Credentials: This field identifies the credential information needed to access the server.

5.2. User-Group

This object represents a user group based on either tag or other information.

The User-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the user group is based on User-tag, User-name or IP-address.

Metadata-Server: This field should be a reference to a Metadata-Source object.

Group-Member: This field is a list of User-tag, User-names or IP addresses based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.3. Device-Group

This object represents a device group based on either tag or other information.

The Device-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the device group is based on Device-tag or Device-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Device-tag, Device-name or IP address based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.4. Application-Group

This object represents an application group based on either tag or other information.

The Application-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the application group is based on App-tag or App-name, or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Application-tag Application-name or IP address and port information based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.5. Location-Group

This object represents a location group based on either tag or other information.

The 'Location-Group' object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the location group is based on Location-tag, Location-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Location-tag, Location-name or IP addresses based on Group-Type.

Risk Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

6. Information Model for Threat Prevention

The threat prevention plays an important part in the overall security posture by reducing the attack surfaces. This information could come in the form of threat feeds such as Botnet and GeoIP feeds usually from a third party or external service.

There are multiple managed objects that constitute this category. This section lists these objects and relationship among these objects.

6.1. Threat-Feed

This object represents a threat feed such as Botnet servers and GeoIP.

The Threat-Feed object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Feed-Type: This field identifies whether a feed type is IP address-based or URL-based.

Feed-Server: This field identifies the information about the feed provider, it may be an external service or local server.

Feed-Priority: This field represents the feed priority level to resolve conflict if there are multiple feed sources; the valid range may be 0 to 9.

6.2. Custom-List

This object represents a custom list created for the purpose of defining exception to threat feeds. An organization may want to allow a certain exception to threat feeds obtained from a third party

The Custom-List object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

List-Type: This field identifies whether the list type is IP address-based or URL-based.

List-Property: This field identifies the attributes of the list property, e.g., Blacklist or Whitelist.

List-Content: This field contains contents such as IP addresses or URL names.

6.3. Malware-Scan-Group

This object represents information needed to detect malware. This information could come from a local server or uploaded periodically from a third party.

The Malware-Scan-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Signature-Server: This field contains information about the server from where signatures can be downloaded periodically as updates become available.

File-Types: This field contains a list of file types needed to be scanned for the virus.

Malware-Signatures: This field contains a list of malware signatures or hash values.

7. Information Model for Telemetry Data

Telemetry provides System Admin with the visibility of the network activities which can be tapped for further security analytics, e.g., detecting potential vulnerabilities, malicious activities, etc.

7.1. Telemetry-Data

This object contains information collected for telemetry.

The Telemetry-Data object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Log-Data: This field identifies whether Log data need to be collected.

Syslog-Data This field identifies whether Syslog data need to be collected.

SNMP-Data: This field identifies whether SNMP traps and alarm data need to be collected.

sFlow-Record: This field identifies whether sFlow records need to be collected.

NetFlow-Record: This field identifies whether NetFlow record need to be collected.

NSF-Stats: This field identifies whether statistics need to be collected from an NSF.

7.2. Telemetry-Source

This object contains information related to telemetry source. The source would be an NSF in the network.

The Telemetry-Source object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Source-Type: This field contains the type of the NSF telemetry source: "NETWORK-NSF", "FIREWALL-NSF", "IDS-NSF", "IPS-NSF", "PROXY-NSF or "OTHER-NSF".

NSF-Source: This field contains information such as IP address and protocol (UDP or TCP) port number of the NSF providing telemetry data.

NSF-Credentials: This field contains a username and a password used to authenticate the NSF.

Collection-Interval: This field contains time in milliseconds between each data collection. For example, a value of 5,000 means data is streamed to collector every 5 seconds. Value of 0 means data streaming is event-based.

Collection-Method: This field contains a method of collection whether it is PUSH-based or PULL-based.

Heartbeat-Interval: This field contains time in seconds when the source must send telemetry heartbeat.

QoS-Marking: This field contains a DSCP value source marked on its generated telemetry packets.

7.3. Telemetry-Destination

This object contains information related to telemetry destination. The destination is usually a collector which is either a part of Security Controller or external system such as SIEM.

The Telemetry-Destination object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Collector-Source: This field contains the information such as IP address and protocol (UDP or TCP) port number for the collector’s destination.

Collector-Credentials: This field contains a username and a password provided by the collector.

Data-Encoding: This field contains the telemetry data encoding, which could in the form of a schema.

Data-Transport: This field contains streaming telemetry data protocols: whether it is gRPC, protocol buffer over UDP, etc.

8. Security Considerations

An information model provides a mechanism to protect Consumer-Facing interface between System Admin (i.e., I2NSF User) and Security Controller. One of the specified mechanism must be used to protect an Enterprise network, data and all resources from external attacks. This information model mandates that the interface must have proper authentication and authorization with Role-Based Access Controls to address the multi-tenancy requirement. The document does not mandate that a particular mechanism should be used because a different organization may have different needs based on their deployment.

9. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

10. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

11. Contributors

This document is the work of I2NSF working group, greatly benefiting from inputs and suggestions by Kunal Modasiya, Prakash T. Sehsadri

and Srinivas Nimmagadda from Juniper Networks. The authors sincerely appreciate their contributions.

The following are contributing authors of this document, who are considered co-authors:

- o Eunsoo Kim (Sungkyunkwan University)

12. Informative References

[I-D.ietf-i2nsf-capability]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-00 (work in progress), September 2017.

[I-D.ietf-i2nsf-client-facing-interface-req]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-04 (work in progress), January 2018.

[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-05 (work in progress), January 2018.

[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>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

Appendix A. Changes from draft-kumar-i2nsf-client-facing-interface-im-04

The following changes have been made from draft-kumar-i2nsf-client-facing-interface-im-04:

- o In Section 3.2, the description of Match-Direction was corrected.

- o References are updated.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rakeshkumarcloud@gmail.com

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Nabil Bitar
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: nabil.bitar@nokia.com

Senad Palislamovic
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: senad.palislamovic@nokia.com

Liang Xia
Huawei
101 Software Avenue
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957
Fax: +82 31 290 7996
Email: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Remote Attestation Procedures for Network Security Functions (NSFs) through the I2NSF Security Controller
draft-pastor-i2nsf-nsf-remote-attestation-07

Abstract

This document describes the procedures a client can follow to assess the trust on an external NSF platform and its client-defined configuration through the I2NSF Security Controller. The procedure to assess trustworthiness is based on a remote attestation of the platform and the NSFs running on it performed through a Trusted Platform Module (TPM) invoked by the Security Controller.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

As described in [RFC8192], the use of externally provided NSF implies several additional concerns in security. The most relevant threats associated with a externalized platform are detailed in [RFC8329]. As stated there, mutual authentication between the user and the NSF environment and, more importantly, the attestation of the components in this environment by clients, could address these threats and provide an acceptable level of risk. In particular:

- Client impersonation will be minimized by mutual authentication, and since appropriate records of such authentications will be made available, events are suitable for auditing (as a minimum) in the case of an incident.

- Attestation of the NSF environment, especially when performed periodically, will allow clients to detect the alteration of the processing components, or the installation of malformed components. Mutual authentication will again provide an audit trail.

- Attestation relying on independent Trusted Third Parties will alleviate the impact of malicious activity on the side of the provider by issuing the appropriate alarms in the event of any NSF environment manipulation.

- While it is true that any environment is vulnerable to malicious activity with full physical access (and this is obviously beyond the scope of this document), the application of attestation mechanisms raises the degree of physical control necessary to perform an untraceable malicious modification of the environment.

The client can have a proof that their NSFs and policies are correctly (from the client point of view) enforced by the Security Controller. Taking into account the threats identified in [RFC8329], this document first identifies the user expectations regarding remote trust establishment, briefly analyzes Trusted Computing techniques, and finally describes the proposed mechanisms for remote establishment of trust through the Security Controller.

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Establishing Client Trust

From a high-level standpoint, in any I2NSF platform, the client connects and authenticates to the Security Controller, which then initializes the procedures for authentication and authorization (and likely accounting and auditing) to track the loading and unloading of the client's NSFs, addressing the verification of the whole software stack: firmware, (host and guest) OSes, NSFs themselves and, in a virtualized environment, the virtualization system (hypervisors, container frameworks...). Afterwards, user traffic from the client domain goes through the NSF platform that hosts the corresponding NSFs. The user's expectations of the platform behavior are thus twofold:

- o The user traffic will be treated according to the client-specified NSFs, and no other processing will be performed by the Security Controller or the platform itself (e.g. traffic eavesdropping).
- o Each NSF (and its corresponding policies) behaves as configured by the client.

We will refer to the attestation of these two expectations as the "client-agnostic attestation" and the "client-specific attestation". Trusted Computing techniques play a key role in addressing these expectations.

3.1. First Step: Client-Agnostic Attestation

This is the first interaction between a client and a Security Controller: the client wants to attest that he is connected to a genuine Security Controller before continuing with the authentication. In this context, two properties characterize the genuineness of the Security Controller:

1. That the identity of the Security Controller is correct
2. That it will process the client credentials and set up the client NSFs and policies properly.

Once these two properties are proven to the client, the client knows that their credentials will only be used by the Security Controller to set up the execution of their NSFs.

3.2. Second Step: Client-Specific Attestation

From the security enforcement point of view, the client agnostic attestation focuses on the initialization of the execution platform for the NSFs. This second step aims to prove to clients that their security is enforced accordingly with their choices (i.e. NSFs and policies). The attestation can be performed at the initialization of the NSFs, before any user traffic is processed by the NSFs, and optionally during the execution of the NSFs.

Support of static attestation, performed at initialization time, for the execution platform and the NSFs is REQUIRED for a Security Controller managing NSFs, and MUST be performed before any user traffic is redirected through any set of NSFs. The Security Controller MUST provide proof to the client that the instantiated NSFs and policies are the ones chosen.

In addition to the platform and executable component attestation, the infrastructure network topology supporting the NSFs may need to attested, in order to assess the enforcement of the security policies requested by the client. Whilst platform and NSF attestation can be considered sufficient in I2NSF environments in which network elements are connected following a fairly static configuration, the dynamicity brought by networking techniques such as NFV, SDN and SFC make attestation of dynamic topology network topologies a desirable feature in a number of cases. Depending on the level of asurance desired, the client MAY request the Security Controller proof of the network topology connecting the instantiated NSFs.

Additionally to the NSFs instantiation attestation, a continuous attestation of the Security Controller and the NSF execution MAY be required by a client to ensure their security. The sampling periods for the continuous attestation of NSFs an Controller MAY be different.

3.3. Trusted Computing

In a nutshell, Trusted Computing (TC) aims at answering the following question: "As a user or administrator, how can I have some assurance that a computing system is behaving as it should?". The major enterprise level TC initiative is the Trusted Computing Group [TCG], which has been established for more than a decade, that primarily focuses on developing TC for commodity computers (servers, desktops, laptops, etc.).

The overall scheme proposed by TCG for using Trusted Computing is based on a step-by-step extension of trust, called a Chain of Trust. It uses a transitive mechanism: if a user can trust the first

execution step and each step correctly attests the next executable
software for trustworthiness, then a user can trust the system.

```
                +-----------+
                |           |    extends PCR
                | Platform  +------------------------+
                |           |                        |
                +-----^-----+                        |
                      |                              |
                      |measures                      |
                +-----------+                        |
                | Security  |    extends PCR         |
                |           +---------------------+  |
                | Controller|                     |  |
                +-----^-----+                     |  |
                      |                           |  |
                      |measures                 +-v--v----------+
                +-----------+                   |               |
                |           |    extends PCR    |               |
                | Bootloader+------------------->  Root of Trust |
                |           |                   |               |
                +-----^-----+                   |               |
                      |                         +-^--^----------+
                      |measures                   |  |
                +-----------+                     |  |
                |           |    extends PCR      |  |
                | BIOS      +---------------------+  |
                |           |                        |
                +-----^-----+                        |
                      |                              |
                      |measures                      |
                +-----------+                        |
                | Bootblock |    extends PCR         |
                |  (CRTM)   +------------------------+
                |           |
                +-----------+
```

Figure 1: Applying Trusted Computing

Effectively, during the loading of each piece of software, the
integrity of each piece of software is measured and stored inside a
log that reflects the different boot stages, as illustrated in the
figure above. Later, at the request of a user, the platform can
present this log (signed with the unique identity of the platform),
which can be checked to prove the platform identity and attest the
state of the system. The base element for the extension of the Chain
of Trust is called the Core Root of Trust.

The TCG has created a standard for the design and usage of a secure crypto-processor to address the storage of keys, general secrets, identities, and platform integrity measurements: the Trusted Platform Module (TPM). When using a TPM as a root of trust, measurements of the software stack are stored in special on-board Platform Configuration Registers (PCRs) on a discrete TPM. There are normally a small number of PCRs that can be used for storing measurements; however, it is not possible to directly write to a PCR. Instead, measurements must be stored using a process called Extending PCRs.

The extend operation can update a PCR by producing a global hash of the concatenated values of the previous PCR value with the new measurement value. The Extend operation allows for an unlimited number of measurements to be captured in a single PCR, since the size of the value is always the same and it retains a verifiable ordered chain of all the previous measurements.

Attestation of the virtualization platform will thus rely on a process of measuring the booted software and storing a chained log of measurements, typically referred to as Trusted Boot. The user will either validate the signed set of measurements with a trusted third party verifier who will assess whether the software configuration is trusted, or the user can check for themselves against their own set of reference digest values (measurements) that they have obtained a priori, and having already known the public endorsement key of the remote Root of Trust.

Trusted Boot should not be confused with a different mechanism known as "Secure Boot", as they both are designed to solve different problems. Secure Boot is a mechanism for a platform owner to lock a platform to only execute particular software. Software components that do not match the configuration digests will not be loaded or executed. This mechanism is particularly useful in preventing malicious software that attempts to install itself in the boot record (a bootkit) from successfully infecting a platform on reboot. A common standard for implementing Secure Boot is described in [UEFI]. Secure Boot only enforces a particular configuration of software, it does not allow a user to attest or quote for a series of measurements.

3.4. Topology Attestation

There are two methods able to attest the deployment of a topology addressing client requirements on a dynamically controled network infrastructure. The first one asumes the newtork infrastructure is built by means of SDN-enabled fowarding elements, and the second relies on the application of SFC [RFC7665] to build the NSF processing paths. In both cases, a network topology verifier is used.

In the first case, a SDN verifier is introduced, and network forwarding elements required to provide attestation features, as described in the previous section, to provide measures on the enforced SDN configuration. The SDN verifier retrieves from the SDN controller the configuration for the attested network elements, challenges them for their SDN configuration, and assessesit is consistent with the expected SDN configuration retrieved from the SDN controller. The SDN verifier on the network elements leverage a TPM, with the network element implementing a regular measured boot.

The second option considers the application of Proof of Transit (POT) [I-D.ietf-sfc-proof-of-transit] to a SFC-based network, where the NSFs act as service functions. A SFC verifier can inject specific packets requesting POT, and verifying it at the egress of the service path to assess a correct topoloogy is being enforced, by means of the cryptographic proof provided by POT.

4. NSF Attestation Principles

Following the general requirements described in [RFC8329] the Security Controller will become the essential element to implement the measurements described above, relying on a TPM for the Root of Trust.

A mutual authentication of clients and the Security Controller MUST be performed, establishing the desired level of assurance. This level of assurance will determine how stringent are the requirements for authentication (in both directions), and how detailed the collected measurements and their verification will be. Furthermore, the NSF platform MUST run a TPM, able to collect measurements of the platform itself, the Security Controller, and the NSFs being executed. The availability of a network topology verifier is OPTIONAL, though a client MAY require it to fulfill the required level of assurance. The Security Controller MUST make the attestation measurements available to the client, directly or by means of a Trusted Third Party.

As described in [RFC8329], a trusted connection between the client and the Security Controller MUST be established and all traffic to and from the NSF environment MUST flow through this connection

NOTE: The reference to results from WGs such as NEA and SACM is currently under consideration and will be included here.

4.1. Requirements for a Trusted NSF Platform

Although a discrete hardware TPM is RECOMMENDED, relaxed alternatives (such as embedded CPU TPMs, or memory and execution isolation mechanisms) MAY also be applied when the required level of assurance is lower. This reduced level of assurance MUST be communicated to the client by the Security Controller during the initial mutual authentication phase. The Security Controller MUST use a set of capabilities to negotiate the level of assurance with the client.

4.1.1. Trusted Boot

NOTE: This section is derived from the original version of the document, focused on virtual NSFs. Although it seems to be applicable to any modern physical appliance, we must be sure all these considerations are 100% applicable to physical NSFs as well, and provide exceptions when that is not the case. Support from an expert in physical node attestation is required here.

All clients who interact with a Security Controller MUST be able to:

a. Identify the Security Controller based on the public key of a Root of Trust.

b. Retrieve a set of measurements of all the base software the Security Controller has booted (i.e. the NSF platform).

This requires that firmware and software MUST be measured before loading, with the resulting value being used to extend the appropriate PCR register. The general usage of PCRs by each software component SHOULD conform to open standards, in order to make verifying attestation reports interoperable, as it is the case of TCG Generic Server Specification [TCGGSS].

The following list describes which PCR registers SHOULD be used during a Trusted Boot process:

- PCRs 00-03: for use by the CRTM (Core Root of Trust for Measurement, at the initial EEPROM or PC BIOS)

- PCRs 04-07: for use by the bootloader stages

- PCRs 08-15: for use by the booted base system

A signed audit log of boot measurements should also be provided. The PCR values can also be used as an identity for dynamically decrypting encrypted blobs on the platform (such as encryption keys or configurations that belong to operating system components). Software can choose to submit pieces of data to be encrypted by the Root of Trust (which has its own private asymmetric key and PCR registers) and only have it decrypted based on some criteria. These criteria can be that the platform booted into a particular state (e.g. a set of PCR values). Once the desired criteria are described and the sensitive data is encrypted by the root of trust, the data has been sealed to that platform state. The sealed data will only be decrypted when the platform measurements held in the root of trust match the particular state.

Trusted Boot requires the use of a root of trust for safely storing measurements and secrets. Since the Root of Trust is self-contained and isolated from all the software that is measured, it is able to produce a signed set of platform measurements to a local or remote user. However, Trusted Boot does not provide enforcement of a configuration, since the root of trust is a passive component not in the execution path, and is solely used for safe independent storage of secrets and platform measurements. It will respond to attestation requests with the exact measurements that were made during the software boot process. Sealing and unsealing of sensitive data is also a strong advantage of Trusted Boot, since it prevents leakage of secrets in the event of an untrusted software configuration.

4.1.2. Remote Attestation Service

A service MUST be present for providing signed attestation report (e.g. the measurements) from the Root of Trust (RoT) to the client. In case of failure to communicate with the service, the client MUST assume the service cannot be trusted and seek an alternative Security Controller.

Since some forms of RoT require serialised access (i.e. due to slow access to hardware), latency of getting an attestation report could increase with simultaneous requests. Simultaneous requests could occur if multiple Trusted Third Parties (TTP) request attestation reports at the same time. This MAY be improved through batching of requests, in a special manner. In a typical remote attestation protocol, the client sends a random number ("nonce") to the RoT in order to detect any replay attacks. Therefore, caching of an attestation report does not work, since there is the possibility that it may not be a fresh report. The solution is to batch the nonce for each requestor until the RoT is ready for creating the attestation report. The report will be signed by the embedded identity of the RoT to provide data integrity and authenticity, and the report will include all the nonces of the requestors. Regardless of the number of the number of nonces included, the requestor verifying the attestation report MUST check to see if the requestor's nonce was included in order to detect replay attacks. In addition to the attestation report containing PCRs, an additional report known as an SML (Secure Measurement Log) can be returned to the requestor to provide more information on how to verify the report (e.g. how to reproduce the PCR values). The integrity of the SML is protected by a PCR measurement in the RoT. An example of an open standard for responses is [TCGIRSS]. Further details are discussed in Section 5.2.

As part of initial contact, the Security Controller MAY present a list of external TTPs that the client can use to verify it. However, the client MUST assess whether these external verifiers can be trusted. The client can also choose to ignore or discard the presented verifiers.

If available, the network topology verifier MUST be colocated or integrated with the RoT.

Finally, to prevent malicious relaying of attestation reports from a different host, the authentication material of the secure channel (e.g. TLS, IPSec, etc.) SHOULD be bound to the RoT and verified by the connected client, unless the lowest levels of assurance have been chosen and an explicit warning issued. This is also addressed in Section 5.1.

4.1.3. Secure Boot

Using a mechanism such as Secure Boot helps provide strong prevention of software attacks. Furthermore, in combination with a hardware-based TPM, Secure Boot can provide some resilience to physical attacks (e.g. preventing a class of offline attacks and unauthorized system replacement). For NSF providers, it is RECOMMENDED that Secure Boot is employed wherever possible with an appropriate firmware update mechanism, due to the possible threat of software/firmware modifications in either public places or privately with inside attackers.

5. Remote Attestation Procedures

The establishment of trust with the Security Controller and the NSF platform consists of three main phases, which need to be coordinated by the client:

1. Trusted channel with the Security Controller. During this phase, the client securely connects to the Security Controller to avoid that any data can be tampered with or modified by an attacker if the network cannot be considered trusted. The establishment of the trusted channel is completed after the next step.

2. Security Controller attestation. During this phase, the client verifies that the Security Controller components responsible for handling the credentials and for the isolation with respect to other potential clients are behaving correctly. Furthermore, it is verified that the identity of the platform attested is the same of the one presented by the Security Controller during the establishment of the secure connection.

3. Platform attestation. During this step, which can be repeated periodically until the connection is terminated, the Security Controller verifies the integrity of the elements composing the NSF platform. The components responsible for this task have been already attested during the previous phase.

```
                                   +----------+
                 3. Attestation    | Trusted  |   3. Attestation
              +--------------------> Third    <----------+
              |                    | Party    |          |
              |                    +----------+ +--------+-------+
   +----------v-------+                         | +-----v-----+  |
   |      Client      |                         | | Security  |  |
   |                  |  1. Trusted channel     | | Controller|  |
   | 2. Get Cert      +------+ handshake +--------->          |  |
   | 3. Attestation   |                         | +-----------+  |
   | 4. Cont.handshake|                         |                |
   |                  |                         |                |
   |                  |                         | +---------+    |
   |                  |                         | |   NSF   |    |
   |                  |                         | +---------+    |
   +------------------+                         +----------------+
```

Figure 2: Steps for remote attestation

In the following each step, as depicted in the above figure, is discussed in more detail.

5.1. Trusted Channel with the Security Controller

A trusted channel is an enhanced version of the secured channel that. It adds the requirement of integrity verification of the contacted endpoint by the other peer during the initial handshake to the functionality of the secured channel. However, simply transmitting the integrity measurements over the channel does not guarantee that the platform verified is the channel endpoint. The public key or the certificate for the secure communication MUST be included as part of the measurements presented by the contacted endpoint during the remote attestation. This way, a malicious platform cannot relay the attestation to another platform as its certificate will not be present in the measurements list of the genuine platform.

In addition, the problem of a potential loss of control of the private key must be addressed (a malicious endpoint could prove the identity of the genuine endpoint). This is done by defining a long-lived Platform Property Certificate. Since this certificate connects the platform identity to the AIK public key, an attacker cannot use a stolen private key without revealing his identity, as it may use the certificate of the genuine endpoint but cannot create a quote with the AIK of the other platform.

Finally, since the platform identity can be verified from the Platform Property Certificate, the information in the certificate to be presented during the establishment of a secure communication is redundant. This allows for the use of self-signed certificates. This would simplify operational procedures in many environments, especially when they are multi-tenant. Thus, in place of certificates signed by trusted CAs, the use of self-signed certificates (which still need to be included in the measurements list) is RECOMMENDED.

The steps required for the establishment of a trusted channel with the Security Controller are as follows:

1. The client begins the trusted channel handshake with the selected Security Controller.

2. The certificate of the Security Controller is collected and used for verifying the binding of the attestation result to the contacted endpoint.

3. The client performs the remote attestation protocol with the Security Controller, either directly or with the help of a Trusted Third Party. The Trusted Third Party MAY perform the verification of attestation quotes on behalf of multiple clients.

4. If the result of the attestation is positive, the application continues the handshake and establishes the trusted channel. Otherwise, it closes the connection.

5.2. Security Controller Attestation

During the establishment of the trusted channel, the client attests the Security Controller by verifying the identity of the contacted endpoint and its integrity. Initially the Security Controller measures all of the hardware and software components involved in the boot process of the NSF platform, in order to build the chain of trust.

Since a client may not have enough functionality to perform the integrity verification of a Security Controller, the client MAY request the status of a Security Controller to be computed by a Trusted Third Party (TTP). This choice has the additional advantage of preventing an attacker from easily determining the software running at the Security Controller.

If the client directly performs the remote attestation, it executes the following steps:

1. Ask the Security Controller to generate an integrity report with the format defined in [TCGIRSS].

2. The Security Controller retrieves the measurements and asks the TPM to sign the PCRs with an Attestation Identity Key (AIK). This signature provides the client with the evidence that the measurements received belong to the Security Controller being attested.

3. Once the integrity report has been generated it is sent back to the client.

4. The client first checks if the integrity report is valid by verifying the quote and the certificate associated to the AIK, and then determines if the Security Controller is behaving as expected (i.e. its software has not been compromised and isolation among the clients connected to it is enforced). As part of the verification, the client also checks that the digest of the certificate, received during the trusted channel handshake, is present among measurements.

If the client has limited computation resources, it may contact a TTP which, in turn, attests the Security Controller and returns the result of the integrity evaluation to the client, following the same steps depicted above.

5.3. Platform Attestation

The main outcome of the Security Controller attestation is to detect whether or not it is correctly configuring the operational environment for NSFs to be managed by the connecting client (the NSF platform, or just platform) in a way that any user traffic is processed only by these NSFs that are part of the platform. Platform attestation, instead, evaluates the integrity of the NSFs running on the platform.

Platform attestation does not imply a validation of the mechanisms the Security Controller can apply to select the appropriate NSFs to enforce the Service Policies applicable to specific flows. The selection of these NSFs is supposed to happen independent of the attestation procedures, and trust on the selection process and the translation of policies into function capabilities has to be based on the trust clients have on the Security Controller being attested as the one that was intended to be used. An attestation of the selection and policy mapping procedures constitute an interesting research problem, but it is out of the scope of this document.

The procedures are essentially similar to the ones described in the previous section. This step MAY be applied periodically if the level of assurance selected by the user requires it.

Attesting NSFs, especially if they are running as virtual machines, can become a costly operation, especially if periodic monitoring is required by the requested level of assurance. There are several proposals to make this feasible, from the proposal of virtual TPMs in [VTPM] to the application of Virtual Machine Introspection through an integrity monitor described by [VMIA].

6. Security Considerations

This document is specifically oriented to security and it is considered along the whole text.

7. IANA Considerations

This document requires no IANA actions.

8. Acknowledgments

This work has been partially supported by the European Commission under Horizon 2020 grant agreement no. 700199 "Securing against intruders and other threats through a NFV-enabled environment (SHIELD)". This support does not imply endorsement.

9. References

9.1. Normative References

[I-D.ietf-sfc-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., Aguado, A., and D. Lopez, "Proof of Transit", draft-ietf-sfc-proof-of-transit-01 (work in progress), October 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>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

[TCG] "Trusted Computing Group (TCG)", <https://www.trustedcomputinggroup.org/>.

[TCGGSS] "TCG Generic Server Specification, Version 1.0", <http://www.trustedcomputinggroup.org/>.

[TCGIRSS] "Infrastructure Work Group Integrity Report Schema Specification, Version 1.0", <https://www.trustedcomputinggroup.org/>.

9.2. Informative References

[UEFI] "UEFI Specification Version 2.2 (Errata D), Tech. Rep.".

[VMIA] Schiffman, J., Vijayakumar, H., and T. Jaeger, "Verifying System Integrity by Proxy", <http://dl.acm.org/citation.cfm?id=2368379>.

[VTPM] "vTPM:Virtualizing the Trusted Platform Module", <https://www.usenix.org/legacy/events/sec06/tech/berger.html>.

Authors' Addresses

Antonio Pastor
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain

Phone: +34 913 128 778
Email: antonio.pastorperales@telefonica.com

Diego R. Lopez
Telefonica I+D
Editor Jose Manuel Lara, 9 (1-B)
Seville, 41013
Spain

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Adrian L. Shaw
ARM

Email: als@arm.com

Network Working Group   A. Rein
Internet-Draft   L. Xia
Intended status: Informational   Huawei
Expires: March 8, 2019   September 04, 2018

The Remote Attestation NFV Use Cases
draft-rein-remote-attestation-nfv-use-cases-01

Abstract

This document proposes the use cases on an architectural level in terms of Remote Attestation for virtualized environments, especially in the context of Network Function Virtualization (NFV).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Stakeholders

Stakeholders play a major role in NFV and there is a strict hierarchical separation between the stakeholders in terms of responsibility, accessibility and visibility within the the NFV architecture. Although these issues are also relevant for other virtualized environments, for example in private or hybrid clouds, they are most apparent in NFV, especially in multi vendor deployments.

The stakeholders in NFV are:

- Cloud Service Provider (CSP): The CSP provides the platform, i.e. the hardware and core services, acting as the Virtual Machine Manager (VMM) or hypervisor for the provisioning of Virtual Machines (VM). With regard to this document, the CSP is not responsible for the provisioning itself. The CSP only provides the platform w.r.t. to CSP NFV Infrastructure (CSP:NFVI) role. The actual provisioning of specific VMs is carried out by the CSP Management and Orchestration (CSP:MANO) role, whereas both roles may be represented by the same or different organizations. This contribution, however, is not concerned with the internal operations and procedures of the CSP:MANO and therefore does address CSP:MANO neither as a role nor as a functional component.

- Cloud Service Customer (CSC): The CSC is the actual user of the VMM and requests the provisioning of specific VMs that eventually provide some service. The CSC is also in full control in terms of which specific VM is actually launched and thus not constrained in this regard.

- o Cloud Service User (CSU): The CSU is an external entity that uses the CSC's provided services. The CSU only has access to public API's provided by the offered service and has neither any responsibilities nor obligations within the NFV internals.

1.2. Major Issue

The most significant issue related to remote attestation is that the stakeholders may be constrained with regards to the information available to them. This means that in a strict model that involves a multi-vendor NFV deployment, access to certain information may only be available to one particular stakeholder. For instance, the CSP may only have direct access to the collected platform information and not to the information of provisioned VMs. Similarly, the CSC may be limited to the information collected on the provisioned VMs and does not have access to the information of the platform. This issue can be resolved by generally allowing the access to the information to all interested entities, w.r.t. a relaxed model, or allowing the access based on the enforcement of access permission policies.

More severe is the information necessary to carry out an appraisal of the measured information, that is the information about the expected configuration of the specific entity.

In principle there are two concerns, either this appraisal information should not be made available to a different entity (a stakeholder from a different organization) or the other entity does not want to carry out the appraisal by itself for any system not under its control. This means, even under the consideration that the collected information is shared between multiple stakeholders, the information necessary for carrying out the appraisal may not be available for different reasons.

To simplify the terminology, this contribution distinguishes between Remote Attestation Information Providers (RAIP) and Remote Attestation Information Consumers (RAIC). In particular:

- o CSP is limited to acting only as a RAIP for all authorized entities

- o CSC is limited to acting as a RAIP for all authorized entities and is a specific RAIC of CSP

- o RATP is limited to acting as a RAIP for all authorized entities and is a specific RAIC of CSP and CSC

Furthermore a new term, the Level of Assurance (LoA) is introduced. The LoA is defined as an hierarchical model that specifies the systems and components to be attested and whether an attestation is carried out on a local or remote basis.

With regards to this document, the overall targeted LoA equivalent to the NFV defined LoA levels 4 and 5 that corresponds to the remote attestation of the VMMs and VMs including the appraisal of load and runtime of applications within the VM’s scope.

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

3. Remote Attestation Use Cases

With regard to the main issue mentioned above, there are two options:

- o Decentralized Model: Each entity carries out the remote attestation only for the particular system under its control. This is referred to as the decentralized remote attestation model and involved multiple remote attestation servers.

- o Centralized Model: A new entity, referred to as a Remote Attestation Trusted Party (RATP), is introduced that has access to all information necessary to carry out a remote attestation on its own. This is referred to as the centralized remote attestation model.

However, the goal of the remote attestation is to convince an internal or external entity that a specific service (a networking function) has been provided by a trusted VM that was executed on a trusted VMM. For case (1.) this implies that the internal or external entity that want to know about the trustworthiness of a service must inherently trust the appraised results of both, the CSP and CSC. For case (2.) this means that the internal or external entity must only trust in the decision made by the RATP.

3.1. Decentralized Model Use Case

In this case there are multiple independent remote attestation servers controlled and maintained by the CSP or CSC stakeholders.

Assumptions:

- o It is assumed that the model satisfies LoA of 4 and 5

Pre-Conditions:

- o Relevant collected evidence (RA measurement information) is available. Access permissions policies are either defined or enforced, or information is available to all RAICs.

- o Role-specific RA appraisal information is available to the RAIC, but limited to the systems and components directly managed by the corresponding stakeholder.

Use case description:

A deployed NFV system consists of multiple RAIPs and RAICs that are under the control of stakeholders from different organizations. The RAIPs collect evidence on systems and offer this information, for the purpose of appraising them, to RAICs that are also under the control of stakeholders from the same organization.

For example, any RAIP under the control of stakeholder S1 will only share its collected evidence with RAICs that are also under the control of stakeholder S1.

In addition, the information necessary to carry out an appraisal of the collected evidences (i.e., the RA appraisal information) are limited; RAICs from stakeholder S1 only have access to appraisal information for RAIPs that are under the control of the same stakeholder, i.e. S1. For this reason, a RAIC can only appraise collected evidence from a RAIP operated by the same stakeholder.

For example, there is RAIP1 (i.e. a VMM) and RAIC1 both under the control of CSP. RAIC1 receives the collected evidence from RAIP1, appraises it and makes a statement based on the appraised evidence (i.e. AR1).

```
          VMM
        --------       --------     -----
CSP:    |RAIP 1|  ---> |RAIC 1| --> |AR1|
        --------       --------     -----
```

Similarly, there is RAIP2 (i.e. a VM instantiated on top of CSP's VMM) and RAIC2 both under the control of CSC. RAIC2 receives the collected evidence from RAIP2, appraises it and makes a statement based on the appraised evidence (i.e. AR2).

```
           VM
        --------       --------     -----
CSC:    |RAIP 2|  ---> |RAIC 2| --> |AR2|
        --------       --------     -----
```

Under the consideration of the requirements defined by LoA 4 and 5 a single RAIC does not have the capability to satisfy these requirements. More specifically this means that at least CSP's and CSC's RAICs must work together to be compliant to LoA 4 and 5 requirements. As a result, RAICs may share appraisal results with other RAICs or other external entities. This sharing may be constrained by access permission policies. For instance an external entity may request AR1 from RAIC1 and AR2 from RAIC2 and derive AR12 under the assumption it trusts the individual appraised results from either RAIC.

```
--------       --------    -----
|RAIP 1|  ---> |RAIC 1| -->|AR1|--|
--------       --------    -----  |     -----------------     ------
                                  |---> |external entity|---->|AR12|
--------       --------    -----  |     -----------------     ------
|RAIP 2|  ---> |RAIC 2| -->|AR2|--|
--------       --------    -----
```

However, the appraisal result generated from any other system but a RAIC (e.g. derived by an arbitrary external entity) is not trusted by any other entity (e.g. CSP, CSC or CSU). This mean that in this case AR12 only has any semantic meaning for the system (i.e. external entity) who derived it.

Alternatively, RAIC2 may use AR1 as an additional input to RAIP2's collected evidence input and derive AR12 accordingly. This is also under the consideration that RAIC2 implicitly trusts the statement made by RAIC1.

```
             VMM
          --------       --------     -----
CSP:      |RAIP 1|  ---> |RAIC 1| --> |AR1|
          --------       --------     -----
                                        |
                           ,-----------'
                           |
              VM           v
          --------       --------     ------
CSC:      |RAIP 2|  ---> |RAIC 2| --> |AR12|
          --------       --------     ------
```

In this case, AR12 is derived from CSC's RAIC and therefore other systems do trust this statement because it came from an authorized entity within the system.

See the summary table as below:

	CSP	CSC	CSU	External Entity
Provides RA measurement information	anyone authorized	anyone authorized	-	-
Has RA appraisal information	Only CSP	Only CSC	-	-
Provides RA appraisal results	anyone authorized	anyone authorized	-	-
Has access to RA Appraisal Results	From CSP and, if eligible, CSC	From CSC and, if eligible, CSP	From CSC or CSP (if eligible)	From CSC or CSP (if eligible)

3.2. Centralized Model (in a Single Trust Domain) Use Case

In this case there are multiple independent remote attestation servers controlled and maintained by the CSP or CSC stakeholders.

Assumptions:

- o It is assumed that the model satisfies LoA of 4 and 5

Pre-Conditions:

- o Relevant collected evidence (RA measurement information) is available to RATP without any restriction.
- o Role-specific RA appraisal information is available to RATP without any restriction.

Use case description:

A deployed NFV system consists of multiple RAIPs and RAICs that are under the control of stakeholders from different organizations and, in addition one RATP that is implicitly trusted by the other entities in the system. The RAIPs collect evidence on systems and offer this information, for the purpose of appraising them, to RAICs that are also under the control of stakeholders from the same organization or to RATP.

For example, any RAIP under the control of stakeholder S1 will only share its collected evidence with RAICs that are also under the control of stakeholder S1 and RATP.

In addition, the information necessary to carry out an appraisal of the collected evidences (i.e., the RA appraisal information) are limited; RAICs from stakeholder S1 only have access to appraisal information for RAIPs that are under the control of the same stakeholder, i.e. S1.

In contrast to this, RATP has access to all appraisal information of any system under evaluation from all stakeholders, i.e. CSP and CSC.

Similar to use-case 1, a RAIC can only appraise collected evidence from a RAIP operated by the same stakeholder, whereas RATP can appraise collected evidence from all stakeholders.

For example, there is RAIP1 (i.e. a VMM) under the control of CSP and RATP. RATP receives the collected evidence from RAIP1, appraises it and makes a statement based on the appraised evidence (i.e. AR1).

```
             VMM
           --------       ------     -----
CSP:       |RAIP 1|  ---> |RATP| --> |AR1|
           --------       ------     -----
```

Similarly, there is RAIP2 (i.e. a VM instantiated on top of CSP's VMM) under the control of CSC and RATP. RAIC2 receives the collected evidence from RAIP2, appraises it and makes a statement based on the appraised evidence (i.e. AR2).

```
              VM
           --------       ------     -----
CSC:       |RAIP 2|  ---> |RATP| --> |AR2|
           --------       ------     -----
```

Under the consideration of the requirements defined by LoA 4 and 5, RATP satisfies these requirements under the assumption that it received the correlated collected evidences from both VMM and VM. RATP may share its appraisal results with any other entity, however, access permissions policies may constrain access to the information. For instance, considering access permissions allow it, an external entity may request AR12 from RATP.

```
--------
|RAIP 1|  -,
--------   |   ------      ------    -----------------
            -> |RATP| ---> |AR12| -> |external entity|
--------   |   ------      ------    -----------------
|RAIP 2|  -'
--------
```

Important to note in this case is that the appraisal result provided by RATP is ultimately trusted by all participating systems from any stakeholder and all external entities the request this appraisal result.

See the summary table as below:

	CSP	CSC	CSU	RATP	External System
Provides RA measurement information	To RATP or CSP	To RATP or CSC	-	-	
Has RA appraisal information	Only CSP	Only CSC	-	CSP and CSC	
Provides RA appraisal results	-	-	-	For CSP, CSC, CSU, External system (access restricti ons may be defined)	
Has access to RA Appraisal results	From RATP (if eligible)	From RATP (if eligible)	From RATP (if eligible)	From RATP (if eligible)	From RATP (if eligible)

4. IANA Considerations

This document makes no request of IANA.

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

5. Security Considerations

To be added.

6. Acknowledgements

7. 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>.

Authors' Addresses

Andre Rein
Huawei

Email: Andre.Rein@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com