Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

S. Hares
Huawei
J. Jeong
J. Kim
Sungkyunkwan University
R. Moskowitz
HTT Consulting
Q. Lin
Huawei

October 30, 2017

I2NSF Capability YANG Data Model
draft-hares-i2nsf-capability-data-model-05

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.

- 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.

- 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. Objectives

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 objects are defined as user security event capabilities, device security event capabilities, system security event capabilities, and time security event capabilities. These objects 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 objects are defined as user security event, device security event, system security event, and time security event. These objects are 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 objects 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 objects are defined as ingress action capabilities, egress action capabilities, and apply profile action capabilities. These objects 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 time-zone
            |  +--rw start-time?   boolean
            |  +--rw end-time?     boolean
            +--rw rule-description?        boolean
            +--rw rule-rev?                boolean
            +--rw rule-priority?           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 time-zone
          |  +--rw start-time?   boolean
          |  +--rw end-time?     boolean
          +--rw rule-description?       boolean
          +--rw rule-rev?               boolean
          +--rw rule-priority?          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-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-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 time-zone
            |  +--rw start-time?   boolean
            |  +--rw end-time?     boolean
            +--rw rule-description?        boolean
            +--rw rule-rev?                boolean
            +--rw rule-priority?           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 time-zone
            |  +--rw start-time?   boolean
            |  +--rw end-time?     boolean
            +--rw rule-description?        boolean
            +--rw rule-rev?                boolean
            +--rw rule-priority?           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 time-zone
            |  +--rw start-time?   boolean
            |  +--rw end-time?     boolean
            +--rw rule-description?        boolean
            +--rw rule-rev?                boolean
            +--rw rule-priority?           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@2017-10-30.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 "2017-10-30"{
      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";
        }

        container time-zone {
          description
            "This can be used to apply rules according to time";
          leaf start-time {
            type boolean;
            description
              "This is start time for time zone";
          }
          leaf end-time {
            type boolean;
            description
              "This is end time for time zone";
```

```
          }
        }

        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 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-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 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 [i2nsf-framework].

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, "I2NSF Problem Statement and Use cases", RFC 8192, July 2017.

12.2. 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-08 (work in progress), October 2017.

[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-nsf-yang]
Kim, J., Jeong, J., Park, J., Hares, S., and L. Xia, "I2NSF Network Security Functions-Facing Interface YANG Data Model", draft-kim-i2nsf-nsf-facing-interface-data-model-03 (work in progress), October 2017.

[i2nsf-sfc]
Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-03 (work in progress), 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-04 (work in progress), July 2017.

[i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-08 (work in progress), July 2017.

[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. 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. draft-hares-i2nsf-capability-data-model-04

The following changes are made from draft-hares-i2nsf-capability-data-model-04:

1. Overview section is added to help explain of this Capability YANG data model.

2. Objectives section is added to specify objectives of this Capability YANG data model.

3. Capabilities of Event, Condition, Action, Resolution Strategy, and Default Action are added to express capabilities that NSFs can support.

4. RPC is added to acquire an appropriate network security function according to type of NSF and/or target devices.

5. This YANG data model is modified for vendors to be extended the YANG data model if they need specific capabilities for their devices.

6. An example is added for extended the YANG data model about specific NSF.

7. An examples are added about configuration XML for generic network security function and extended VoIP/VoLTE security function capabilities.

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

EMail: linqiushi@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

D. Hong
J. Jeong
J. Kim
Sungkyunkwan University
S. Hares
L. Xia
Huawei

October 30, 2017

YANG Data Model for Monitoring I2NSF Network Security Functions
draft-hong-i2nsf-nsf-monitoring-data-model-01

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 monitoring-message
    +--rw monitoring-messages* [message-id]
       +--rw message-id                    uint8
       +--rw message-version               uint8
       +--rw (message-type)?
       |  +--:(event)
       |  |  +--rw event-name                    string
       |  |  +--rw (event-type)?
       |  |     +--:(system-event)
       |  |     |  +--rw access-violation
       |  |     |  |  +--rw group                string
       |  |     |  |  +--rw login-ip             inet:ipv4-address
```

```
     |  |     |  |  +--rw authentication-mode
     |  |     |  |     +--rw local-authentication              boolean
     |  |     |  |     +--rw third-part-server-authentication  boolean
     |  |     |  |     +--rw exemption-authentication       boolean
     |  |     |  |     +--rw sso-authentication             boolean
     |  |     |  +--rw config-change
     |  |     |    +--rw group                  string
     |  |     |    +--rw login-ip               inet:ipv4-address
     |  |     |    +--rw authentication-mode
     |  |     |      +--rw local-authentication              boolean
     |  |     |      +--rw third-part-server-authentication  boolean
     |  |     |      +--rw exemption-authentication       boolean
     |  |     |      +--rw sso-authentication             boolean
     |  |     +--:(nsf-event)
     |  |        +--rw user-name?                     string
     |  |        +--rw ddos-event
     |  |        |  +--rw message?           string
     |  |        |  +--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 rule-id            uint8
     |  |        |  +--rw rule-name          string
     |  |        |  +--rw profile?           string
     |  |        |  +--rw raw-info?          string
     |  |        |  +--rw ddos-attack-type
     |  |        |  |  +--rw syn-flood?             boolean
     |  |        |  |  +--rw ack-flood?             boolean
     |  |        |  |  +--rw syn-ack-flood?         boolean
     |  |        |  |  +--rw fin-rst-flood?         boolean
     |  |        |  |  +--rw tcp-connection-flood?   boolean
     |  |        |  |  +--rw udp-flood?             boolean
     |  |        |  |  +--rw icmp-flood?            boolean
     |  |        |  |  +--rw https-flood?           boolean
     |  |        |  |  +--rw http-flood?            boolean
     |  |        |  |  +--rw dns-reply-flood?       boolean
     |  |        |  |  +--rw dns-query-flood?       boolean
     |  |        |  |  +--rw sip-flood?             boolean
     |  |        |  +--rw start-time         yang:date-and-time
     |  |        |  +--rw end-time           yang:date-and-time
     |  |        |  +--rw attack-rate?       uint32
     |  |        |  +--rw attack-speed?      uint32
     |  |        +--rw session-table-event
     |  |        |  +--rw current-session?   uint8
     |  |        |  +--rw maximum-session?   uint8
     |  |        |  +--rw threshold?         uint8
```

```
     |  |        |  +--rw message?           string
     |  |        +--rw virus-event
     |  |        |  +--rw message?      string
     |  |        |  +--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 rule-id       uint8
     |  |        |  +--rw rule-name     string
     |  |        |  +--rw profile?      string
     |  |        |  +--rw raw-info?     string
     |  |        |  +--rw virus-type
     |  |        |  |  +--rw trajan?   boolean
     |  |        |  |  +--rw worm?     boolean
     |  |        |  |  +--rw macro?    boolean
     |  |        |  +--rw virus-name?   string
     |  |        |  +--rw file-type?    string
     |  |        |  +--rw file-name?    string
     |  |        +--rw intrusion-event
     |  |        |  +--rw message?                 string
     |  |        |  +--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 rule-id                  uint8
     |  |        |  +--rw rule-name                string
     |  |        |  +--rw profile?                 string
     |  |        |  +--rw raw-info?                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 intrusion-attack-type
     |  |        |     +--rw brutal-force?      boolean
     |  |        |     +--rw buffer-overflow?   boolean
     |  |        +--rw botnet-event
     |  |        |  +--rw message?       string
     |  |        |  +--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 rule-id        uint8
     |  |        |  +--rw rule-name      string
     |  |        |  +--rw profile?       string
     |  |        |  +--rw raw-info?      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 botnet-name?   string
     |  |        |  +--rw role?          string
     |  |        +--rw web-attack-event
     |  |           +--rw message?           string
     |  |           +--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 rule-id            uint8
     |  |           +--rw rule-name          string
     |  |           +--rw profile?           string
     |  |           +--rw raw-info?          string
     |  |           +--rw web-attack-type
     |  |           |  +--rw sql-injection?       boolean
     |  |           |  +--rw command-injection?   boolean
     |  |           |  +--rw xss?                 boolean
     |  |           |  +--rw csrf?                boolean
     |  |           +--rw req-method
     |  |           |  +--rw put?   boolean
     |  |           |  +--rw get?   boolean
     |  |           +--rw req-url?           string
     |  |           +--rw url-category?      string
     |  |           +--rw filtering-type
     |  |              +--rw blacklist?            boolean
     |  |              +--rw whitelist?            boolean
     |  |              +--rw user-defined?         boolean
     |  |              +--rw balicious-category?   boolean
     |  |              +--rw unknown?              boolean
     |  +--:(log)
     |  |  +--rw (log-type)?
     |  |     +--:(system-log)
```

```
     |  |     |  +--rw access-logs
     |  |     |  |  +--rw login-ip          inet:ipv4-address
     |  |     |  |  +--rw administartor?    string
     |  |     |  |  +--rw login-mode?       login-mode
     |  |     |  |  +--rw operation-type?   operation-type
     |  |     |  |  +--rw result?           string
     |  |     |  |  +--rw content?          string
     |  |     |  +--rw resource-utiliz-logs
     |  |     |  |  +--rw system-status?       string
     |  |     |  |  +--rw cpu-usage?           uint8
     |  |     |  |  +--rw memory-usage?        uint8
     |  |     |  |  +--rw disk-usage?          uint8
     |  |     |  |  +--rw disk-left?           uint8
     |  |     |  |  +--rw session-num?         uint8
     |  |     |  |  +--rw process-num?         uint8
     |  |     |  |  +--rw in-traffic-rate?     uint32
     |  |     |  |  +--rw out-traffic-rate?    uint32
     |  |     |  |  +--rw in-traffic-speed?    uint32
     |  |     |  |  +--rw out-traffic-speed?   uint32
     |  |     |  +--rw user-activity-logs
     |  |     |     +--rw user                    string
     |  |     |     +--rw group                   string
     |  |     |     +--rw login-ip                inet:ipv4-address
     |  |     |     +--rw authentication-mode
     |  |     |     |  +--rw local-authentication              boolean
     |  |     |     |  +--rw third-part-server-authentication  boolean
     |  |     |     |  +--rw exemption-authentication       boolean
     |  |     |     |  +--rw sso-authentication             boolean
     |  |     |     +--rw access-mode
     |  |     |     |  +--rw ppp?     boolean
     |  |     |     |  +--rw svn?     boolean
     |  |     |     |  +--rw local?   boolean
     |  |     |     +--rw online-duration?       string
     |  |     |     +--rw logout-duration?       string
     |  |     |     +--rw addtional-info?        string
     |  |     +--:(nsf-log)
     |  |        +--rw ddos-logs
     |  |        |  +--rw attack-type?        string
     |  |        |  +--rw attack-ave-rate?    uint32
     |  |        |  +--rw attack-ave-speed?   uint32
     |  |        |  +--rw attack-pkt-num?     uint32
     |  |        |  +--rw attack-src-ip?      inet:ipv4-address
     |  |        |  +--rw action?             all-action
     |  |        |  +--rw os?                 string
     |  |        +--rw virus-logs
     |  |        |  +--rw protocol
     |  |        |  |  +--rw tcp?      boolean
     |  |        |  |  +--rw udp?      boolean
```

```
        |  |        |  |  +--rw icmp?     boolean
        |  |        |  |  +--rw icmpv6?   boolean
        |  |        |  |  +--rw ip?       boolean
        |  |        |  |  +--rw http?     boolean
        |  |        |  |  +--rw ftp?      boolean
        |  |        |  +--rw attack-type?   string
        |  |        |  +--rw action?        all-action
        |  |        |  +--rw os?            string
        |  |        |  +--rw time           yang:date-and-time
        |  |        +--rw intrusion-logs
        |  |        |  +--rw attack-type?    string
        |  |        |  +--rw action?         all-action
        |  |        |  +--rw time            yang:date-and-time
        |  |        |  +--rw attack-rate?    uint32
        |  |        |  +--rw attack-speed?   uint32
        |  |        +--rw botnet-logs
        |  |        |  +--rw attack-type?      string
        |  |        |  +--rw botnet-pkt-num?   uint8
        |  |        |  +--rw action?           all-action
        |  |        |  +--rw os?               string
        |  |        +--rw dpi-logs
        |  |        |  +--rw dpi-type?      dpi-type
        |  |        |  +--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 file-type?     string
        |  |        |  +--rw file-name?     string
        |  |        +--rw vulnerability-scanning-logs* [vulnerability-id]
        |  |        |  +--rw vulnerability-id    uint8
        |  |        |  +--rw victim-ip?          inet:ipv4-address
        |  |        |  +--rw protocol
        |  |        |  |  +--rw tcp?      boolean
```

```
     |  |        |  |  +--rw udp?      boolean
     |  |        |  |  +--rw icmp?     boolean
     |  |        |  |  +--rw icmpv6?   boolean
     |  |        |  |  +--rw ip?       boolean
     |  |        |  |  +--rw http?     boolean
     |  |        |  |  +--rw ftp?      boolean
     |  |        |  +--rw port-num?           inet:port-number
     |  |        |  +--rw level?              severity
     |  |        |  +--rw os?                 string
     |  |        |  +--rw addtional-info?     string
     |  |        +--rw web-attack-logs
     |  |           +--rw attack-type?     string
     |  |           +--rw rsp-code?        string
     |  |           +--rw req-clientapp?   string
     |  |           +--rw req-cookies?     string
     |  |           +--rw req-host?        string
     |  |           +--rw raw-info?        string
     |  +--:(counters)
     |     +--rw (counter-type)?
     |        +--:(system-counter)
     |        |  +--rw interface-counters
     |        |     +--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
     |        +--:(nsf-counter)
     |        |  +--rw firewall-counters
     |        |     +--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 bound
       |        |        +--rw in-interface?    boolean
       |        |        +--rw out-interface?   boolean
       |        +--:(policy-hit-counters)
       |           +--rw policy-hit-counters
       |              +--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
       +--rw message                        string
       +--rw time-stamp                     yang:date-and-time
       +--rw 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@2017-10-30.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 "2017-10-29" {
       description "Initial revision";
       reference
         "draft-zhang-i2nsf-info-model-monitoring-04";
     }

     typedef severity {
       type enumeration {
         enum high {
           description
             "high-level";
         }
         enum middle {
           description
             "middle-level";
         }
         enum low {
           description
             "low-level";
         }
       }
       description
         "This is used for indicating the severity";
     }
     typedef all-action {
       type enumeration {
         enum allow {
           description
             "TBD";
         }
         enum alert {
           description
             "TBD";
         }
         enum block {
           description
             "TBD";
         }
         enum discard {
           description
             "TBD";
         }
         enum declare {
           description
             "TBD";
```

```
          }
          enum block-ip {
            description
              "TBD";
          }
          enum block-service{
            description
              "TBD";
          }
        }
        description
          "This is used for protocol";
      }
      typedef dpi-type{
        type enumeration {
          enum file-blocking{
            description
              "TBD";
          }
          enum data-filtering{
            description
              "TBD";
          }
          enum application-behavior-control{
            description
              "TBD";
          }
        }
        description
          "This is used for dpi type";
      }
      typedef operation-type{
        type enumeration {
          enum login{
            description
              "TBD";
          }
          enum logout{
            description
              "TBD";
          }
          enum configuration{
            description
              "TBD";
          }
        }
        description
          "This is used for operation type";
```

```
     }
     typedef login-mode{
       type enumeration {
         enum root{
           description
             "TBD";
         }
         enum user{
           description
             "TBD";
         }
         enum guest{
           description
             "TBD";
         }
       }
       description
         "This is used for login mode";
     }
     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 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-event-type-content {
       description
         "A set of system event type contents";
       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.";
       }
       container authentication-mode {
         description
           "User authentication mode. e.g., Local Authentication,
           Third-Party Server Authentication,
           Authentication Exemption, SSO Authentication.";
         leaf local-authentication {
           type boolean;
           mandatory true;
           description
             "Authentication-mode : local authentication.";
         }
         leaf third-part-server-authentication {
           type boolean;
           mandatory true;
           description
             "TBD";
```

```
         }
         leaf exemption-authentication {
           type boolean;
           mandatory true;
           description
             "TBD";
         }
         leaf sso-authentication {
           type boolean;
           mandatory true;
           description
             "TBD";
         }
       }
     }
     grouping i2nsf-nsf-event-type-content {
       description
         "A set of nsf event type contents";
       leaf 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 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;
          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 protocol;
        uses traffic-rates;
      }

      container monitoring-message {
```

```
       description
         "The message for monitoring information";
       list monitoring-messages {
         key message-id;
         description
           "The messages according to monitoring information";
         leaf message-id {
           type uint8;
           mandatory true;
           description
             "This is message ID
              This is key for monitoring messages";
         }
         leaf message-version {
           type uint8;
           mandatory true;
           description
             "The version of message";
         }
         choice message-type {
           description
             "The type of message";
           case event {
             description
               "If the message type is event";
             leaf event-name {
               type string;
               mandatory true;
               description
                 "The name of the event";
             }
             choice event-type {
               description
                 "This is event type such as system event
                 and nsf event.";
               case system-event {
                 description
                   "If the event type is system event";
                 container access-violation {
                   description
                     "If the system event is
                     access violation";
                   uses i2nsf-system-event-type-content;
                 }
                 container config-change {
                   description
                     "If the system event is
                     config change violation";
```

```
              uses i2nsf-system-event-type-content;
            }
          }
          case nsf-event {
            description
              "If the event type is nsf event";
            leaf user-name {
              type string;
              description
                "This is user name for NSF event";
            }
            container ddos-event {
              description
                "If the event type is DDoS event";
              uses i2nsf-nsf-event-type-content;
              container ddos-attack-type{
                description
                  "Type of DDoS attack";
                leaf syn-flood{
                  type boolean;
                  description
                    "If the DDoS attack is
                    syn flood";
                }
                leaf ack-flood{
                  type boolean;
                  description
                    "If the DDoS attack is
                    ack flood";
                }
                leaf syn-ack-flood{
                  type boolean;
                  description
                    "If the DDoS attack is
                    syn ack flood";
                }
                leaf fin-rst-flood{
                  type boolean;
                  description
                    "If the DDoS attack is
                    fin rst flood";
                }
                leaf tcp-connection-flood{
                  type boolean;
                  description
                    "If the DDoS attack is
                    tcp connection flood";
                }
```

```
                     leaf udp-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         udp flood";
                     }
                     leaf icmp-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         icmp flood";
                     }
                     leaf https-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         https flood";
                     }
                     leaf http-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         http flood";
                     }
                     leaf dns-reply-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         dns reply flood";
                     }
                     leaf dns-query-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         dns query flood";
                     }
                     leaf sip-flood{
                       type boolean;
                       description
                         "If the DDoS attack is
                         sip flood";
                     }
                   }
                   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 of attack traffic";
               }
               leaf attack-speed {
                 type uint32;
                 description
                   "the bps of attack traffic";
               }
             }
             container session-table-event {
               description
                 "If the event type is session
                 table event";
                 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 message {
                   type string;
                   description
                    "The number of session table
                     exceeded the threshold";
```

```
                    }
                }
                container virus-event {
                  description
                    "If the event type is virus event";
                  uses i2nsf-nsf-event-type-content;
                  container virus-type {
                    description
                      "The type of virus";
                    leaf trajan {
                      type boolean;
                      description
                       "If the virus type is trajan";
                    }
                    leaf worm {
                      type boolean;
                      description
                        "If the virus type is worm";
                    }
                    leaf macro {
                      type boolean;
                      description
                        "If the virus type is macro";
                    }
                  }
                  leaf virus-name {
                    type string;
                    description
                      "The name of virus";
                  }
                  leaf file-type {
                    type string;
                    description
                      "The type of file";
                  }
                  leaf file-name {
                    type string;
                    description
                      "The name of file";
                  }
                }
                container intrusion-event {
                  description
                   "If the event type is intrusion event";
                  uses i2nsf-nsf-event-type-content;
                  uses protocol;
                  container intrusion-attack-type {
                    description
```

```
                  "The attack type of intrusion";
                leaf brutal-force {
                  type boolean;
                  description
                    "The intrusion type is
                    brutal force";
                }
                leaf buffer-overflow {
                  type boolean;
                  description
                    "The intrusion type is
                    buffer overflow";
                }
              }
            }
            container botnet-event {
              description
                "If the event type is botnet event";
              uses i2nsf-nsf-event-type-content;
              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";
              }
            }
            container web-attack-event {
              description
                "If the event type is web
                attack event";
              uses i2nsf-nsf-event-type-content;
              container web-attack-type {
                description
                  "To determine the attack
                  type";
                leaf sql-injection {
                  type boolean;
                  description
                    "If the web attack type is
                    sql injection";
                }
                leaf command-injection {
```

```
                  type boolean;
                  description
                    "If the web attack type is
                    command injection";
                }
                leaf xss {
                  type boolean;
                  description
                    "If the web attack type is
                    xss injection";
                }
                leaf csrf {
                  type boolean;
                  description
                    "If the web attack type is
                    csrf injection";
                }
              }
              container req-method {
                description
                 "The method of requirement.
                  For instance, PUT or GET
                  in HTTP";
                leaf put{
                  type boolean;
                  description
                    "If req method is PUT";
                }
                leaf get {
                 type boolean;
                  description
                    "If req method is GET";
                }
              }
              leaf req-url {
                type string;
                description
                  "Requested URL";
              }
              leaf url-category {
                type string;
                description
                  "Matched URL category";
              }
              container filtering-type {
                description
                 "URL filtering type,
                  e.g., Blacklist, Whitelist,
```

```
                      User-Defined, Predefined,
                      Malicious Category, Unknown";
                    leaf blacklist {
                      type boolean;
                      description
                        "The filtering type is
                        blacklist";
                    }
                    leaf whitelist {
                      type boolean;
                      description
                        "The filtering type is
                        whitelist";
                    }
                    leaf user-defined {
                      type boolean;
                      description
                        "The filtering type is
                        user defined";
                    }
                    leaf balicious-category{
                      type boolean;
                      description
                        "The filtering type is
                        balicious category";
                    }
                    leaf unknown {
                      type boolean;
                      description
                        "The filtering type is
                        unknown";
                    }
                  }
                }
              }
            }
          }
          case log {
            description
              "If the message type is log";
            choice log-type {
              description
                "The type of log";
              case system-log{
                description
                  "If the log type is system log";
                container access-logs {
                  description
```

```
                      "If the log is access logs
                      in system log";
                    leaf login-ip {
                      type inet:ipv4-address;
                      mandatory true;
                      description
                        "Login IP address of a user.";
                    }
                    leaf administartor {
                      type string;
                      description
                        "Administrator that
                        operates on the device";
                    }
                    leaf login-mode {
                      type login-mode;
                      description
                        "Specifies the
                        administrator logs 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
                        "Operation performed by
                        an administrator after login.";
                    }
                  }
                  container resource-utiliz-logs {
                    description
                      "If the log is resource utilize
                      logs in system log";
                    leaf system-status {
                      type string;
                      description
                        "TBD";
                    }
                    leaf cpu-usage {
```

```
                      type uint8;
                      description
                        "specifies the amount of
                        cpu usage";
                    }
                    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 outbound
                        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";
                   }
                 }
                 container user-activity-logs {
                   description
                     "If the log is user activity
                     logs in system log";
                   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.";
                   }
                   container authentication-mode {
                     description
                       "User authentication mode. e.g.,
                       Local Authentication,
                       Third-Party Server Authentication,
                       Authentication Exemption, SSO Authentication.";
                     leaf local-authentication {
                       type boolean;
                       mandatory true;
                       description
                         "Authentication-mode : local authentication.";
                     }
                     leaf third-part-server-authentication {
                       type boolean;
```

```
              mandatory true;
              description
                "TBD";
            }
            leaf exemption-authentication {
              type boolean;
              mandatory true;
              description
                "TBD";
            }
            leaf sso-authentication {
              type boolean;
              mandatory true;
              description
                "TBD";
            }
          }
          container access-mode {
            description
              "TBD";
            leaf ppp{
              type boolean;
              description
                "TBD";
            }
            leaf svn{
              type boolean;
              description
                "TBD";
            }
            leaf local{
              type boolean;
              description
                "TBD";
            }
          }
          leaf online-duration {
            type string;
            description
              "TBD";
          }
          leaf logout-duration {
            type string;
            description
              "TBD";
          }
          leaf addtional-info {
            type string;
```

```
                    description
                      "TBD";
                  }
                }
              }
              case nsf-log{
                description
                  "If the log type is nsf log";
                container ddos-logs {
                  description
                    "If the log is DDoS logs
                    in nsf log";
                  leaf attack-type{
                    type string;
                    description
                      "DDoS";
                  }
                  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
                      "TBD";
                  }
                  leaf action {
                    type all-action;
                    description
                      "TBD";
                  }
                  leaf os  {
                    type string;
                    description
```

```
                "simple os information";
            }
          }
          container virus-logs {
            description
              "If the log is virus logs
              in nsf log";
            uses protocol;
            leaf attack-type{
              type string;
              description
                "Virus";
            }
            leaf action{
              type all-action;
              description
                "TBD";
            }
            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";
            }
          }
          container intrusion-logs {
            description
              "If the log is intrusion logs
              in nsf log";
            leaf attack-type{
              type string;
              description
                "Intrusion";
            }
            leaf action{
              type all-action;
              description
                "TBD";
            }
            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";
          }
        }
        container botnet-logs {
          description
            "If the log is botnet logs
            in nsf log";
          leaf attack-type{
            type string;
            description
              "Botnet";
          }
          leaf botnet-pkt-num{
            type uint8;
            description
              "The number of the packets
              sent to or from the
              detected botnet";
          }
          leaf action{
            type all-action;
            description
              "TBD";
          }
          leaf os{
            type string;
            description
              "simple os information";
          }
        }
        container dpi-logs {
          description
            "If the log is dpi logs
            in nsf log";
          leaf dpi-type{
            type dpi-type;
```

```
                      description
                        "The type of 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 protocol;
          leaf file-type {
            type string;
            description
              "The type of file";
          }
          leaf file-name {
            type string;
            description
              "The name of file";
          }
        }
        list vulnerability-scanning-logs {
          key vulnerability-id;
          description
            "If the log is vulnerability
            scanning logs in 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";
              }
            }
            container web-attack-logs {
              description
                "If the log is web attack
                logs in nsf log";
              leaf attack-type{
                type string;
                description
                  "Web Attack";
              }
              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.";
                  }
                }
              }
            }
          }
          case counters {
            description
              "If the message type is counters";
            choice counter-type {
              description
                "The type of counter";
              case system-counter {
                container interface-counters {
                  description
                    "The system counter type is
                    interface counter";
                  uses i2nsf-system-counter-type-content;
                }
              }
              case nsf-counter{
                container firewall-counters {
                  description
                    "The nsf counter type is
                    firewall counter";
                  uses i2nsf-nsf-counters-type-content;
                  container bound{
                    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 policy-hit-counters {
                description
                  "The counters of policy hit";
                uses i2nsf-nsf-counters-type-content;
                leaf hit-times{
```

```
                    type uint32;
                    description
                      "The hit times for policy";
                  }
                }
              }
            }
          }
          leaf message {
            type string;
            mandatory true;
            description
              "This is a message for monitoring information";
          }
          leaf time-stamp {
            type yang:date-and-time;
            mandatory true;
            description
              "Indicate the time when the message is generated";
          }
          leaf severity {
            type severity;
            mandatory true;
            description
              "The severity of the alarm such as
              critical, high, middle, low.";
          }
        }
      }
    }
    <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]
Hares,, S., Lopez,, J., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 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-04 (work in progress), 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-04 (work in progress), July 2017.

[i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.

Appendix A. draft-hong-i2nsf-nsf-monitoring-data-model-01

The following changes are made from draft-hong-i2nsf-nsf-monitoring-data-model-00:

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

2. 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

Network Working Group   S. Hyun
Internet-Draft   J. Jeong
Intended status: Standards Track   Sungkyunkwan University
Expires: May 3, 2018   J. Park
                                                                   ETRI
                                                               S. Hares
                                                                 Huawei
                                                       October 30, 2017

Service Function Chaining-Enabled I2NSF Architecture
draft-hyun-i2nsf-nsf-triggered-steering-04

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.

Status of This Memo

Copyright Notice

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 [i2nsf-framework]. 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

- 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.

- 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.

- 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 [sfc-nsh].

- 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 [sfc-nsh].

- 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 [Network-Service-Header]. 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

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. Acknowledgements

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

[RFC7665] Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015.

[sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header (NSH)", draft-ietf-sfc-nsh-27 (work in progress), October 2017.

10.2. 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-08 (work in progress), October 2017.

[i2nsf-problem]
Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "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-04 (work in progress), July 2017.

[Network-Service-Header]
P. Quinn, Ed, Cisco Systems, Inc, and U. Elzur, Ed., "Network Service Header", May 2016.

[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 L. Xia, "I2NSF Network Security Functions-Facing Interface YANG Data Model", draft-kim-i2nsf-nsf-facing-interface-data-model-03 (work in progress), October 2017.

[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.

Appendix A. Changes from draft-hyun-i2nsf-nsf-triggered-steering-03

The following changes have been made from draft-hyun-i2nsf-nsf-triggered-steering-03:

- o Section 7 has been added to discuss implementation considerations of the SFC-enabled I2NSF architecture.
- o 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
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

S. Hyun
T. Roh
S. Wi
J. Jeong
Sungkyunkwan University
J. Park
ETRI

October 30, 2017

Registration Interface Data Model
draft-hyun-i2nsf-registration-interface-dm-02

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.

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 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], [i2nsf-framework], [nsf-triggered-steering], [supa-policy-data-model], and [supa-policy-info-model].

- 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]

- 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]

- Network Security Function Profile (NSF Profile): NSF Profile specifies the inspection capabilities of the associated NSF instance. Each NSF instance has its own NSF Profile to specify the type of security service it provides and its resource capacity etc. [nsf-triggered-steering]

- 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]

- 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-profile
    |  uses i2nsf-nsf-profile
    +--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-profile
        |  uses i2nsf-nsf-profile
      +--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 Profile which specifies required NSF capability information. The deinstanciation-request is used to remove an existing NSF with NSF Access Information.

4.4. NSF Profile

This section expands the i2nsf-nsf-profile in Figure 2 and Figure 3.

```
NSF Profile
  +--rw i2nsf-nsf-profile
    +--rw i2nsf-capability
    |  uses ietf-i2nsf-capability
    +--rw performance-capability
    |  uses i2nsf-nsf-performance-caps
```

Figure 4: High-Level YANG of I2NSF NSF Profile

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].

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.

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@2017-10-27.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:tkroh@imtl.skku.ac.kr>

          Editor: Sarang Wi
          <mailto:sarang@imtl.skku.ac.kr>

          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 "2017-10-27"{
           description "The second revision";
           reference
           "draft-hares-i2nsf-capability-data-model-03
           draft-hyun-i2nsf-registration-interface-im-03.txt";
         }

         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-profile {
         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 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-profile {
          description
          "nsf-profile";
          uses i2nsf-nsf-profile;
         }

         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-profile {
             description
             "nsf-profile";
             uses i2nsf-nsf-profile;
           }
          }

          case deinstanciation-request {
           description
           "deinstanciation-request";
           container nsf-access-info {
             description
             "nsf-access-info";
             uses i2nsf-nsf-access-info;
           }
         }
        }
       }
   }

   <CODE ENDS>
```

Figure 7: 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-profile>
        <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-profile>
         <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 8: 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 [i2nsf-framework] for the purpose of achieving secure communication among components in the proposed architecture.

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).

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-xibassnez-i2nsf-capability-02 (work in progress), July 2017.

[i2nsf-capability-dm]
Hares, S., Jeong, J., Kim, J., Moskowitz, R., and L. Xia, "I2NSF Capability YANG Data Model", draft-hares-i2nsf-capability-data-model-04 (work in progress), October 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-reg-inf-im]
Hyun, S., Jeong, J., Woo, S., Yeo, Y., and J. Park, "Registration Interface Information Model", draft-hyun-i2nsf-registration-interface-im-01 (work in progress), 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-04 (work in progress), July 2017.

[i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-08 (work in progress), July 2017.

[nsf-triggered-steering]
Hyun, S., Jeong, J., Park, J., and S. Hares, "NSF-Triggered Traffic Steering", draft-hyun-i2nsf-nsf-triggered-steering-in-i2nsf-03 (work in progress), July 2017.

[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-01

The following changes are made from draft-hyun-i2nsf-registration-interface-dm-00:

o The description of NSF Performance Capability is specified in more detail than the previous version.

o The description of YANG data module for Registration interface was updated.

The following changes are made from draft-hyun-i2nsf-registration-interface-dm-01:

o We simplified the performance capability by abstracting away the details of each type of resources.

o We replaced the existing dynamic life cycle management sub-model with the instance management sub-model.

o The description of YANG data module for Registration interface was updated.

o 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/

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.ac.kr
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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.ac.kr
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

S. Hyun
T. Roh
S. Wi
J. Jeong
Sungkyunkwan University
J. Park
ETRI

October 30, 2017

Registration Interface Information Model
draft-hyun-i2nsf-registration-interface-im-03

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 registration and NSF instantiation 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 [i2nsf-framework]. 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][i2nsf-framework] [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 another NSF’s inspection request: 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 client’s policy. However, if there is no available NSF instance to serve the former NSF’s request, 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 Client: In I2NSF framework, users decide which security service is necessary in the system. If there is no NSF instances to enforce the client’s security policy, then we should also create the required instances by requesting DMS 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 Design                           |
|                                                                     |
|       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+       +-+-+-+-+-+-+-+-+-+-+-+-+   |
|       |    Instance Management  |       |    Registration      |   |
|       |         Sub-Model       |       |     Sub-Model        |   |
|       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+       +-+-+-+-+-+-+-+-+-+-+-+-+   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: The Registration Interface Information Model Design

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 Profile  |            |  NSF Access   |
|               |            |  Information  |
+-+-+-+-+-+-+-+-+-+            +-+-+-+-+-+-+-+-+-+
```

Figure 2: Instance Management Sub-Model Overview

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 Profile |               | NSF Access    |
           |             |               | Information |
           +-+-+-+-+-+-+-+-+             +-+-+-+-+-+-+-+-+
```

Figure 3: Registration Mechanism Sub-Model Overview

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) [draft-ietf-nvo3-vxlan-gpe-04], 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 Profile (Capabilities of an NSF instance)

NSF Profile basically describes the inspection 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. In addition, NSF Profile 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 [i2nsf-framework] for the purpose of achieving secure communication between components in the proposed architecture.

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 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.

[draft-ietf-nvo3-vxlan-gpe-04]
Maino, Ed., F., Kreeger, Ed., L., and U. Elzur, Ed., "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 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., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-04 (work in progress), July 2017.

[nsf-triggered-steering]
Hyun, S., Jeong, J., Park, J., and S. Hares, "NSF-Triggered Traffic Steering", draft-hyun-i2nsf-nsf-triggered-steering-03 (work in progress), July 2017.

Appendix A. Changes from draft-hyun-i2nsf-registration-interface-im-02

The following changes are made from draft-hyun-i2nsf-registration-interface-im-02:

- We added the contents of role-based ACL to NSF profile.
- We simplified the performance capability by abstracting away the details of each type of resources.
- We replaced the existing dynamic life cycle management with the instance management.
- 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/

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

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

R. Marin-Lopez
G. Lopez-Millan
University of Murcia
F. Pereniguez-Garcia
University Defense Center

August 5, 2019

Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-ietf-i2nsf-sdn-ipsec-flow-protection-07

Abstract

This document describes how providing IPsec-based flow protection by means of a Software-Defined Network (SDN) controller (aka. Security Controller) and establishes the requirements to support this service. It considers two main well-known scenarios in IPsec: (i) gateway-to-gateway and (ii) host-to-host. The SDN-based service described in this document allows the distribution and monitoring of IPsec information from a Security Controller to one or several flow-based Network Security Function (NSF). The NSFs implement IPsec to protect data traffic between network resources.

The document focuses on the NSF Facing Interface by providing models for configuration and state data required to allow the Security Controller to configure the IPsec databases (SPD, SAD, PAD) and IKEv2 to establish Security Associations with a reduced intervention of the network administrator.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software. The SDN paradigm relocates the control of network resources to a dedicated network element, namely SDN Controller. The SDN controller (or Security Controller in the context of this document) manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications. The SDN application can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via this interface [RFC7149] [ITU-T.Y.3300] [ONF-SDN-Architecture] [ONF-OpenFlow].

Recently, several network scenarios are considering a centralized way of managing different security aspects. For example, Software-Defined WANs (SD-WAN), an SDN extension providing a software abstraction to create secure network overlays over traditional WAN and branch networks. SD-WAN is based on IPsec as underlying security protocol and aims to provide flexible, automated, fast deployment and on-demand security network services such as IPsec SA management from a centralized point.

Therefore, with the growth of SDN-based scenarios where network resources are deployed in an autonomous manner, a mechanism to manage IPsec SAs according to the SDN architecture becomes more relevant. Thus, the SDN-based service described in this document will autonomously deal with IPsec SAs management following the SDN paradigm.

IPsec architecture [RFC4301] defines clear separation between the processing to provide security services to IP packets and the key management procedures to establish the IPsec Security Associations. In this document, we define a service where the key management procedures can be carried by an external and centralized entity: the Security Controller.

First, this document exposes the requirements to support the protection of data flows using IPsec [RFC4301]. We have considered two general cases:

1) IKE case. The Network Security Function (NSF) implements the Internet Key Exchange (IKE) protocol and the IPsec databases: the Security Policy Database (SPD), the Security Association Database (SAD) and the Peer Authorization Database (PAD). The Security Controller is in charge of provisioning the NSF with the required information to IKE, the SPD and the PAD.

2) IKE-less case. The NSF only implements the IPsec databases (no IKE implementation). The Security Controller will provide the required parameters to create valid entries in the SPD and the SAD into the NSF. Therefore, the NSF will have only support for IPsec while automated key management functionality is moved to the Security Controller.

In both cases, an interface/protocol is required to carry out this provisioning in a secure manner between the Security Controller and the NSF. In particular, IKE case requires the provision of SPD and PAD entries, the IKE credential and information related with the IKE negotiation (e.g. IKE_SA_INIT). IKE-less case requires the management of SPD and SAD entries. Based on YANG models in [netconf-vpn] and [I-D.tran-ipsecme-yang], RFC 4301 [RFC4301] and RFC 7296 [RFC7296], this document defines the required interfaces with a YANG model for configuration and state data for IKE, PAD, SPD and SAD (see Appendix A, Appendix B and Appendix C). Examples of the usage of these models can found in Appendix D, Appendix E and Appendix F.

This document considers two typical scenarios to manage autonomously IPsec SAs: gateway-to-gateway and host-to-host [RFC6071]. In these cases, hosts, gateways or both may act as NSFs. Finally, it also discusses the situation where two NSFs are under the control of two different Security Controllers. The analysis of the host-to-gateway (roadwarrior) scenario is out of scope of this document.

Finally, this work pays attention to the challenge "Lack of Mechanism for Dynamic Key Distribution to NSFs" defined in [RFC8192] in the particular case of the establishment and management of IPsec SAs. In fact,this I-D could be considered as a proper use case for this particular challenge in [RFC8192].

2. Requirements Language

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 [RFC2119]. When these words appear in lower case, they have their natural language meaning.

3. Terminology

This document uses the terminology described in [RFC7149], [RFC4301], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow], [ITU-T.X.1252], [ITU-T.X.800] and [I-D.ietf-i2nsf-terminology]. In addition, the following terms are defined below:

- Software-Defined Networking. A set of techniques enabling 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].

- Flow/Data Flow. Set of network packets sharing a set of characteristics, for example IP dst/src values or QoS parameters.

- Security Controller. An entity that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. In the context of this document, it provides IPsec management information.

- Network Security Function (NSF). Software that provides a set of security-related services.

- Flow-based NSF. A NSF that inspects network flows according to a set of policies intended for enforcing security properties. The NSFs considered in this document fall into this classification.

- Flow-based Protection Policy. The set of rules defining the conditions under which a data flow MUST be protected with IPsec, and the rules that MUST be applied to the specific flow.

- Internet Key Exchange (IKE) v2. Protocol to establish IPsec Security Associations (SAs). It requires information about the required authentication method (i.e. raw RSA/ECDSA keys or X.509 certificates), Diffie-Hellman (DH) groups, IPsec SAs parameters and algorithms for IKE SA negotiation, etc.

- Security Policy Database (SPD). It includes information about IPsec policies direction (in, out), local and remote addresses (traffic selectors information), inbound and outboud IPsec SAs, etc.

- Security Associations Database (SAD). It includes information about IPsec SAs, such as SPI, destination addresses, authentication and encryption algorithms and keys to protect IP flows.

- Peer Authorization Database (PAD). It provides the link between the SPD and a security association management protocol. It is used when the NSF deploys IKE implementation (IKE case).

4. Objectives

- To describe the architecture for the SDN-based IPsec management, which implements a security service to allow the establishment and management of IPsec security associations from a central point, in order to protect specific data flows.

- To define the interfaces required to manage and monitor the IPsec Security Associations (SA) in the NSF from a Security Controller. YANG models are defined for configuration and state data for IPsec management.

5. SDN-based IPsec management description

As mentioned in Section 1, two cases are considered, depending on whether the NSF ships an IKEv2 implementation or not: IKE case and IKE-less case.

5.1. IKE case: IKE/IPsec in the NSF

In this case the NSF ships an IKEv2 implementation besides the IPsec support. The Security Controller is in charge of managing and applying IPsec connection information (determining which nodes need to start an IKE/IPsec session, deriving and delivering IKE Credentials such as a pre-shared key, certificates, etc.), and applying other IKE configuration parameters (e.g. cryptographic algorithms for establishing an IKE SA) to the NSF for the IKE negotiation.

With these entries, the IKEv2 implementation can operate to establish the IPsec SAs. The application (administrator) establishes the IPsec requirements and information about the end points information (through the Client Facing Interface, [RFC8192]), and the Security Controller translates these requirements into IKE, SPD and PAD entries that will be installed into the NSF (through the NSF Facing Interface). With that information, the NSF can just run IKEv2 to establish the required IPsec SA (when the data flow needs protection). Figure 1 shows the different layers and corresponding functionality.

```
               +-------------------------------------------+
               |IPsec Management/Orchestration Application | Client or
               |          I2NSF Client                     | App Gateway
               +-------------------------------------------+
                                        |    Client Facing Interface
               +-------------------------------------------+
      Vendor   |             Application Support           |
      Facing<->|-------------------------------------------| Security
      Interface| IKE Credential,PAD and SPD entries Distr. | Controller
               +-------------------------------------------+
                                        |       NSF Facing Interface
               +-------------------------------------------+
               |                 I2NSF Agent               |
               |-------------------------------------------| Network
               |   IKE     |       IPsec(SPD,PAD)          | Security
               |-------------------------------------------| Function
               |         Data Protection and Forwarding    |
               +-------------------------------------------+
```

Figure 1: IKE case: IKE/IPsec in the NSF

5.1.1. Interface Requirements for IKE case

SDN-based IPsec flow protection services provide dynamic and flexible management of IPsec SAs in flow-based NSFs. In order to support this capability in IKE case, the following interface requirements need to be met:

- o A YANG data model for IKEv2, SPD and PAD configuration data, and for IKE state data.

- o In scenarios where multiple Security Controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface [RFC7426] is required to exchange IPsec-related information. For example, if two gateways need to establish an IPsec SA and both are under the control of two different controllers, then both Security Controllers need to exchange information to properly configure their own NSFs. That is, the may need to agree on whether IKEv2 authentication will be based on raw public keys, pre-shared keys, etc. In case of using pre-shared keys they will have to agree in the PSK.

5.2. IKE-less case: IPsec (no IKEv2) in the NSF.

In this case, the NSF does not deploy IKEv2 and, therefore, the Security Controller has to perform the IKE security functions and management of IPsec SAs by populating and managing the SPD and the SAD.

```
              +-----------------------------------------+
              |   IPsec Management  Application         | Client or
              |             I2NSF Client                | App Gateway
              +-----------------------------------------+
                                  |   Client Facing Interface
              +-----------------------------------------+
        Vendor|             Application Support         |
     Facing<->|-----------------------------------------| Security
     Interface|     SPD, SAD and PAD Entries Distr.     | Controller
              +-----------------------------------------+
                                  |   NSF Facing Interface
              +-----------------------------------------+
              |              I2NSF Agent                | Network
              |-----------------------------------------| Security
              |            IPsec (SPD,SAD)              | Function (NSF)
              |-----------------------------------------|
              |     Data Protection and Forwarding      |
              +-----------------------------------------+
```

Figure 2: IKE-less case: IPsec (no IKE) in the NSF

As shown in Figure 2, applications for flow protection run on the top of the Security Controller. When an administrator enforces flow-based protection policies through the Client Facing Interface, the Security Controller translates these requirements into SPD and SAD entries, which are installed in the NSF. PAD entries are not required since there is no IKEv2 in the NSF.

5.2.1. Interface Requirements for IKE-less case

In order to support the IKE-less case, the following requirements need to be met:

- o A YANG data model for configuration data for SPD and SAD and for state data for SAD.

- o In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface [RFC7426] is required to exchange IPsec-related information. NOTE: A possible east-west protocol for this IKE-less case could be IKEv2. However, this needs to be explore since the IKEv2 peers would be the Security Controllers.

Specifically, the IKE-less case assumes that the SDN controller has to perform some security functions that IKEv2 typically does, namely (non-exhaustive):

- IV generation.
- Prevent counter resets for the same key.
- Generation of pseudo-random cryptographic keys for the IPsec SAs.
- Rekey of the IPsec SAs based on notifications from the NSF (i.e. expire).
- Generation of the IPsec SAs when required based on notifications (i.e. sadb-acquire) from the NSF.
- NAT Traversal discovery and management.

Additionally to these functions, another set of tasks must be performed by the Security Controller (non-exhaustive list):

- IPsec SA's SPI random generation.
- Cryptographic algorithm/s selection.
- Usage of extended sequence numbers.
- Establishment of proper traffic selectors.

5.3. IKE case vs IKE-less case

In principle, IKE case is easier to deploy than IKE-less case because current gateways typically have an IKEv2/IPsec implementation. Moreover hosts can install easily an IKE implementation. As downside, the NSF needs more resources to hold IKEv2. Moreover, the IKEv2 implementation needs to implement an internal interface so that the IKE configuration sent by the Security Controller can be enforced in runtime.

Alternatively, IKE-less case allows lighter NSFs (no IKEv2 implementation), which benefits the deployment in constrained NSFs. Moreover, IKEv2 does not need to be performed in gateway-to-gateway and host-to-host scenarios under the same Security Controller (see Section 7.1). On the contrary, the overload of creating and managing IPsec SAs is shifted to the Security Controller since IKEv2 is not in the NSF. As a consequence, this may result in a more complex implementation in the controller side in comparison with IKE case. For example, the Security Controller have to deal with the latency existing in the path between the Security Controller and the NSF in order to solve tasks such as, rekey or creation and installation of new IPsec SAs. However, this is not specific to our contribution but a general aspect in any SDN-based network. In summary, this overload may create some scalability and performance issues when the number of NSFs is high.

Nevertheless, literature around SDN-based network management using a centralized Security Controller is aware about scalability and performance issues and solutions have been already provided and discussed (e.g. hierarchical Security Controllers; having multiple replicated Security Controllers, dedicated high-speed management networks, etc). In the context of SDN-based IPsec management, one way to reduce the latency and alleviate some performance issues can be the installation of the IPsec policies and IPsec SAs at the same time (proactive mode, as described in Section 7.1) instead of waiting for notifications (e.g. a notification sadb-acquire when a new IPsec SA is required) to proceed with the IPsec SA installations (reactive mode). Another way to reduce the overhead and the potential scalability and performance issues in the Security Controller is to apply the IKE case described in this document, since the IPsec SAs are managed between NSFs without the involvement of the Security Controller at all, except by the initial IKE configuration provided by the Security Controller. Other solutions, such as Controller-IKE [I-D.carrel-ipsecme-controller-ike], have proposed that NSFs provide their DH public keys to the Security Controller, so that the Security Controller distributes all public keys to all peers. All peers can calculate a unique pairwise secret for each other peer and there is no inter-NSF messages. A rekey mechanism is further described in [I-D.carrel-ipsecme-controller-ike].

In terms of security, IKE case provides better security properties than IKE-less case, as we discuss in section Section 9. The main reason is that the NSFs are generating the session keys and not the Security Controller.

5.3.1. Rekeying process.

For IKE case, the rekeying process is carried out by IKEv2, following the information defined in the SPD and SAD. Therefore, connections will live unless something different is required by the administrator or the Security Controller detects something wrong.

Traditionally, during a rekey process of the IPSec SA using IKE, a bundle of inbound and outbound IPsec SAs is taken into account from the perspective of one of the NSFs. For example, if the inbound IPsec SA expires both the inbound and outbound IPsec SA are rekeyed at the same time in that NSF. However, when IKE is not used, we have followed a different approach to avoid any packet loss during rekey: the Security Controller installs first the new inbound SAs in both NSFs and then, the outbound IPsec SAs.

In other words, for the IKE-less case, the Security Controller needs to take care of the rekeying process. When the IPsec SA is going to expire (e.g. IPsec SA soft lifetime), it has to create a new IPsec SA and remove the old one. This rekeying process starts when the Security Controller receives a sadb-expire notification or it decides so, based on lifetime state data obtained from the NSF.

To explain the rekeying process between two IPsec NSFs A and B, let assume that SPIa1 identifies the inbound IPsec SA in A, and SPIb1 the inbound IPsec SA in B. The rekeying process will take the following steps:

1. The Security Controller chooses two random values as SPI for the new inbound IPsec SAs: for example, SPIa2 for A and SPIb2 for B. These numbers MUST NOT be in conflict with any IPsec SA in A or B. Then, the Security Controller creates an inbound IPsec SA with SPIa2 in A and another inbound IPsec SA in B with SPIb2. It can send this information simultaneously to A and B.

2. Once the Security Controller receives confirmation from A and B, the controller knows that the inbound IPsec A are correctly installed. Then it proceeds to send in parallel to A and B, the outbound IPsec SAs: it sends the outbound IPsec SA to A with SPIb2 and the outbound IPsec SA to B with SPIa2. At this point the new IPsec SAs are ready.

3. Once the Security Controller receives confirmation from A and B that the outbound IPsec SAs have been installed, the Security Controller, in parallel, deletes the old IPsec SAs from A (inbound SPIa1 and outbound SPIb1) and B (outbound SPIa1 and inbound SPIb1).

If some of the operations in step 1 fail (e.g. the NSF A reports an error when the Security Controller is trying to install a new inbound IPsec SA) the Security Controller must perform rollback operations by removing any new inbound SA that had been successfully installed during step 1.

If step 1 is successful but some of the operations in step 2 fails (e.g. the NSF A reports an error when the Security Controller is trying to install the new outbound IPsec SA), the Security Controller must perform a rollback operation by deleting any new outbound SA that had been successfully installed during step 2 and by deleting the inbound SAs created in step 1.

If the steps 1 an 2 are successful and the step 3 fails the Security Controller will avoid any rollback of the operations carried out in step 1 and step 2 since new and valid IPsec SAs were created and are functional. The Security Controller may reattempt to remove the old inbound and outbound SAs in NSF A and NSF B several times until it receives a success or it gives up. In the last case, the old IPsec SAs will be removed when the hard lifetime is reached.

5.3.2. NSF state loss.

If one of the NSF restarts, it will lose the IPsec state (affected NSF). By default, the Security Controller can assume that all the state has been lost and therefore it will have to send IKEv2, SPD and PAD information to the NSF in the IKE case, and SPD and SAD information in IKE-less case.

In both cases, the Security Controller is aware of the affected NSF (e.g. the NETCONF/TCP connection is broken with the affected NSF, the Security Controller is receiving sadb-bad-spi notification from a particular NSF, etc.). Moreover, the Security Controller has a register about all the NSFs that have IPsec SAs with the affected NSF. Therefore, it knows the affected IPsec SAs.

In IKE case, the Security Controller will configure the affected NSF with the new IKEv2, SPD and PAD information. It has also to send new parameters (e.g. a new fresh PSK for authentication) to the NSFs which have IKEv2 SAs and IPsec SAs with the affected NSF. Finally, the Security Controller will instruct the affected NSF to start the IKEv2 negotiation with the new configuration.

In IKE-less case, if the Security Controller detects that a NSF has lost the IPsec SAs it will delete the old IPsec SAs on the non-failed nodes, established with the failed node (step 1). This prevents the non-failed nodes from leaking plaintext. If the affected node comes to live, the Security Controller will configure the new inbound IPsec SAs between the affected node and all the nodes it was talking to (step 2). After these inbound IPsec SAs have been established, the Security Controller can configure the outbound IPsec SAs in parallel (step 3).

Nevertheless other more optimized options can be considered (e.g. making the IKEv2 configuration permanent between reboots).

5.3.3. NAT Traversal

In the IKE case, IKEv2 already provides a mechanism to detect whether some of the peers or both are located behind a NAT. If there is a NAT network configured between two peers, it is required to activate the usage of UDP or TCP/TLS encapsulation for ESP packets ([RFC3948], [RFC8229]). Note that the usage of IPsec transport mode when NAT is required MUST NOT be used in this specification.

On the contrary, the IKE-less case does not have any protocol in the NSFs to detect whether they are located behind a NAT or not. However, the SDN paradigm generally assumes the Security Controller has a view of the network under its control. This view is built either requesting information to the NSFs under its control, or because these NSFs inform the Security Controller. Based on this information, the Security Controller can guess if there is a NAT configured between two hosts, and apply the required policies to both NSFs besides activating the usage of UDP or TCP/TLS encapsulation of ESP packets ([RFC3948], [RFC8229]).

For example, the Security Controller could directly request the NSF for specific data such as networking configuration, NAT support, etc. Protocols such as NETCONF or SNMP can be used here. For example, RFC 7317 [RFC7317] provides a YANG data model for system management or [RFC8512] a data model for NAT management. The Security Controller can use this NETCONF module with a NSF to collect NAT information or even configure a NAT network. In any case, if this NETCONF module is not available in the NSF and the Security Controller does not have a mechanism to know whether a host is behind a NAT or not, then the IKE case should be the right choice and not the IKE-less case.

5.3.4. NSF Discovery

The assumption in this document is that, for both cases, before a NSF can operate in this system, it MUST be registered in the Security Controller. In this way, when the NSF comes to live and establishes a connection to the Security Controller, it knows that the NSF is valid for joining the system.

Either during this registration process or when the NSF connects with the Security Controller, the Security Controller MUST discover certain capabilities of this NSF, such as what is the cryptographic suite supported, authentication method, the support of the IKE case or the IKE-less case, etc. This discovery process is out of the scope of this document.

6. YANG configuration data models

In order to support the IKE and IKE-less cases we have modeled the different parameters and values that must be configured to manage IPsec SAs. Specifically, IKE requires modeling IKEv2, SPD and PAD, while IKE-less case requires configuration models for the SPD and SAD. We have defined three models: ietf-ipsec-common (Appendix A), ietf-ipsec-ike (Appendix B, IKE case), ietf-ipsec-ikeless (Appendix C, IKE-less case). Since the model ietf-ipsec-common has only typedef and groupings common to the other modules, we only show a simplified view of the ietf-ipsec-ike and ietf-ipsec-ikeless models.

6.1. IKE case model

The model related to IKEv2 has been extracted from reading IKEv2 standard in [RFC7296], and observing some open source implementations, such as Strongswan [strongswan] or Libreswan [libreswan].

The definition of the PAD model has been extracted from the specification in section 4.4.3 in [RFC4301] (NOTE: We have observed that many implementations integrate PAD configuration as part of the IKEv2 configuration).

```
module: ietf-ipsec-ike
+--rw ipsec-ike
  +--rw pad
  |  +--rw pad-entry* [name]
  |     +--rw name                          string
  |     +--rw (identity)
  |     |  +--:(ipv4-address)
  |     |  |  +--rw ipv4-address?           inet:ipv4-address
  |     |  +--:(ipv6-address)
  |     |  |  +--rw ipv6-address?           inet:ipv6-address
  |     |  +--:(fqdn-string)
  |     |  |  +--rw fqdn-string?            inet:domain-name
  |     |  +--:(rfc822-address-string)
  |     |  |  +--rw rfc822-address-string?   string
  |     |  +--:(dnx509)
  |     |  |  +--rw dnx509?                 string
  |     |  +--:(gnx509)
  |     |  |  +--rw gnx509?                 string
  |     |  +--:(id-key)
  |     |  |  +--rw id-key?                 string
  |     |  +--:(id-null)
  |     |     +--rw id-null?                empty
  |     +--rw auth-protocol?                auth-protocol-type
  |     +--rw peer-authentication
  |        +--rw auth-method?               auth-method-type
  |        +--rw eap-method
  |        |  +--rw eap-type                uint8
  |        +--rw pre-shared
```

```
   |         |  +--rw secret?                  yang:hex-string
   |         +--rw digital-signature
   |            +--rw ds-algorithm?           uint8
   |            +--rw (public-key)
   |            |  +--:(raw-public-key)
   |            |  |  +--rw raw-public-key?   binary
   |            |  +--:(cert-data)
   |            |     +--rw cert-data?        ct:x509
   |            +--rw private-key?           binary
   |            +--rw ca-data*               ct:x509
   |            +--rw crl-data?              ct:crl
   |            +--rw crl-uri?               inet:uri
   |            +--rw oscp-uri?              inet:uri
   +--rw conn-entry* [name]
   |  +--rw name                                    string
   |  +--rw autostartup?                            autostartup-type
   |  +--rw initial-contact?                        boolean
   |  +--rw version?                                auth-protocol-type
   |  +--rw fragmentation?                          boolean
   |  +--rw ike-sa-lifetime-soft
   |  |  +--rw rekey-time?    uint32
   |  |  +--rw reauth-time?   uint32
   |  +--rw ike-sa-lifetime-hard
   |  |  +--rw over-time?   uint32
   |  +--rw authalg*  ic:integrity-algorithm-type
   |  +--rw encalg*   ic:encryption-algorithm-type
   |  +--rw dh-group?                               pfs-group
   |  +--rw half-open-ike-sa-timer?                 uint32
   |  +--rw half-open-ike-sa-cookie-threshold?   uint32
   |  +--rw local
   |  |  +--rw local-pad-entry-name?   string
   |  +--rw remote
   |  |  +--rw remote-pad-entry-name?   string
   |  +--rw encapsulation-type
   |  |  +--rw espencap?   esp-encap
   |  |  +--rw sport?      inet:port-number
   |  |  +--rw dport?      inet:port-number
   |  |  +--rw oaddr*      inet:ip-address
   |  +--rw spd
   |  |  +--rw spd-entry* [name]
   |  |     +--rw name                   string
   |  |     +--rw ipsec-policy-config
   |  |        +--rw anti-replay-window?   uint64
   |  |        +--rw traffic-selector
   |  |        |  +--rw local-subnet      inet:ip-prefix
   |  |        |  +--rw remote-subnet     inet:ip-prefix
   |  |        |  +--rw inner-protocol?   ipsec-inner-protocol
   |  |        |  +--rw local-ports* [start end]
```

```
     |  |      |  |  +--rw start           inet:port-number
     |  |      |  |  +--rw end             inet:port-number
     |  |      |  +--rw remote-ports* [start end]
     |  |      |     +--rw start           inet:port-number
     |  |      |     +--rw end             inet:port-number
     |  |      +--rw processing-info
     |  |      |  +--rw action?         ipsec-spd-action
     |  |      |  +--rw ipsec-sa-cfg
     |  |      |    +--rw pfp-flag?              boolean
     |  |      |    +--rw ext-seq-num?           boolean
     |  |      |    +--rw seq-overflow?          boolean
     |  |      |    +--rw stateful-frag-check?   boolean
     |  |      |    +--rw mode?                  ipsec-mode
     |  |      |    +--rw protocol-parameters? ipsec-protocol-parameters
     |  |      |    +--rw esp-algorithms
     |  |      |    |  +--rw integrity*  integrity-algorithm-type
     |  |      |    |  +--rw encryption* encryption-algorithm-type
     |  |      |    |  +--rw tfc-pad?      boolean
     |  |      |    +--rw tunnel
     |  |      |       +--rw local           inet:ip-address
     |  |      |       +--rw remote          inet:ip-address
     |  |      |       +--rw df-bit?         enumeration
     |  |      |       +--rw bypass-dscp?    boolean
     |  |      |       +--rw dscp-mapping?   yang:hex-string
     |  |      |       +--rw ecn?            boolean
     |  |      +--rw spd-mark
     |  |         +--rw mark?   uint32
     |  |         +--rw mask?   yang:hex-string
     |  +--rw child-sa-info
     |  |  +--rw pfs-groups*               pfs-group
     |  |  +--rw child-sa-lifetime-soft
     |  |  |  +--rw time?      uint32
     |  |  |  +--rw bytes?     uint32
     |  |  |  +--rw packets?   uint32
     |  |  |  +--rw idle?      uint32
     |  |  |  +--rw action?    ic:lifetime-action
     |  |  +--rw child-sa-lifetime-hard
     |  |     +--rw time?      uint32
     |  |     +--rw bytes?     uint32
     |  |     +--rw packets?   uint32
     |  |     +--rw idle?      uint32
     |  +--ro state
     |     +--ro initiator?             boolean
     |     +--ro initiator-ikesa-spi?   ike-spi
     |     +--ro responder-ikesa-spi?   ike-spi
     |     +--ro nat-local?             boolean
     |     +--ro nat-remote?            boolean
     |     +--ro encapsulation-type
```

```
     |     |  +--ro espencap?   esp-encap
     |     |  +--ro sport?      inet:port-number
     |     |  +--ro dport?      inet:port-number
     |     |  +--ro oaddr*      inet:ip-address
     |     +--ro established?          uint64
     |     +--ro current-rekey-time?    uint64
     |     +--ro current-reauth-time?   uint64
     +--ro number-ike-sas
        +--ro total?               uint64
        +--ro half-open?           uint64
        +--ro half-open-cookies?   uint64
```

Appendix D shows an example of IKE case configuration for a NSF, in tunnel mode (gateway-to-gateway), with NSFs authentication based on X.509 certificates.

6.2. IKE-less case model

For this case, the definition of the SPD model has been mainly extracted from the specification in section 4.4.1 and Appendix D in [RFC4301], though with some simplications. For example, each IPsec policy (spd-entry) contains one traffic selector, instead a list of them. The reason is that we have observed real kernel implementations only admit a traffic selector per IPsec policy.

The definition of the SAD model has been extracted from the specification in section 4.4.2 in [RFC4301]. Note that this model not only allows to associate an IPsec SA with its corresponding policy through the specific traffic selector but also an identifier (reqid).

The notifications model has been defined using as reference the PF_KEYv2 standard in [RFC2367].

```
module: ietf-ipsec-ikeless
+--rw ipsec-ikeless
 +--rw spd
 |  +--rw spd-entry* [name]
 |     +--rw name                   string
 |     +--rw direction?             ic:ipsec-traffic-direction
 |     +--rw reqid?                 uint64
 |     +--rw ipsec-policy-config
 |        +--rw anti-replay-window?   uint64
 |        +--rw traffic-selector
 |        |  +--rw local-subnet      inet:ip-prefix
 |        |  +--rw remote-subnet     inet:ip-prefix
```

```
   |         |  +--rw inner-protocol?   ipsec-inner-protocol
   |         |  +--rw local-ports* [start end]
   |         |  |  +--rw start          inet:port-number
   |         |  |  +--rw end            inet:port-number
   |         |  +--rw remote-ports* [start end]
   |         |     +--rw start          inet:port-number
   |         |     +--rw end            inet:port-number
   |         +--rw processing-info
   |         |  +--rw action?         ipsec-spd-action
   |         |  +--rw ipsec-sa-cfg
   |         |     +--rw pfp-flag?              boolean
   |         |     +--rw ext-seq-num?           boolean
   |         |     +--rw seq-overflow?          boolean
   |         |     +--rw stateful-frag-check?   boolean
   |         |     +--rw mode?                  ipsec-mode
   |         |     +--rw protocol-parameters?
   |         |     +--rw esp-algorithms
   |         |     |  +--rw integrity*    integrity-algorithm-type
   |         |     |  +--rw encryption*  encryption-algorithm-type
   |         |     |  +--rw tfc-pad?      boolean
   |         |     +--rw tunnel
   |         |        +--rw local           inet:ip-address
   |         |        +--rw remote          inet:ip-address
   |         |        +--rw df-bit?         enumeration
   |         |        +--rw bypass-dscp?    boolean
   |         |        +--rw dscp-mapping?   yang:hex-string
   |         |        +--rw ecn?            boolean
   |         +--rw spd-mark
   |            +--rw mark?   uint32
   |            +--rw mask?   yang:hex-string
   +--rw sad
    +--rw sad-entry* [name]
      +--rw name               string
      +--rw reqid?             uint64
      +--rw ipsec-sa-config
      |  +--rw spi                      uint32
      |  +--rw ext-seq-num?             boolean
      |  +--rw seq-number-counter?      uint64
      |  +--rw seq-overflow?            boolean
      |  +--rw anti-replay-window?      uint32
      |  +--rw traffic-selector
      |  |  +--rw local-subnet       inet:ip-prefix
      |  |  +--rw remote-subnet      inet:ip-prefix
      |  |  +--rw inner-protocol?    ipsec-inner-protocol
      |  |  +--rw local-ports*       [start end]
      |  |  |  +--rw start           inet:port-number
      |  |  |  +--rw end             inet:port-number
      |  |  +--rw remote-ports* [start end]
```

```
       |  |     +--rw start            inet:port-number
       |  |     +--rw end              inet:port-number
       |  +--rw protocol-parameters?  ic:ipsec-protocol-parameters
       |  +--rw mode?                  ic:ipsec-mode
       |  +--rw esp-sa
       |  |  +--rw encryption
       |  |  |  +--rw encryption-algorithm? ic:encryption-algorithm-type
       |  |  |  +--rw key?                    yang:hex-string
       |  |  |  +--rw iv?                     yang:hex-string
       |  |  +--rw integrity
       |  |     +--rw integrity-algorithm? ic:integrity-algorithm-type
       |  |     +--rw key?                   yang:hex-string
       |  +--rw sa-lifetime-hard
       |  |  +--rw time?      uint32
       |  |  +--rw bytes?     uint32
       |  |  +--rw packets?   uint32
       |  |  +--rw idle?      uint32
       |  +--rw sa-lifetime-soft
       |  |  +--rw time?      uint32
       |  |  +--rw bytes?     uint32
       |  |  +--rw packets?   uint32
       |  |  +--rw idle?      uint32
       |  |  +--rw action?    ic:lifetime-action
       |  +--rw tunnel
       |  |  +--rw local           inet:ip-address
       |  |  +--rw remote          inet:ip-address
       |  |  +--rw df-bit?         enumeration
       |  |  +--rw bypass-dscp?    boolean
       |  |  +--rw dscp-mapping?   yang:hex-string
       |  |  +--rw ecn?            boolean
       |  +--rw encapsulation-type
       |     +--rw espencap?   esp-encap
       |     +--rw sport?      inet:port-number
       |     +--rw dport?      inet:port-number
       |     +--rw oaddr*      inet:ip-address
       +--ro ipsec-sa-state
          +--ro sa-lifetime-current
          |  +--ro time?      uint32
          |  +--ro bytes?     uint32
          |  +--ro packets?   uint32
          |  +--ro idle?      uint32
          +--ro replay-stats
             +--ro replay-window?        uint64
             +--ro packet-dropped?       uint64
             +--ro failed?               uint32
             +--ro seq-number-counter?   uint64

    notifications:
```

```
     +---n sadb-acquire
     |  +--ro ipsec-policy-name          string
     |  +--ro traffic-selector
     |     +--ro local-subnet            inet:ip-prefix
     |     +--ro remote-subnet           inet:ip-prefix
     |     +--ro inner-protocol?         ipsec-inner-protocol
     |     +--ro local-ports* [start end]
     |     |  +--ro start                inet:port-number
     |     |  +--ro end                  inet:port-number
     |     +--ro remote-ports* [start end]
     |        +--ro start                inet:port-number
     |        +--ro end                  inet:port-number
     +---n sadb-expire
     |  +--ro ipsec-sa-name           string
     |  +--ro soft-lifetime-expire?   boolean
     |  +--ro lifetime-current
     |     +--ro time?                uint32
     |     +--ro bytes?               uint32
     |     +--ro packets?             uint32
     |     +--ro idle?                uint32
     +---n sadb-seq-overflow
     |  +--ro ipsec-sa-name           string
     +---n sadb-bad-spi
        +--ro spi                     uint32
```

Appendix E shows an example of IKE-less case configuration for a NSF, in transport mode (host-to-host), with NSFs authentication based on shared secrets. For the IKE-less case, Appendix F shows examples of IPsec SA expire, acquire, sequence number overflow and bad SPI notifications.

7. Use cases examples

This section explains how different traditional configurations, that is, host-to-host and gateway-to-gateway, are deployed using this SDN-based IPsec management service. In turn, these configurations will be typical in modern networks where, for example, virtualization will be key.

7.1. Host-to-host or gateway-to-gateway under the same Security Controller

```
                      +----------------------------------------+
                      |           Security Controller          |
                      |                                        |
                   (1)|   +--------------+ (2)+--------------+ |
      Flow-based  ------> |Translate into|--->| South. Prot. | |
      Security. Pol.  |   |IPsec Policies|    |              | |
                      |   +--------------+    +--------------+ |
                      |                          |     |       |
                      |                          |     |       |
                      +--------------------------|-----|-------+
                                                 |     |
                                                 | (3) |
                       |-------------------------+     +---|
                       V                                   V
           +----------------------+         +----------------------+
           |    NSF A             |<=======>|   NSF B              |
           |IKEv2/IPsec(SPD/PAD)  |         |IKEv2/IPsec(SPD/PAD)  |
           +----------------------+  (4)    +----------------------+
```

Figure 3: Host-to-host / gateway-to-gateway single Security Controller for the IKE case.

Figure 3 describes the IKE case:

1. The administrator defines general flow-based security policies. The Security Controller looks for the NSFs involved (NSF A and NSF B).

2. The Security Controller generates IKEv2 credentials for them and translates the policies into SPD and PAD entries.

3. The Security Controller inserts an IKEv2 configuration that includes the SPD and PAD entries in both NSF A and NSF B. If some of operations with NSF A and NSF B fail the Security Controller will stop the process and perform a rollback operation by deleting any IKEv2, SPD and PAD configuration that had been successfully installed in NSF A or B.

4. If the previous step is successful, the flow is protected by means of the IPsec SA established with IKEv2.

```
                        +----------------------------------------+
                        |    (1)     Security Controller         |
            Flow-based  |                                        |
            Security -----------|                                |
            Policy      |       V                                |
                        |  +---------------+ (2)+-------------+  |
                        |  |Translate into |--->| South. Prot.|  |
                        |  |IPsec policies |    |             |  |
                        |  +---------------+    +-------------+  |
                        |                         |     |        |
                        |                         |     |        |
                        +-------------------------| --- |--------+
                                                  |     |
                                                  | (3) |
                           |----------------------+     +--|
                           V                               V
                  +------------------+       +------------------+
                  |    NSF A         |<=====>|   NSF B          |
                  |IPsec(SPD/SAD)    |   4)  |IPsec(SPD/SAD)    |
                  +------------------+       +------------------+
```

Figure 4: Host-to-host / gateway-to-gateway single Security Controller for IKE-less case.

In the IKE-less case, flow-based security policies defined by the administrator are translated into IPsec SPD entries and inserted into the corresponding NSFs. Besides, fresh SAD entries will be also generated by the Security Controller and enforced in the NSFs. In this case, the Security Controller does not run any IKEv2 implementation (neither the NSFs), and it provides the cryptographic material for the IPsec SAs. These keys will be also distributed securely through the southbound interface. Note that this is possible because both NSFs are managed by the same Security Controller.

Figure 4 describes the IKE-less case, when a data packet needs to be protected in the path between the NSF A and NSF B:

1. The administrator establishes the flow-based security policies, and the Security Controller looks for the involved NSFs.

2. The Security Controller translates the flow-based security policies into IPsec SPD and SAD entries.

3. The Security Controller inserts these entries in both NSF A and NSF B IPsec databases (SPD and SAD). The following text describes how this happens between two NSFs A and B:

* The Security Controller chooses two random values as SPIs: for example, SPIa1 for NSF A and SPIb1 for NSF B. These numbers MUST NOT be in conflict with any IPsec SA in NSF A or NSF B. It also generates fresh cryptographic material for the new inbound/outbound IPsec SAs and their parameters and send simultaneously the new inbound IPsec SA with SPIa1 and new outbound IPsec SAs with SPIb1 to NSF A; and the new inbound IPsec SA with SPIb1 and new outbound IPsec SAs with SPIa1 to B, together with the corresponding IPsec policies.

* Once the Security Controller receives confirmation from NSF A and NSF B, the controller knows that the IPsec SAs are correctly installed and ready.

If some of the operations described above fails (e.g. the NSF A reports an error when the Security Controller is trying to install the SPD entry, the new inbound and outbound IPsec SAs) the Security Controller must perform rollback operations by deleting any new inbound or outbound SA and SPD entry that had been successfully installed in any of the NSFs (e.g NSF B) and stop the process (NOTE: the Security Controller may retry several times before giving up). Other alternative to this operation is: the Security Controller sends first the IPsec policies and new inbound IPsec SAs to A and B and once it obtains a successful confirmation of these operations from NSF A and NSF B, it proceeds with installing to the new outbound IPsec SAs. However, this may increase the latency to complete the process. As an advantage, no traffic is sent over the network until the IPsec SAs are completely operative. In any case other alternatives may be possible. Finally, it is worth mentioning that the Security Controller associates a lifetime to the new IPsec SAs. When this lifetime expires, the NSF will send a sadb-expire notification to the Security Controller in order to start the rekeying process.

4. The flow is protected with the IPsec SA established by the Security Controller.

Instead of installing IPsec policies in the SPD and IPsec SAs in the SAD in step 3 (proactive mode), it is also possible that the Security Controller only installs the SPD entries in step 3 (reactive mode). In such a case, when a data packet requires to be protected with IPsec, the NSF that saw first the data packet will send a sadb-acquire notification that informs the Security Controller that needs SAD entries with the IPsec SAs to process the data packet. In such as reactive mode, since IPsec policies are already installed in the SPD, the Security Controller installs first the new IPsec SAs in NSF A and B with the operations described in step 3 but without sending any IPsec policies. Again, if some of the operations installing the new inbound/outbound IPsec SAs fail, the Security Controller stops the process and performs a rollback operation by deleting any new inbound/outbound SAs that had been successfully installed.

Both NSFs could be two hosts that exchange traffic and require to establish an end-to-end security association to protect their communications (host-to-host) or two gateways (gateway-to-gateway), for example, within an enterprise that needs to protect the traffic between the networks of two branch offices.

Applicability of these configurations appear in current and new networking scenarios. For example, SD-WAN technologies are providing dynamic and on-demand VPN connections between branch offices, or between branches and SaaS cloud services. Beside, IaaS services providing virtualization environments are deployments solutions based on IPsec to provide secure channels between virtual instances (host-to-host) and providing VPN solutions for virtualized networks (gateway-to-gateway).

In general (for IKE and IKE-less cases), this system has various advantages:

1. It allows to create IPsec SAs among two NSFs, based only on the application of general Flow-based Security Policies at the application layer. Thus, administrators can manage all security associations in a centralized point with an abstracted view of the network.

2. Any NSF deployed in the system does not need manual configuration, therefore allowing its deployment in an automated manner.

7.2. Host-to-host or gateway-to-gateway under different Security Controllers

It is also possible that two NSFs (i.e. NSF A and NSF B) are under the control of two different Security Controllers. This may happen, for example, when two organizations, namely Enterprise A and Enterprise B, have their headquarters interconnected through a WAN connection and they both have deployed a SDN-based architecture to provide connectivity to all their clients.

```
                  +-------------+           +-------------+
                  |             |           |             |
        Flow-based|   Security  |<=========>|   Security <--Flow-based
        Sec. Pol.--> Controller |  (3)      |  Controller | Sec. Pol.
              (1) |      A      |           |      B      |   (2)
                  +-------------+           +-------------+
                          |                         |
                          | (4)                 (4) |
                          V                         V
              +--------------------+          +--------------------+
              |    NSF A           |<=======>|   NSF B            |
              |IKEv2/IPsec(SPD/PAD)|          |IKEv2/IPsec(SPD/PAD)|
              +--------------------+  (5)     +--------------------+
```

Figure 5: Different Security Controllers in the IKE case.

Figure 5 describes IKE case when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow-based Security Policies in Security Controller A.

2. The B's administrator establishes general Flow-based Security Policies in Security Controller B.

3. The Security Controller A realizes that protection is required between the NSF A and NSF B, but the NSF B is under the control of another Security Controller (Security Controller B), so it starts negotiations with the other controller to agree on the IPsec SPD policies and IKEv2 credentials for their respective NSFs. NOTE: This may require extensions in the East/West interface.

4. Then, both Security Controllers enforce the IKEv2 credentials, related parameters and the SPD and PAD entries in their respective NSFs.

5. The flow is protected with the IPsec SAs established with IKEv2 between both NSFs.

```
                   +--------------+             +--------------+
                   |              |             |              |
         Flow-based. --->         |             |         <---Flow-based
            Prot.  |   Security   |<===========>|   Security   |Sec.
            Pol.(1)|  Controller  |     (3)     |  Controller  |Pol. (2)
                   |       A      |             |       B      |
                   +--------------+             +--------------+
                           |                               |
                           |  (4)                     (4)  |
                           V                               V
                  +--------------+      (5)       +--------------+
                  |    NSF A     |<==============>|    NSF B     |
                  |IPsec(SPD/SAD)|                |IPsec(SPD/SAD)|
                  +--------------+                +--------------+
```

Figure 6: Different Security Controllers in the IKE-less case.

Figure 6 describes IKE-less case when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow Protection Policies in Security Controller A.

2. The B's administrator establishes general Flow Protection Policies in Security Controller B.

3. The Security Controller A realizes that the flow between NSF B and NSF B MUST be protected. Nevertheless, it notices that NSF B is under the control of another Security Controller B, so it starts negotiations with the other controller to agree on the IPsec SPD and SAD entries that define the IPsec SAs. NOTE: It would worth evaluating IKEv2 as the protocol for the East/West interface in this case.

4. Once the Security Controllers have agreed on the key material and the details of the IPsec SAs, they both enforce this information into their respective NSFs.

5. The flow is protected with the IPsec SAs established by both Security Controllers in their respective NSFs.

8. IANA Considerations

This document registers three URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-ipsec-common
Registrant Contact: The I2NSF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ipsec-ike
Registrant Contact: The I2NSF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless
Registrant Contact: The I2NSF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

This document registers three YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
Name:         ietf-ipsec-common
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ipsec-common
Prefix:       ic
Reference:    RFC XXXX

Name:         ietf-ipsec-ike
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ipsec-ike
Prefix:       ike
Reference:    RFC XXXX

Name:         ietf-ipsec-ikeless
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless
Prefix:       ikeless
Reference:    RFC XXXX
```

9. Security Considerations

First of all, this document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300] and [RFC8192].

On the one hand, it is important to note that there MUST exit a security association between the Security Controller and the NSFs to protect of the critical information (cryptographic keys, configuration parameter, etc...) exchanged between these entities. For example, when NETCONF is used as southbound protocol between the Security Controller and the NSFs, it is defined that TLS or SSH security association MUST be established between both entities.

On the other hand, if encryption is mandatory for all traffic of a NSF, its default policy MUST be to drop (DISCARD) packets to prevent cleartext packet leaks. This default policy MUST be in the startup configuration datastore in the NSF before the NSF contacts with the Security Controller. Moreover, the startup configuration datastore MUST be pre-configured with the required ALLOW policies that allow to communicate the NSF with the Security Controller once the NSF is deployed. This pre-configuration step is not carried out by the Security Controller but by some other entity before the NSF deployment. In this manner, when the NSF starts/reboots, it will always apply first the configuration in the startup configuration before contacting the Security Controller.

Finally, we have divided this section in two parts in order to analyze different security considerations for both cases: NSF with IKEv2 (IKE case) and NSF without IKEv2 (IKE-less case). In general, the Security Controller, as typically in the SDN paradigm, is a target for different type of attacks. Thus, the Security Controller is a key entity in the infrastructure and MUST be protected accordingly. In particular, the Security Controller will handle cryptographic material so that the attacker may try to access this information. Although we can assume this attack will not likely to happen due to the assumed security measurements to protect the Security Controller, it deserves some analysis in the hypothetical case the attack occurs. The impact is different depending on the IKE case or IKE-less case.

9.1. IKE case

In IKE case, the Security Controller sends IKE credentials (PSK, public/private keys, certificates, etc.) to the NSFs using the security association between Security Controller and NSFs. The general recommendation is that the Security Controller MUST NOT store the IKE credentials after distributing them. Moreover, the NSFs MUST NOT allow the reading of these values once they have been applied by the Security Controller (i.e. write only operations). One option is to return always the same value (i.e. all 0s) if a read operation is carried out. If the attacker has access to the Security Controller during the period of time that key material is generated, it might have access to the key material. Since these values are used during NSF authentication in IKEv2, it may impersonate the affected NSFs. Several recommendations are important. If PSK authentication is used in IKEv2, the Security Controller MUST remove the PSK immediately after generating and distributing it. Moreover, the PSK MUST have a proper length (e.g. minimum 128 bit length) and strength. When public/private keys are used, the Security Controller MAY generate both public key and private key. In such a case, the Security Controller MUST remove the associated private key immediately after distributing them to the NSFs. Alternatively, the NSF could generate the private key and export only the public key to the Security Controller. If certificates are used, the NSF MAY generate the private key and exports the public key for certification to the Security Controller. How the NSF generates these cryptographic material (public key/private keys) and export the public key, or it is instructed to do so, it is out of the scope of this document.

9.2. IKE-less case

In the IKE-less case, the Security Controller sends the IPsec SA information to the NSF's SAD that includes the private session keys required for integrity and encryption. The general recommendation is that it MUST NOT store the keys after distributing them. Moreover, the NSFs receiving private key material MUST NOT allow the reading of these values by any other entity (including the Security Controller itself) once they have been applied (i.e. write only operations) into the NSFs. Nevertheless, if the attacker has access to the Security Controller during the period of time that key material is generated, it may obtain these values. In other words, the attacker might be able to observe the IPsec traffic and decrypt, or even modify and re-encrypt the traffic between peers.

9.3. YANG modules

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in these YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

The YANG modules describe configuration data for the IKE case (ietf-ipsec-ike) and IKE-less case (ietf-ipsec-ikeless). There is a common module (ietf-ipsec-common) used in both cases.

For the IKE case (ietf-ipsec-ike):

/ipsec-ike: The entire container in this module is sensitive to write operations. An attacker may add/modify the credentials to be used for the authentication (e.g. to impersonate a NSF), the trust root (e.g. changing the trusted CA certificates), the cryptographic algorithms (allowing a downgrading attack), the IPsec policies (e.g. by allowing leaking of data traffic by changing to a allow policy), and in general changing the IKE SA conditions and credentials between any NSF.

For the IKE-less case (ietf-ipsec-ikeless):

/ipsec-ikeless: The entire container in this module is sensitive to write operations. An attacker may add/modify/ delete any IPsec policies (e.g. by allowing leaking of data traffic by changing to a allow policy) in the /ipsec-ikeless/ spd container, and add/modify/delete any IPsec SAs between two NSF by means of /ipsec-ikeless/sad container and, in general changing any IPsec SAs and IPsec policies between any NSF.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

For the IKE case (ietf-ipsec-ike):

/ipsec-ike/pad: This container includes sensitive information to read operations. This information should never be returned to a client. For example, cryptographic material configured in the NSFs: peer-authentication/pre-shared/secret and peer-authentication/digital-signature/private-key are already protected by the NACM extension "default-deny-all" in this document.

For the IKE-less case (ietf-ipsec-ikeless):

/ipsec-ikeless/sad/ipsec-sa-config/esp-sa: This container includes symmetric keys for the IPsec SAs. For example, encryption/key contains a ESP encryption key value and encryption/iv contains a initialization vector value. Similarly, integrity/key has ESP integrity key value. Those values must not be read by anyone and are protected by the NACM extension "default-deny-all" in this document.

10. Acknowledgements

Authors want to thank Paul Wouters, Valery Smyslov, Sowmini Varadhan, David Carrel, Yoav Nir, Tero Kivinen, Martin Bjorklund, Graham Bartlett, Sandeep Kampati, Linda Dunbar, Carlos J. Bernardos, Alejandro Perez-Mendez, Alejandro Abad-Carrascosa, Ignacio Martinez and Ruben Ricart for their valuable comments.

11. References

11.1. 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>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

11.2. Informative References

[I-D.carrel-ipsecme-controller-ike]
Carrel, D. and B. Weiss, "IPsec Key Exchange using a Controller", draft-carrel-ipsecme-controller-ike-01 (work in progress), March 2019.

[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-08 (work in progress), July 2019.

[I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-01 (work in progress), June 2015.

[ITU-T.X.1252]
"Baseline Identity Management Terms and Definitions", April 2010.

[ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[ITU-T.Y.3300]
"Recommendation ITU-T Y.3300", June 2014.

[libreswan]
The Libreswan Project, "Libreswan VPN software", August 2019.

[netconf-vpn]
Stefan Wallin, "Tutorial: NETCONF and YANG", January 2014.

[ONF-OpenFlow]
ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.

[ONF-SDN-Architecture]
"SDN Architecture", June 2014.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, DOI 10.17487/RFC2367, July 1998, <https://www.rfc-editor.org/info/rfc2367>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <https://www.rfc-editor.org/info/rfc6071>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[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>.

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

[strongswan] CESNET, "StrongSwan: the OpenSource IPsec-based VPN Solution", August 2019.

Appendix A. Appendix A: Common YANG model for IKE and IKE-less cases

```
<CODE BEGINS> file "ietf-ipsec-common@2019-08-05.yang"

module ietf-ipsec-common {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-common";
    prefix "ipsec-common";

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

    organization "IETF I2NSF Working Group";

    contact
    "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/about/>
     WG List: <mailto:i2nsf@ietf.org>

    Author: Rafael Marin-Lopez
            <mailto:rafa@um.es>

    Author: Gabriel Lopez-Millan
            <mailto:gabilm@um.es>

    Author: Fernando Pereniguez-Garcia
            <mailto:fernando.pereniguez@cud.upct.es>
    ";

    description
        "Common Data model for the IKE and IKE-less cases
         defined by the SDN-based IPsec flow protection service.

        Copyright (c) 2019 IETF Trust and the persons
        identified as authors of the code.  All rights reserved.
        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the
        Simplified BSD License set forth in Section 4.c of the
        IETF Trust's Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX;;
        see the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
```

```
              'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
              document are to be interpreted as described in BCP 14
              (RFC 2119) (RFC 8174) when, and only when, they appear
              in all capitals, as shown here.";

          revision "2019-08-05" {
              description "Revision 06";
              reference "RFC XXXX: YANG Groupings and typedef
                         for IKE and IKE-less case";
          }

          typedef encryption-algorithm-type {
              type uint16;
              description
                  "The encryption algorithm is specified with a 16-bit
                  number extracted from IANA Registry. The acceptable
                  values MUST follow the requirement levels for
                  encryption algorithms for ESP and IKEv2.";
              reference
                   "IANA Registry- Transform Type 1 - Encryption
                   Algorithm Transform IDs. RFC 8221 - Cryptographic
                   Algorithm Implementation Requirements and Usage
                   Guidance for Encapsulating Security Payload (ESP)
                   and Authentication Header (AH) and RFC 8247 -
                   Algorithm Implementation Requirements and Usage
                   Guidance for the Internet Key Exchange Protocol
                   Version 2 (IKEv2).";
          }

          typedef integrity-algorithm-type {
              type uint16;
              description
                  "The integrity algorithm is specified with a 16-bit
                  number extracted from IANA Registry.
                  The acceptable values MUST follow the requirement
                  levels for encryption algorithms for ESP and IKEv2.";
              reference
                  "IANA Registry- Transform Type 3 - Integrity
                   Algorithm Transform IDs. RFC 8221 - Cryptographic
                   Algorithm Implementation Requirements and Usage
                   Guidance for Encapsulating Security Payload (ESP)
                   and Authentication Header (AH) and RFC 8247 -
                   Algorithm Implementation Requirements and Usage
                   Guidance for the Internet Key Exchange Protocol
                   Version 2 (IKEv2).";
          }

          typedef ipsec-mode {
```

```
               type enumeration {
                   enum transport {
                       description
                           "IPsec transport mode. No Network Address
                            Translation (NAT) support.";
                   }
                   enum tunnel {
                       description "IPsec tunnel mode.";
                   }
               }
               description
                   "Type definition of IPsec mode: transport or
                    tunnel.";
               reference
                   "Section 3.2 in RFC 4301.";
           }

           typedef esp-encap {
               type enumeration {
                   enum espintcp {
                       description
                           "ESP in TCP encapsulation.";
                       reference
                           "RFC 8229 - TCP Encapsulation of IKE and
                            IPsec Packets.";
                   }
                   enum espintls {
                       description
                           "ESP in TCP encapsulation using TLS.";
                       reference
                           "RFC 8229 - TCP Encapsulation of IKE and
                            IPsec Packets.";
                   }
                   enum espinudp {
                       description
                           "ESP in UDP encapsulation.";
                       reference
                           "RFC 3948 - UDP Encapsulation of IPsec ESP
                           Packets.";
                   }
                   enum none {
                       description
                           "NOT ESP encapsulation.";
                   }
               }
               description
                   "Types of ESP encapsulation when Network Address
                    Translation (NAT) is present between two NSFs.";
```

```
            reference
                "RFC 8229 - TCP Encapsulation of IKE and IPsec
                 Packets and RFC 3948 - UDP Encapsulation of IPsec
                 ESP Packets.";
        }

        typedef ipsec-protocol-parameters {
            type enumeration {
                enum esp { description "IPsec ESP protocol."; }
            }
            description
                "Only the Encapsulation Security Protocol (ESP) is
                 supported but it could be extended in the future.";
            reference
                "RFC 4303- IP Encapsulating Security Payload
                (ESP).";

        }

        typedef lifetime-action {
            type enumeration {
                enum terminate-clear {
                    description
                        "Terminates the IPsec SA and allows the
                         packets through.";
                }
                enum terminate-hold {
                    description
                        "Terminates the IPsec SA and drops the
                         packets.";
                }
                enum replace  {
                    description
                        "Replaces the IPsec SA with a new one:
                        rekey. ";
                }
            }
            description
                "When the lifetime of an IPsec SA expires an action
                 needs to be performed over the IPsec SA that
                 reached the lifetime. There are three posible
                 options: terminate-clear, terminate-hold and
                 replace.";
            reference
                "Section 4.5 in RFC 4301.";
        }

        typedef ipsec-traffic-direction {
```

```
                type enumeration {
                    enum inbound {
                        description "Inbound traffic.";
                    }
                    enum outbound {
                        description "Outbound traffic.";
                    }
                }
                description
                    "IPsec traffic direction is defined in two
                     directions: inbound and outbound. From a NSF
                     perspective inbound means the traffic that enters
                     the NSF and outbound is the traffic that is sent
                     from the NSF.";
                reference
                    "Section 5 in RFC 4301.";
            }

            typedef ipsec-spd-action {
                type enumeration {
                    enum protect {
                        description
                            "PROTECT the traffic with IPsec.";
                    }
                    enum bypass {
                        description
                            "BYPASS the traffic. The packet is forwarded
                             without IPsec protection.";
                    }
                    enum discard {
                        description
                            "DISCARD the traffic. The IP packet is
                             discarded.";
                    }
                }
                description
                    "The action when traffic matches an IPsec security
                     policy. According to RFC 4301 there are three
                     possible values: BYPASS, PROTECT AND DISCARD";
                reference
                    "Section 4.4.1 in RFC 4301.";
            }

            typedef ipsec-inner-protocol {
                type union {
                    type uint8;
                    type enumeration {
                        enum any {
```

```
                        value 256;
                        description
                            "Any IP protocol number value.";
                    }
                }
            }
            default any;
            description
                "IPsec protection can be applied to specific IP
                 traffic and layer 4 traffic (TCP, UDP, SCTP, etc.)
                 or ANY protocol in the IP packet payload. We
                 specify the IP protocol number with an uint8 or
                 ANY defining an enumerate with value 256 to
                 indicate the protocol number.";
            reference
                "Section 4.4.1.1 in RFC 4301.
                 IANA Registry - Protocol Numbers.";
        }

        grouping encap {
            description
                "This group of nodes allows to define the type of
                 encapsulation in case NAT traversal is
                 required and port information.";
            leaf espencap {
                type esp-encap;
                description
                    "ESP in TCP, ESP in UDP or ESP in TLS.";
            }
            leaf sport {
                type inet:port-number;
                default 4500;
                description
                    "Encapsulation source port.";
            }
            leaf dport {
                type inet:port-number;
                default 4500;
                description
                    "Encapsulation destination port.";
            }

            leaf-list oaddr {
                type inet:ip-address;
                description
                    "If required, this is the original address that
                     was used before NAT was applied over the Packet.
                     ";
```

```
            }
            reference
                "RFC 3947 and RFC 8229.";
        }

        grouping lifetime {
            description
                "Different lifetime values limited to an IPsec SA.";
            leaf time {
                type uint32;
                default 0;
                description
                    "Time in seconds since the IPsec SA was added.
                     For example, if this value is 180 seconds it
                     means the IPsec SA expires in 180 seconds since
                     it was added. The value 0 implies infinite.";
            }
            leaf bytes {
                type uint32;
                default 0;
                description
                    "If the IPsec SA processes the number of bytes
                    expressed in this leaf, the IPsec SA expires and
                    should be rekeyed. The value 0 implies
                    infinite.";
            }
            leaf packets {
                type uint32;
                default 0;
                description
                    "If the IPsec SA processes the number of packets
                    expressed in this leaf, the IPsec SA expires and
                    should be rekeyed. The value 0 implies
                    infinite.";
            }
            leaf idle {
                type uint32;
                default 0;
                description
                    "When a NSF stores an IPsec SA, it
                     consumes system resources. In an idle NSF this
                     is a waste of resources. If the IPsec SA is idle
                     during this number of seconds the IPsec SA
                     should be removed. The value 0 implies
                     infinite.";
            }
            reference
                "Section 4.4.2.1 in RFC 4301.";
```

```
           }

           grouping port-range  {
               description
                   "This grouping defines a port range, such as
                    expressed in RFC 4301. For example: 1500 (Start
                    Port Number)-1600 (End Port Number). A port range
                    is used in the Traffic Selector.";

               leaf start {
                   type inet:port-number;
                   description
                       "Start port number.";
               }
               leaf end {
                   type inet:port-number;
                   description
                       "End port number.";
               }
               reference "Section 4.4.1.2 in RFC 4301.";
           }

           grouping tunnel-grouping {
               description
                   "The parameters required to define the IP tunnel
                    endpoints when IPsec SA requires tunnel mode. The
                    tunnel is defined by two endpoints: the local IP
                    address and the remote IP address.";

               leaf local {
                   type inet:ip-address;
                   mandatory true;
                   description
                       "Local IP address' tunnel endpoint.";
               }
               leaf remote {
                   type inet:ip-address;
                   mandatory true;
                   description
                       "Remote IP address' tunnel endpoint.";
               }
               leaf df-bit {
                   type enumeration {
                       enum clear {
                           description
                               "Disable the DF (Don't Fragment) bit
                                from the outer header. This is the
                                default value.";
```

```
                    }
                    enum set {
                        description
                            "Enable the DF bit in the outer header.";
                    }
                    enum copy {
                        description
                            "Copy the DF bit to the outer header.";
                    }
                }
                default clear;
                description
                    "Allow configuring the DF bit when encapsulating
                     tunnel mode IPsec traffic. RFC 4301 describes
                     three options to handle the DF bit during
                     tunnel encapsulation: clear, set and copy from
                     the inner IP header.";
                reference
                    "Section 8.1 in RFC 4301.";
            }
            leaf bypass-dscp {
                type boolean;
                default true;
                description
                    "If DSCP (Differentiated Services Code Point)
                     values in the inner header have to be used to
                     select one IPsec SA among several that match
                     the traffic selectors for an outbound packet";
                reference
                    "Section 4.4.2.1. in RFC 4301.";
            }
            leaf dscp-mapping {
                type yang:hex-string;
                description
                    "DSCP values allowed for packets carried over
                     this IPsec SA.";
                reference
                    "Section 4.4.2.1. in RFC 4301.";
            }
            leaf ecn {
                type boolean;
                default false;
                description
                    "Explicit Congestion Notification (ECN). If true
                     copy CE bits to inner header.";
                reference
                    "Section 5.1.2 and Annex C in RFC 4301.";
            }
```

```
           }

           grouping selector-grouping {
               description
                   "This grouping contains the definition of a Traffic
                    Selector, which is used in the IPsec policies and
                    IPsec SAs.";

               leaf local-subnet {
                   type inet:ip-prefix;
                   mandatory true;
                   description
                       "Local IP address subnet.";
               }
               leaf remote-subnet {
                   type inet:ip-prefix;
                   mandatory true;
                   description
                       "Remote IP address subnet.";
               }
               leaf inner-protocol {
                   type ipsec-inner-protocol;
                   default any;
                   description
                       "Inner Protocol that is going to be
                       protected with IPsec.";
               }
               list local-ports {
                   key "start end";
                   uses port-range;
                   description
                       "List of local ports. When the inner
                        protocol is ICMP this 16 bit value represents
                        code and type.";
               }
               list remote-ports {
                   key "start end";
                   uses port-range;
                   description
                       "List of remote ports. When the upper layer
                       protocol is ICMP this 16 bit value represents
                       code and type.";
               }
               reference
                   "Section 4.4.1.2 in RFC 4301.";
           }

           grouping ipsec-policy-grouping {
```

```
               description
                   "Holds configuration information for an IPsec SPD
                    entry.";

               leaf anti-replay-window {
                   type uint64;
                   default 32;
                   description
                       "A 64-bit counter used to determine whether an
                        inbound ESP packet is a replay.";
                   reference
                       "Section 4.4.2.1 in RFC 4301.";
               }
               container traffic-selector {
                   description
                       "Packets are selected for
                        processing actions based on the IP and inner
                        protocol header information, selectors,
                        matched against entries in the SPD.";
                   uses selector-grouping;
                   reference
                       "Section 4.4.4.1 in RFC 4301.";
               }
               container processing-info {
                   description
                       "SPD processing. If the required processing
                        action is protect, it contains the required
                        information to process the packet.";
                   leaf action {
                       type ipsec-spd-action;
                       default discard;
                       description
                           "If bypass or discard, container
                           ipsec-sa-cfg is empty.";
                   }
                   container ipsec-sa-cfg {
                       when "../action = 'protect'";
                       description
                           "IPSec SA configuration included in the SPD
                           entry.";
                       leaf pfp-flag {
                           type boolean;
                           default false;
                           description
                                "Each selector has a Populate From
                                 Packet (PFP) flag. If asserted for a
                                 given selector X, the flag indicates
                                 that the IPSec SA to be created should
```

```
                          take its value (local IP address,
                          remote IP address, Next Layer
                          Protocol, etc.) for X from the value
                          in the packet. Otherwise, the IPsec SA
                          should take its value(s) for X from
                          the value(s) in the SPD entry.";
                 }
                 leaf ext-seq-num {
                     type boolean;
                     default false;
                     description
                          "True if this IPsec SA is using extended
                          sequence numbers. True 64 bit counter,
                          False 32 bit.";
                 }
                 leaf seq-overflow {
                     type boolean;
                     default false;
                     description
                         "The flag indicating whether
                         overflow of the sequence number
                         counter should prevent transmission
                         of additional packets on the IPsec
                         SA (false) and, therefore needs to
                         be rekeyed, or whether rollover is
                         permitted (true). If Authenticated
                         Encryption with Associated Data
                         (AEAD) is used this flag MUST be
                         false.";
                 }
                 leaf stateful-frag-check {
                     type boolean;
                     default false;
                     description
                         "Indicates whether (true) or not (false)
                          stateful fragment checking applies to
                          the IPsec SA to be created.";
                 }
                 leaf mode {
                     type ipsec-mode;
                     default transport;
                     description
                         "IPsec SA has to be processed in
                          transport or tunnel mode.";
                 }
                 leaf protocol-parameters {
                     type ipsec-protocol-parameters;
                     default esp;
```

```
                        description
                             "Security protocol of the IPsec SA:
                             Only ESP is supported but it could be
                             extended in the future.";
                    }
                    container esp-algorithms {
                        when "../protocol-parameters = 'esp'";
                        description
                             "Configuration of Encapsulating
                             Security Payload (ESP) parameters and
                             algorithms.";
                        leaf-list integrity {
                            type integrity-algorithm-type;
                            default 0;
                            ordered-by user;
                            description
                                "Configuration of ESP authentication
                                based on the specified integrity
                                algorithm. With AEAD algorithms,
                                the integrity node is not
                                used.";
                            reference
                                "Section 3.2 in RFC 4303.";
                        }
                        leaf-list encryption {
                            type encryption-algorithm-type;
                            default 20;
                            ordered-by user;
                            description
                                "Configuration of ESP encryption
                                algorithms. The default value is
                                20 (ENCR_AES_GCM_16).";
                            reference
                                "Section 3.2 in RFC 4303.";
                        }
                        leaf tfc-pad {
                            type boolean;
                            default false;
                            description
                                "If Traffic Flow Confidentiality
                                 (TFC) padding for ESP encryption
                                 can be used (true) or not (false)";
                            reference
                                "Section 2.7 in RFC 4303.";
                        }
                        reference
                            "RFC 4303.";
                    }
```

```
                        container tunnel {
                            when "../mode = 'tunnel'";
                            uses tunnel-grouping;
                            description
                               "IPsec tunnel endpoints definition.";
                        }
                    }
                    reference
                        "Section 4.4.1.2 in RFC 4301.";
                }
                container spd-mark {
                        description
                            "The Mark to set for the IPsec SA of this
                             connection. This option is only available
                             on linux NETKEY/XFRM kernels. It can be
                             used with iptables to create custom
                             iptables rules using CONNMARK. It can also
                             be used with Virtual Tunnel Interfaces
                             (VTI) to direct marked traffic to
                             specific vtiXX devices.";
                        leaf mark {
                            type uint32;
                            default 0;
                            description
                                "Mark used to match XFRM policies and
                                 states.";
                        }
                        leaf mask {
                            type yang:hex-string;
                            default 00:00:00:00;
                            description
                                "Mask used to match XFRM policies and
                                states.";
                        }
                }
            }
        }

        <CODE ENDS>
```

Appendix B. Appendix B: YANG model for IKE case

```
        <CODE BEGINS> file "ietf-ipsec-ike@2019-08-05.yang"
```

```
       module ietf-ipsec-ike {
           yang-version 1.1;
           namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-ike";
           prefix "ike";

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

           import ietf-crypto-types {
               prefix ct;
               reference
                   "draft-ietf-netconf-crypto-types-10:
                   Common YANG Data Types for Cryptography.";
           }

           import ietf-ipsec-common {
               prefix ic;
               reference
                   "RFC XXXX: module ietf-ipsec-common, revision
                    2019-08-05.";
           }

           import ietf-netconf-acm {
                  prefix nacm;
                  reference
                    "RFC 8341: Network Configuration Access Control
                     Model.";
           }

           organization "IETF I2NSF Working Group";

           contact
           "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/about/>
            WG List: <mailto:i2nsf@ietf.org>

           Author: Rafael Marin-Lopez
                   <mailto:rafa@um.es>

           Author: Gabriel Lopez-Millan
                   <mailto:gabilm@um.es>

           Author: Fernando Pereniguez-Garcia
                   <mailto:fernando.pereniguez@cud.upct.es>
           ";

           description

           "This module contains IPSec IKE case model for the SDN-based
```

```
             IPsec flow protection service. An NSF will implement this
             module.


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

            Redistribution and use in source and binary forms, with or
            without modification, is permitted pursuant to, and subject
            to the license terms contained in, the Simplified BSD License
            set forth in Section 4.c of the IETF Trust's Legal Provisions
            Relating to IETF Documents
            (http://trustee.ietf.org/license-info).

            This version of this YANG module is part of RFC XXXX; see
            the RFC itself for full legal notices.

            The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
            'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
            'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
            document are to be interpreted as described in BCP 14
            (RFC 2119) (RFC 8174) when, and only when, they appear
            in all capitals, as shown here.";

            revision "2019-08-05" {
                description "Revision 6";
                reference
                    "RFC XXXX: YANG model for IKE case.";
            }

            typedef ike-spi {
                type uint64 { range "0..max"; }
                description
                    "Security Parameter Index (SPI)'s IKE SA.";
                reference
                    "Section 2.6 in RFC 7296.";
            }

            typedef autostartup-type {
                type enumeration {
                    enum add {
                        description
                            "IKE/IPsec configuration is only loaded into
                             IKE implementation but IKE/IPsec SA is not
                             started.";
                    }
                    enum on-demand {
                        description
```

```
                        "IKE/IPsec configuration is loaded
                        into IKE implementation. The IPsec policies
                        are transferred to the NSF's kernel but the
                        IPsec SAs are not established immediately.
                        The IKE implementation will negotiate the
                        IPsec SAs when the NSF's kernel requests it
                        (i.e. through an ACQUIRE notification).";
                }
                enum start {
                    description "IKE/IPsec configuration is loaded
                    and transferred to the NSF's kernel, and the
                    IKEv2 based IPsec SAs are established
                    immediately without waiting any packet.";
                }
            }
            description
                "Different policies to set IPsec SA configuration
                 into NSF's kernel when IKEv2 implementation has
                 started.";
        }



        typedef pfs-group {
            type uint16;
            description
                "DH groups for IKE and IPsec SA rekey.";
            reference
                "Section 3.3.2 in RFC 7296. Transform Type 4 -
                 Diffie-Hellman Group Transform IDs in IANA Registry
                   - Internet Key Exchange Version 2 (IKEv2)
                 Parameters.";
        }

        typedef auth-protocol-type {
            type enumeration {
                enum ikev2 {
                    value 2;
                    description
                        "IKEv2 authentication protocol. It is the
                         only defined right now. An enum is used for
                         further extensibility.";
                }
            }
            description
                "IKE authentication protocol version specified in the
                 Peer Authorization Database (PAD). It is defined as
                 enumerate to allow new IKE versions in the
```

```
                  future.";
              reference
                  "RFC 7296.";
          }

          typedef auth-method-type {
              type enumeration {
                  enum pre-shared {
                      description
                          "Select pre-shared key as the
                          authentication method.";
                      reference
                          "RFC 7296.";
                  }
                  enum eap {
                      description
                          "Select EAP as the authentication method.";
                      reference
                          "RFC 7296.";
                  }
                  enum digital-signature {
                      description
                          "Select digital signature method.";
                      reference
                          "RFC 7296 and RFC 7427.";
                  }
                  enum null {
                      description
                          "Null authentication.";
                      reference
                          "RFC 7619.";
                  }

              }
              description
                  "Peer authentication method specified in the Peer
                   Authorization Database (PAD).";
          }

          container ipsec-ike {
              description
                  "IKE configuration for a NSF. It includes PAD
                   parameters, IKE connections information and state
                   data.";

              container pad {
                  description
                     "Configuration of Peer Authorization Database
```

```
                       (PAD). The PAD contains information about IKE
                       peer (local and remote). Therefore, the Security
                       Controller also stores authentication
                       information for this NSF and can include
                       several entries for the local NSF not only
                       remote peers. Storing local and remote
                       information makes possible to specify that this
                       NSF with identity A will use some particular
                       authentication with remote NSF with identity B
                       and what are the authentication mechanisms
                       allowed to B.";
                   list pad-entry {
                       key "name";
                       ordered-by user;
                       description
                           "Peer Authorization Database (PAD) entry. It
                            is a list of PAD entries ordered by the
                            Security Controller.";
                       leaf name {
                           type string;
                           description
                               "PAD unique name to identify this
                                entry.";
                       }
                       choice identity {
                           mandatory true;
                           description
                               "A particular IKE peer will be
                               identified by one of these identities.
                               This peer can be a remote peer or local
                               peer (this NSF).";
                           reference
                               "Section 4.4.3.1 in RFC 4301.";
                           case ipv4-address{
                               leaf ipv4-address {
                                   type inet:ipv4-address;
                                   description
                                       "Specifies the identity as a
                                        single four (4) octet.";
                               }
                           }
                           case ipv6-address{
                               leaf ipv6-address {
                                   type inet:ipv6-address;
                                   description
                                       "Specifies the identity as a
                                        single sixteen (16) octet IPv6
                                        address. An example is
```

```
                                      2001:DB8:0:0:8:800:200C:417A.";
                             }
                         }
                         case fqdn-string {
                             leaf fqdn-string {
                                 type inet:domain-name;
                                 description
                                     "Specifies the identity as a
                                      Fully-QualifiedDomain Name
                                      (FQDN) string. An example is:
                                      example.com. The string MUST
                                      NOT contain any terminators
                                      (e.g., NULL, CR, etc.).";
                             }
                         }
                         case rfc822-address-string {
                             leaf rfc822-address-string {
                                 type string;
                                 description
                                     "Specifies the identity as a
                                      fully-qualified RFC822 email
                                      address string. An example is,
                                      jsmith@example.com. The string
                                      MUST NOT contain any
                                      terminators e.g., NULL, CR,
                                      etc.).";
                                 reference
                                     "RFC 822.";
                             }
                         }
                         case dnx509 {
                             leaf dnx509 {
                                 type string;
                                 description
                                     "Specifies the identity as a
                                      ASN.1 X.500 Distinguished
                                      Name. An example is
                                      C=US,O=Example
                                      Organisation,CN=John Smith.";
                                 reference
                                     "RFC 2247.";
                             }
                         }
                         case gnx509 {
                             leaf gnx509 {
                                 type string;
                                 description
                                     "ASN.1 X.509 GeneralName. RFC
```

```
                                         3280.";
                                }
                            }
                            case id-key {
                                leaf id-key {
                                    type string;
                                    description
                                        "Opaque octet stream that may be
                                         used to pass vendor-specific
                                         information for proprietary
                                         types of identification.";
                                    reference
                                        "Section 3.5 in RFC 7296.";
                                }
                            }
                            case id-null {
                                leaf id-null {
                                    type empty;
                                    description
                                        "ID_NULL identification used
                                         when IKE identification payload
                                         is not used." ;
                                    reference
                                        "RFC 7619.";
                                }
                            }
                        }
                        leaf auth-protocol {
                            type auth-protocol-type;
                            default ikev2;
                            description
                                "Only IKEv2 is supported right now but
                                 other authentication protocols may be
                                 supported in the future.";
                        }
                        container peer-authentication {
                            description
                                "This container allows the Security
                                 Controller to configure the
                                 authentication method (pre-shared key,
                                 eap, digitial-signature, null) that
                                 will use a particular peer and the
                                 credentials, which will depend on the
                                 selected authentication method.";
                            leaf auth-method {
                               type auth-method-type;
                               default pre-shared;
                               description
```

```
                                   "Type of authentication method
                                   (pre-shared, eap, digital signature,
                                    null).";
                              reference
                                  "Section 2.15 in RFC 7296.";
                           }
                           container eap-method {
                               when "../auth-method = 'eap'";
                               leaf eap-type {
                                   type uint8;
                                   mandatory true;
                                   description
                                       "EAP method type. This
                                       information provides the
                                       particular EAP method to be
                                       used. Depending on the EAP
                                       method, pre-shared keys or
                                       certificates may be used.";
                               }
                               description
                                   "EAP method description used when
                                   authentication method is 'eap'.";
                               reference
                                   "Section 2.16 in RFC 7296.";
                           }
                           container pre-shared {
                               when
                                   "../auth-method[.='pre-shared' or
                                    .='eap']";
                               leaf secret {
                                   nacm:default-deny-all;
                                   type yang:hex-string;
                                   description
                                       "Pre-shared secret value. The
                                        NSF has to prevent read access
                                        to this value for security
                                        reasons.";
                               }
                               description
                                   "Shared secret value for PSK or
                                    EAP method authentication based on
                                    PSK.";
                           }
                           container digital-signature {
                               when
                                "../auth-method[.='digital-signature'
                               or .='eap']";
                               leaf ds-algorithm {
```

```
                                    type uint8;
                                    description
                                        "The digital signature
                                        algorithm is specified with a
                                        value extracted from the IANA
                                        Registry. Depending on the
                                        algorithm, the following leafs
                                        must contain information. For
                                        example if digital signature
                                        involves a certificate then leaf
                                        'cert-data' and 'private-key'
                                        will contain this information.";
                                    reference
                                        "IKEv2 Authentication Method -
                                         IANA Registry - Internet Key
                                         Exchange Version 2 (IKEv2)
                                         Parameters.";
                                }

                                choice public-key {
                                    mandatory true;
                                    leaf raw-public-key {
                                        type binary;
                                        description
                                          "A binary that contains the
                                          value of the public key.  The
                                          interpretation of the content
                                          is defined by the digital
                                          signature algorithm. For
                                          example, an RSA key is
                                          represented as RSAPublicKey as
                                          defined in RFC 8017, and an
                                          Elliptic Curve Cryptography
                                          (ECC) key is represented
                                          using the 'publicKey'
                                          described in RFC 5915.";
                                    reference
                                          "RFC XXX: Common YANG Data
                                          Types for Cryptography.";
                                    }
                                    leaf cert-data {
                                        type ct:x509;
                                        description
                                            "X.509 certificate data -
                                             PEM4.";
                                        reference
                                            "RFC XXX: Common YANG Data
                                            Types for Cryptography.";
```

```
                                    }
                                    description
                                        "If the Security Controller
                                         knows that the NSF
                                         already owns a private key
                                         associated to this public key
                                         (the NSF generated the pair
                                         public key/private key out of
                                         band), it will only configure
                                         one of the leaf of this
                                         choice. The NSF, based on
                                         the public key value can know
                                         the private key to be used.";
                                }
                                leaf private-key {
                                    nacm:default-deny-all;
                                    type binary;
                                    description
                                        "A binary that contains the
                                         value of the private key. The
                                         interpretation of the content
                                         is defined by the digital
                                         signature algorithm. For
                                         example, an RSA key is
                                         represented as RSAPrivateKey as
                                         defined in RFC 8017, and an
                                         Elliptic Curve Cryptography
                                         (ECC) key is represented as
                                         ECPrivateKey as defined in RFC
                                         5915.";
                                    reference
                                        "RFC XXX: Common YANG Data
                                        Types for Cryptography.";
                                }
                                leaf-list ca-data {
                                    type ct:x509;
                                    description
                                        "List of trusted Certification
                                        Authorities (CA) certificates
                                        encoded using ASN.1
                                        distinguished encoding rules
                                        (DER).";
                                    reference
                                        "RFC XXX: Common YANG Data
                                        Types for Cryptography.";
                                }
                                leaf crl-data {
                                    type ct:crl;
```

```
                             description
                                "A CertificateList structure, as
                                 specified in RFC 5280,
                                 encoded using ASN.1
                                 distinguished encoding rules
                                 (DER),as specified in ITU-T
                                 X.690.";
                             reference
                                 "RFC XXX: Common YANG Data Types
                                  for Cryptography.";
                         }
                         leaf crl-uri  {
                             type inet:uri;
                             description
                                 "X.509 CRL certificate URI.";
                         }
                         leaf oscp-uri {
                             type inet:uri;
                             description
                                 "OCSP URI.";
                         }
                         description
                             "Digital Signature container.";

                     } /*container digital-signature*/
                 } /*container peer-authentication*/
             }
         }

         list conn-entry {
             key "name";
             description
                 "IKE peer connection information. This list
                 contains the IKE connection for this peer
                 with other peers. This will be translated in
                 real time by IKE Security Associations
                 established with these nodes.";
             leaf name {
                 type string;
                 mandatory true;
                 description
                     "Identifier for this connection
                      entry.";
             }
             leaf autostartup {
                   type autostartup-type;
                   default add;
                   description
```

```
                    "By-default: Only add configuration
                     without starting the security
                     association.";
          }
          leaf initial-contact {
              type boolean;
              default false;
              description
                  "The goal of this value is to deactivate the
                  usage of INITIAL_CONTACT notification
                  (true). If this flag remains to false it
                  means the usage of the INITIAL_CONTACT
                  notification will depend on the IKEv2
                  implementation.";
          }
          leaf version {
              type auth-protocol-type;
              default ikev2;
              description
                 "IKE version. Only version 2 is supported
                 so far.";
          }
          leaf fragmentation {
              type boolean;
              default false;
              description
                  "Whether or not to enable IKE
                   fragmentation as per RFC 7383 (true or
                   false).";
              reference
                  "RFC 7383.";
          }
          container ike-sa-lifetime-soft {
              description
                  "IKE SA lifetime soft. Two lifetime values
                   can be configured: either rekey time of the
                   IKE SA or reauth time of the IKE SA. When
                   the rekey lifetime expires a rekey of the
                   IKE SA starts. When reauth lifetime
                   expires a IKE SA reauthentication starts.";
             leaf rekey-time {
                  type uint32;
                  default 0;
                  description
                      "Time in seconds between each IKE SA
                      rekey.The value 0 means infinite.";
             }
             leaf reauth-time {
```

```
                           type uint32;
                           default 0;
                           description
                             "Time in seconds between each IKE SA
                             reauthentication. The value 0 means
                             infinite.";
                      }
                      reference
                          "Section 2.8 in RFC 7296.";
                   }
                   container ike-sa-lifetime-hard {
                       description
                           "Hard IKE SA lifetime. When this
                            time is reached the IKE SA is removed.";
                       leaf over-time {
                           type uint32;
                           default 0;
                           description
                               "Time in seconds before the IKE SA is
                                removed. The value 0 means infinite.";
                       }
                       reference
                           "RFC 7296.";
                   }
                   leaf-list authalg {
                       type ic:integrity-algorithm-type;
                       default 12;
                       ordered-by user;
                       description
                          "Authentication algorithm for establishing
                          the IKE SA. This list is ordered following
                          from the higher priority to lower priority.
                          First node of the list will be the algorithm
                          with higher priority. If this list is empty
                          the default integrity algorithm value assumed
                          is NONE.";
                   }
                   leaf-list encalg {
                       type ic:encryption-algorithm-type;
                       default 12;
                       ordered-by user;
                       description
                          "Encryption or AEAD algorithm for the IKE
                          SAs. This list is ordered following
                          from the higher priority to lower priority.
                          First node of the list will be the algorithm
                          with higher priority. If this list is empty
                          the default encryption value assumed is
```

```
                      NULL.";
              }
              leaf dh-group {
                  type pfs-group;
                  default 14;
                  description
                      "Group number for Diffie-Hellman
                      Exponentiation used during IKE_SA_INIT
                      for the IKE SA key exchange.";
              }
              leaf half-open-ike-sa-timer {
                  type uint32;
                  description
                      "Set the half-open IKE SA timeout
                       duration.";
                  reference
                      "Section 2 in RFC 7296.";
              }

              leaf half-open-ike-sa-cookie-threshold {
                  type uint32;
                  description
                      "Number of half-open IKE SAs that activate
                       the cookie mechanism." ;
                  reference
                      "Section 2.6 in RFC 7296.";
              }
              container local {
                  leaf local-pad-entry-name {
                      type string;
                      description
                          "Local peer authentication information.
                           This node points to a specific entry in
                           the PAD where the authorization
                           information about this particular local
                           peer is stored. It MUST match a
                           pad-entry-name.";
                  }
                  description
                      "Local peer authentication information.";
              }
              container remote {
                  leaf remote-pad-entry-name {
                      type string;
                      description
                          "Remote peer authentication information.
                           This node points to a specific entry in
                           the PAD where the authorization
```

```
                               information about this particular
                               remote peer is stored. It MUST match a
                               pad-entry-name.";
                      }
                      description
                          "Remote peer authentication information.";
                  }
                  container encapsulation-type
                  {
                      uses ic:encap;
                      description
                          "This container carries configuration
                          information about the source and destination
                          ports of encapsulation that IKE should use
                          and the type of encapsulation that
                          should use when NAT traversal is required.
                          However, this is just a best effort since
                          the IKE implementation may need to use a
                          different encapsulation as
                          described in RFC 8229.";
                      reference
                          "RFC 8229.";
                  }
                  container spd {
                      description
                          "Configuration of the Security Policy
                          Database (SPD). This main information is
                          placed in the grouping
                          ipsec-policy-grouping.";
                      list spd-entry {
                          key "name";
                          ordered-by user;
                          leaf name {
                              type string;
                              mandatory true;
                              description
                                  "SPD entry unique name to identify
                                  the IPsec policy.";
                          }
                          container ipsec-policy-config {
                              description
                                  "This container carries the
                                  configuration of a IPsec policy.";
                              uses ic:ipsec-policy-grouping;
                          }
                          description
                              "List of entries which will constitute
                              the representation of the SPD. Since we
```

```
                              have IKE in this case, it is only
                              required to send a IPsec policy from
                              this NSF where 'local' is this NSF and
                              'remote' the other NSF. The IKE
                              implementation will install IPsec
                              policies in the NSF's kernel in both
                              directions (inbound and outbound) and
                              their corresponding IPsec SAs based on
                              the information in this SPD entry.";
                      }
                      reference
                          "Section 2.9 in RFC 7296.";
                  }
                  container child-sa-info {
                      leaf-list pfs-groups {
                          type pfs-group;
                          default 0;
                          ordered-by user;
                          description
                              "If non-zero, it is required perfect
                               forward secrecy when requesting new
                               IPsec SA. The non-zero value is
                               the required group number. This list is
                               ordered following from the higher
                               priority to lower priority. First node
                               of the list will be the algorithm
                               with higher priority.";
                      }
                      container child-sa-lifetime-soft {
                          description
                              "Soft IPsec SA lifetime soft.
                               After the lifetime the action is
                               defined in this container
                               in the leaf action.";
                          uses ic:lifetime;
                          leaf action {
                              type ic:lifetime-action;
                              default replace;
                              description
                                  "When the lifetime of an IPsec SA
                                   expires an action needs to be
                                   performed over the IPsec SA that
                                   reached the lifetime. There are
                                   three possible options:
                                   terminate-clear, terminate-hold and
                                   replace.";
                          reference
                              "Section 4.5 in RFC 4301 and Section 2.8
```

```
                         in RFC 7296.";
                    }
                }
                container child-sa-lifetime-hard {
                    description
                        "IPsec SA lifetime hard. The action will
                         be to terminate the IPsec SA.";
                    uses ic:lifetime;
                    reference
                        "Section 2.8 in RFC 7296.";
                }
                description
                    "Specific information for IPsec SAs
                    SAs. It includes PFS group and IPsec SAs
                    rekey lifetimes.";
            }
            container state {
                config false;

                leaf initiator {
                    type boolean;
                    description
                        "It is acting as initiator for this
                         connection.";
                }
                leaf initiator-ikesa-spi {
                    type ike-spi;
                    description
                        "Initiator's IKE SA SPI.";
                }
                leaf responder-ikesa-spi {
                    type ike-spi;
                    description
                        "Responder's IKE SA SPI.";
                }
                leaf nat-local {
                    type boolean;
                    description
                        "True, if local endpoint is behind a
                         NAT.";
                }
                leaf nat-remote {
                    type boolean;
                    description
                        "True, if remote endpoint is behind
                        a NAT.";
                }
```

```
                        container encapsulation-type
                        {
                            uses ic:encap;
                            description
                                "This container provides information
                                about the source and destination
                                ports of encapsulation that IKE is
                                using, and the type of encapsulation
                                when NAT traversal is required.";
                            reference
                                "RFC 8229.";
                        }
                        leaf established {
                            type uint64;
                            description
                                "Seconds since this IKE SA has been
                                 established.";
                        }
                        leaf current-rekey-time {
                            type uint64;
                            description
                                "Seconds before IKE SA must be rekeyed.";
                        }
                        leaf current-reauth-time {
                            type uint64;
                            description
                                "Seconds before IKE SA must be
                                 re-authenticated.";
                        }
                        description
                            "IKE state data for a particular
                             connection.";
                    } /* ike-sa-state */
                } /* ike-conn-entries */

                container number-ike-sas {
                    config false;
                    leaf total {
                        type uint64;
                        description
                            "Total number of active IKE SAs.";
                    }
                    leaf half-open {
                        type uint64;
                        description
                            "Number of half-open active IKE SAs.";
                    }
                    leaf half-open-cookies {
```

```
                        type uint64;
                        description
                            "Number of half open active IKE SAs with
                             cookie activated.";
                    }
                    description
                        "General information about the IKE SAs. In
                        particular, it provides the current number of
                        IKE SAs.";
                }
            }  /* container ipsec-ike */
        }


        <CODE ENDS>
```

Appendix C. Appendix C: YANG model for IKE-less case

```
        <CODE BEGINS> file "ietf-ipsec-ikeless@2019-08-05.yang"

        module ietf-ipsec-ikeless {

            yang-version 1.1;
            namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless";

            prefix "ikeless";

            import ietf-yang-types { prefix yang; }

            import ietf-ipsec-common {
                prefix ic;
                reference
                    "Common Data model for SDN-based IPSec
                     configuration.";
            }

            import ietf-netconf-acm {
                   prefix nacm;
                   reference
                     "RFC 8341: Network Configuration Access Control
                      Model.";
            }
```

```
          organization "IETF I2NSF Working Group";

          contact
          "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/about/>
           WG List: <mailto:i2nsf@ietf.org>

          Author: Rafael Marin-Lopez
                  <mailto:rafa@um.es>

          Author: Gabriel Lopez-Millan
                  <mailto:gabilm@um.es>

          Author: Fernando Pereniguez-Garcia
                  <mailto:fernando.pereniguez@cud.upct.es>
          ";

          description
              "Data model for IKE-less case in the SDN-base IPsec flow
               protection service.

               Copyright (c) 2019 IETF Trust and the persons
               identified as authors of the code.  All rights reserved.
               Redistribution and use in source and binary forms, with
               or without modification, is permitted pursuant to, and
               subject to the license terms contained in, the
               Simplified BSD License set forth in Section 4.c of the
               IETF Trust's Legal Provisions Relating to IETF Documents
               (https://trustee.ietf.org/license-info).

               This version of this YANG module is part of RFC XXXX;;
               see the RFC itself for full legal notices.

               The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
               'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
               'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
               document are to be interpreted as described in BCP 14
               (RFC 2119) (RFC 8174) when, and only when, they appear
               in all capitals, as shown here.";


          revision "2019-08-05" {
              description "Revision 06";
              reference "RFC XXXX: YANG model for IKE case.";
          }


          container ipsec-ikeless {
              description
```

```
                "Container for configuration of the IKE-less
                 case. The container contains two additional
                 containers: 'spd' and 'sad'. The first allows the
                 Security Controller to configure IPsec policies in
                 the Security Policy Database SPD, and the second
                 allows to configure IPsec Security Associations
                 (IPsec SAs) in the Security Association Database
                 (SAD).";
            reference "RFC 4301.";
            container spd {
                description
                    "Configuration of the Security Policy Database
                     (SPD.)";
                reference "Section 4.4.1.2 in RFC 4301.";

                list spd-entry {
                    key "name";
                    ordered-by user;
                    leaf name {
                        type string;
                        mandatory true;
                        description
                            "SPD entry unique name to identify this
                             entry.";
                    }
                    leaf direction {
                        type ic:ipsec-traffic-direction;
                        description
                            "Inbound traffic or outbound
                             traffic. In the IKE-less case the
                             Security Controller needs to
                             specify the policy direction to be
                             applied in the NSF. In the IKE case
                             this direction does not need to be
                             specified since IKE
                             will determine the direction that
                             IPsec policy will require.";
                    }
                    leaf reqid {
                        type uint64;
                        default 0;
                        description
                            "This value allows to link this
                             IPsec policy with IPsec SAs with the
                             same reqid. It is only required in
                             the IKE-less model since, in the IKE
                             case this link is handled internally
                             by IKE.";
```

```
                     }

                     container ipsec-policy-config {
                         description
                             "This container carries the
                             configuration of a IPsec policy.";
                         uses ic:ipsec-policy-grouping;
                     }
                     description
                         "The SPD is represented as a list of SPD
                          entries, where each SPD entry represents an
                          IPsec policy.";
                 } /*list spd-entry*/
             } /*container spd*/

             container sad {
                 description
                     "Configuration of the IPSec Security Association
                      Database (SAD)";
                 reference "Section 4.4.2.1 in RFC 4301.";
                 list sad-entry {
                     key "name";
                     ordered-by user;
                     leaf name {
                         type string;
                         description
                             "SAD entry unique name to identify this
                              entry.";
                     }
                     leaf reqid {
                         type uint64;
                         default 0;
                         description
                             "This value allows to link this
                              IPsec SA with an IPsec policy with
                              the same reqid.";
                     }

                     container ipsec-sa-config {
                         description
                             "This container allows configuring
                             details of an IPsec SA.";
                         leaf spi {
                             type uint32 { range "0..max"; }
                             mandatory true;
                             description
                                 "Security Parameter Index (SPI)'s
                                  IPsec SA.";
```

```
                          }
                          leaf ext-seq-num {
                              type boolean;
                              default true;
                              description
                                  "True if this IPsec SA is using
                                   extended sequence numbers. True 64
                                   bit counter, FALSE 32 bit.";
                          }
                          leaf seq-number-counter {
                              type uint64;
                              default 0;
                              description
                                   "A 64-bit counter when this IPsec
                                   SA is using Extended Sequence
                                   Number or 32-bit counter when it
                                   is not. It used to generate the
                                   initial Sequence Number field
                                   in ESP headers.";
                          }
                          leaf seq-overflow {
                              type boolean;
                              default false;
                              description
                                  "The flag indicating whether
                                   overflow of the sequence number
                                   counter should prevent transmission
                                   of additional packets on the IPsec
                                   SA (false) and, therefore needs to
                                   be rekeyed, or whether rollover is
                                   permitted (true). If Authenticated
                                   Encryption with Associated Data
                                   (AEAD) is used this flag MUST BE
                                   false.";
                          }
                          leaf anti-replay-window {
                              type uint32;
                              default 32;
                              description
                                  "A 32-bit counter and a bit-map (or
                                   equivalent) used to determine
                                   whether an inbound ESP packet is a
                                   replay. If set to 0 no anti-replay
                                   mechanism is performed.";
                          }
                          container traffic-selector {
                              uses ic:selector-grouping;
                              description
```

```
                                   "The IPsec SA traffic selector.";
                           }
                           leaf protocol-parameters {
                               type ic:ipsec-protocol-parameters;
                               default esp;
                               description
                                   "Security protocol of IPsec SA: Only
                                   ESP so far.";
                           }
                           leaf mode {
                               type ic:ipsec-mode;
                               description
                                   "Tunnel or transport mode.";
                           }
                           container esp-sa {
                               when "../protocol-parameters =
                            'esp'";
                               description
                                   "In case the IPsec SA is
                                    Encapsulation Security Payload
                                    (ESP), it is required to specify
                                    encryption and integrity
                                    algorithms, and key material.";

                               container encryption {
                                   description
                                       "Configuration of encryption or
                                        AEAD algorithm for IPSec
                                        Encapsulation Security Payload
                                        (ESP).";

                                   leaf encryption-algorithm {
                                     type ic:encryption-algorithm-type;
                                     description
                                           "Configuration of ESP
                                            encryption. With AEAD
                                            algorithms, the integrity
                                            node is not used.";
                                   }

                                   leaf key {
                                       nacm:default-deny-all;
                                       type yang:hex-string;
                                       description
                                           "ESP encryption key value.";
                                    }
                                   leaf iv {
                                       nacm:default-deny-all;
```

```
                               type yang:hex-string;
                               description
                                   "ESP encryption IV value.";
                           }
                       }
                       container integrity {
                           description
                               "Configuration of integrity for
                                IPSec Encapsulation Security
                                Payload (ESP). This container
                                allows to configure integrity
                                algorithm when no AEAD
                                algorithms are used, and
                                integrity is required.";
                            leaf integrity-algorithm {
                               type ic:integrity-algorithm-type;
                               description
                                   "Message Authentication Code
                                   (MAC) algorithm to provide
                                   integrity in ESP.";
                           }
                           leaf key {
                               nacm:default-deny-all;
                               type yang:hex-string;
                               description
                                   "ESP integrity key value.";
                           }
                       }
                   } /*container esp-sa*/

                   container sa-lifetime-hard {
                       description
                           "IPsec SA hard lifetime. The action
                           associated is terminate and
                           hold.";
                       uses ic:lifetime;
                   }
                   container sa-lifetime-soft {
                       description
                           "IPSec SA soft lifetime.";
                       uses ic:lifetime;
                       leaf action {
                           type ic:lifetime-action;
                           description
                               "Action lifetime:
                                terminate-clear,
                                terminate-hold or replace.";
                       }
```

```
                            }
                            container tunnel {
                                when "../mode = 'tunnel'";
                                uses ic:tunnel-grouping;
                                description
                                     "Endpoints of the IPsec tunnel.";
                            }
                            container encapsulation-type
                            {
                                uses ic:encap;
                                description
                                    "This container carries
                                     configuration information about
                                     the source and destination ports
                                     which will be used for ESP
                                     encapsulation that ESP packets the
                                     type of encapsulation when NAT
                                     traversal is in place.";
                            }
                        } /*ipsec-sa-config*/

                        container ipsec-sa-state {
                            config false;
                            description
                                "Container describing IPsec SA state
                                data.";
                            container sa-lifetime-current {
                                uses ic:lifetime;
                                description
                                    "SAD lifetime current.";
                            }
                            container replay-stats {
                                description
                                    "State data about the anti-replay
                                     window.";
                                leaf replay-window {
                                    type uint64;
                                    description
                                        "Current state of the replay
                                         window.";
                                }
                                leaf packet-dropped {
                                    type uint64;
                                    description
                                        "Packets detected out of the
                                         replay window and dropped
                                         because they are replay
                                         packets.";
```

```
                              }
                              leaf failed {
                                  type uint32;
                                  description
                                      "Number of packets detected out
                                       of the replay window.";
                              }
                              leaf seq-number-counter {
                                  type uint64;
                                  description
                                      "A 64-bit counter when this
                                       IPsec SA is using Extended
                                       Sequence Number or 32-bit
                                       counter when it is not.
                                       Current value of sequence
                                       number.";
                              }
                          } /* container replay-stats*/
                      } /*ipsec-sa-state*/

                      description
                          "List of SAD entries that conforms the SAD.";
                  } /*list sad-entry*/
              } /*container sad*/
          }/*container ipsec-ikeless*/

          /* Notifications */
          notification sadb-acquire {
              description
                  "An IPsec SA is required. The traffic-selector
                   container contains information about the IP packet
                   that triggers the acquire notification.";
              leaf ipsec-policy-name {
                  type string;
                  mandatory true;
                  description
                      "It contains the SPD entry name (unique) of
                       the IPsec policy that hits the IP packet
                       required IPsec SA. It is assumed the
                       Security Controller will have a copy of the
                       information of this policy so it can
                       extract all the information with this
                       unique identifier. The type of IPsec SA is
                       defined in the policy so the Security
                       Controller can also know the type of IPsec
                       SA that must be generated.";
              }
              container traffic-selector {
```

```
                    description
                        "The IP packet that triggered the acquire
                         and requires an IPsec SA. Specifically it
                         will contain the IP source/mask and IP
                         destination/mask; protocol (udp, tcp,
                         etc...); and source and destination
                         ports.";
                    uses ic:selector-grouping;
               }
          }

          notification sadb-expire {
              description "An IPsec SA expiration (soft or hard).";
              leaf ipsec-sa-name {
                  type string;
                  mandatory true;
                  description
                      "It contains the SAD entry name (unique) of
                       the IPsec SA that has expired.  It is assumed
                       the Security Controller will have a copy of the
                       IPsec SA information (except the cryptographic
                       material and state data) indexed by this name
                       (unique identifier) so it can know all the
                       information (crypto algorithms, etc.) about
                       the IPsec SA that has expired in order to
                       perform a rekey (soft lifetime) or delete it
                       (hard lifetime) with this unique identifier.";
              }
              leaf soft-lifetime-expire {
                  type boolean;
                  default true;
                  description
                      "If this value is true the lifetime expired is
                       soft. If it is false is hard.";
              }
              container lifetime-current {
                  description
                      "IPsec SA current lifetime. If
                       soft-lifetime-expired is true this container is
                       set with the lifetime information about current
                       soft lifetime.";
                  uses ic:lifetime;
              }
          }
          notification sadb-seq-overflow {
              description "Sequence overflow notification.";
              leaf ipsec-sa-name {
                  type string;
```

```
                   mandatory true;
                   description
                       "It contains the SAD entry name (unique) of
                        the IPsec SA that is about to have sequence
                        number overflow and rollover is not permitted.
                        It is assumed the Security Controller will have
                        a copy of the IPsec SA information (except the
                        cryptographic material and state data) indexed
                        by this name (unique identifier) so the it can
                        know all the information (crypto algorithms,
                        etc.) about the IPsec SA that has expired in
                        order to perform a rekey of the IPsec SA.";
               }
           }
           notification sadb-bad-spi {
               description
                   "Notify when the NSF receives a packet with an
                    incorrect SPI (i.e. not present in the SAD).";
               leaf spi {
                   type uint32 { range "0..max"; }
                   mandatory true;
                   description
                       "SPI number contained in the erroneous IPsec
                        packet.";
               }
           }
       }/*module ietf-ipsec*/

       <CODE ENDS>
```

Appendix D. Example of IKE case, tunnel mode (gateway-to-gateway) with X.509 certificate authentication.

This example shows a XML configuration file sent by the Security Controller to establish a IPsec Security Association between two NSFs in tunnel mode (gateway-to-gateway) with ESP, and authentication based on X.509 certificates using IKEv2.

```
                        Security Controller
                              |
                 /---- Southbound interface -----\
                /                                 \
               /                                   \
              /                                     \
             /                                       \
            nsf_h1                                  nsf_h2
    h1---- (:1/:100)===== IPsec_ESP_Tunnel_mode =====(:200/:1)-------h2
    2001:DB8:1:/64       (2001:DB8:123:/64)          2001:DB8:2:/64
```

Figure 7: IKE case, tunnel mode , X.509 certicate authentication.


```
<ipsec-ike xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ike"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <pad>
    <pad-entry>
      <name>nsf_h1_pad</name>
      <ipv6-address>2001:DB8:123::100</ipv6-address>
      <peer-authentication>
         <auth-method>digital-signature</auth-method>
         <digital-signature>
            <cert-data>base64encodedvalue==</cert-data>
            <private-key>base64encodedvalue==</private-key>
            <ca-data>base64encodedvalue==</ca-data>
         </digital-signature>
      </peer-authentication>
    </pad-entry>
    <pad-entry>
      <name>nsf_h2_pad</name>
      <ipv6-address>2001:DB8:123::200</ipv6-address>
      <auth-protocol>ikev2</auth-protocol>
      <peer-authentication>
        <auth-method>digital-signature</auth-method>
        <digital-signature>
          <!-- RSA Digital Signature -->
          <ds-algorithm>1</ds-algorithm>
          <cert-data>base64encodedvalue==</cert-data>
          <ca-data>base64encodedvalue==</ca-data>
        </digital-signature>
      </peer-authentication>
    </pad-entry>
  </pad>
  <conn-entry>
     <name>nsf_h1-nsf_h2</name>
     <autostartup>start</autostartup>
     <version>ikev2</version>
```

```
        <initial-contact>false</initial-contact>
        <fragmentation>true</fragmentation>
        <ike-sa-lifetime-soft>
           <rekey-time>60</rekey-time>
           <reauth-time>120</reauth-time>
        </ike-sa-lifetime-soft>
        <ike-sa-lifetime-hard>
           <over-time>3600</over-time>
        </ike-sa-lifetime-hard>
        <authalg>7</authalg>
        <!--AUTH_HMAC_SHA1_160-->
        <encalg>3</encalg>
        <!--ENCR_3DES -->
        <dh-group>18</dh-group>
        <!--8192-bit MODP Group-->
        <half-open-ike-sa-timer>30</half-open-ike-sa-timer>
        <half-open-ike-sa-cookie-threshold>
           15
        </half-open-ike-sa-cookie-threshold>
        <local>
            <local-pad-entry-name>nsf_h1_pad</local-pad-entry-name>
        </local>
        <remote>
            <remote-pad-entry-name>nsf_h2_pad</remote-pad-entry-name>
        </remote>
        <spd>
          <spd-entry>
              <name>nsf_h1-nsf_h2</name>
              <ipsec-policy-config>
                <anti-replay-window>32</anti-replay-window>
                <traffic-selector>
                   <local-subnet>2001:DB8:1::0/64</local-subnet>
                   <remote-subnet>2001:DB8:2::0/64</remote-subnet>
                   <inner-protocol>any</inner-protocol>
                   <local-ports>
                     <start>0</start>
                     <end>0</end>
                   </local-ports>
                   <remote-ports>
                     <start>0</start>
                     <end>0</end>
                   </remote-ports>
                </traffic-selector>
                <processing-info>
                   <action>protect</action>
                   <ipsec-sa-cfg>
                      <pfp-flag>false</pfp-flag>
                      <ext-seq-num>true</ext-seq-num>
```

```
                     <seq-overflow>false</seq-overflow>
                     <stateful-frag-check>false</stateful-frag-check>
                     <mode>tunnel</mode>
                     <protocol-parameters>esp</protocol-parameters>
                     <esp-algorithms>
                        <!-- AUTH_HMAC_SHA1_96 -->
                        <integrity>2</integrity>
                        <!-- ENCR_AES_CBC -->
                        <encryption>12</encryption>
                        <tfc-pad>false</tfc-pad>
                     </esp-algorithms>
                     <tunnel>
                        <local>2001:DB8:123::100</local>
                        <remote>2001:DB8:123::200</remote>
                        <df-bit>clear</df-bit>
                        <bypass-dscp>true</bypass-dscp>
                        <ecn>false</ecn>
                    </tunnel>
                  </ipsec-sa-cfg>
               </processing-info>
             </ipsec-policy-config>
          </spd-entry>
        </spd>
        <child-sa-info>
           <!--8192-bit MODP Group -->
           <pfs-groups>18</pfs-groups>
           <child-sa-lifetime-soft>
              <bytes>1000000</bytes>
              <packets>1000</packets>
              <time>30</time>
              <idle>60</idle>
              <action>replace</action>
           </child-sa-lifetime-soft>
           <child-sa-lifetime-hard>
              <bytes>2000000</bytes>
              <packets>2000</packets>
              <time>60</time>
              <idle>120</idle>
           </child-sa-lifetime-hard>
        </child-sa-info>
      </conn-entry>
   </ipsec-ike>
```

Appendix E. Example of IKE-less case, transport mode (host-to-host).

This example shows a XML configuration file sent by the Security Controller to establish a IPsec Security association between two NSFs in transport mode (host-to-host) with ESP.

```
                          Security Controller
                                    |
                     /---- Southbound interface -----\
                    /                                 \
                   /                                   \
                  /                                     \
                 /                                       \
              nsf_h1                                   nsf_h2
              (:100)===== IPsec_ESP_Transport_mode =====(:200)
                            (2001:DB8:123:/64)
```

Figure 8: IKE-less case, transport mode.


```
<ipsec-ikeless
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <spd>
    <spd-entry>
        <name>
           in/trans/2001:DB8:123::200/2001:DB8:123::100
        </name>
        <direction>inbound</direction>
        <reqid>1</reqid>
        <ipsec-policy-config>
           <traffic-selector>
             <local-subnet>2001:DB8:123::200/128</local-subnet>
             <remote-subnet>2001:DB8:123::100/128</remote-subnet>
             <inner-protocol>any</inner-protocol>
                <local-ports>
                   <start>0</start>
                   <end>0</end>
                </local-ports>
                <remote-ports>
                   <start>0</start>
                   <end>0</end>
                 </remote-ports>
           </traffic-selector>
           <processing-info>
              <action>protect</action>
              <ipsec-sa-cfg>
                <ext-seq-num>true</ext-seq-num>
```

```
                  <seq-overflow>true</seq-overflow>
                  <mode>transport</mode>
                  <protocol-parameters>esp</protocol-parameters>
                  <esp-algorithms>
                     <!--AUTH_HMAC_SHA1_96-->
                     <integrity>2</integrity>
                     <!--ENCR_AES_CBC -->
                     <encryption>12</encryption>
                  </esp-algorithms>
                </ipsec-sa-cfg>
              </processing-info>
            </ipsec-policy-config>
          </spd-entry>
          <spd-entry>
            <name>out/trans/2001:DB8:123::100/2001:DB8:123::200</name>
            <direction>outbound</direction>
            <reqid>1</reqid>
            <ipsec-policy-config>
              <traffic-selector>
                <local-subnet>2001:DB8:123::100/128</local-subnet>
                <remote-subnet>2001:DB8:123::200/128</remote-subnet>
                <inner-protocol>any</inner-protocol>
                <local-ports>
                  <start>0</start>
                  <end>0</end>
                </local-ports>
                <remote-ports>
                  <start>0</start>
                  <end>0</end>
                </remote-ports>
              </traffic-selector>
              <processing-info>
                <action>protect</action>
                <ipsec-sa-cfg>
                  <ext-seq-num>true</ext-seq-num>
                  <seq-overflow>true</seq-overflow>
                  <mode>transport</mode>
                  <protocol-parameters>esp</protocol-parameters>
                  <esp-algorithms>
                    <!-- AUTH_HMAC_SHA1_96 -->
                    <integrity>2</integrity>
                    <!-- ENCR_AES_CBC -->
                    <encryption>12</encryption>
                  </esp-algorithms>
                 </ipsec-sa-cfg>
               </processing-info>
             </ipsec-policy-config>
          </spd-entry>
```

```
        </spd>
        <sad>
          <sad-entry>
            <name>out/trans/2001:DB8:123::100/2001:DB8:123::200</name>
            <reqid>1</reqid>
            <ipsec-sa-config>
               <spi>34501</spi>
               <ext-seq-num>true</ext-seq-num>
               <seq-number-counter>100</seq-number-counter>
               <seq-overflow>true</seq-overflow>
               <anti-replay-window>32</anti-replay-window>
               <traffic-selector>
                 <local-subnet>2001:DB8:123::100/128</local-subnet>
                 <remote-subnet>2001:DB8:123::200/128</remote-subnet>
                    <inner-protocol>any</inner-protocol>
                    <local-ports>
                       <start>0</start>
                       <end>0</end>
                    </local-ports>
                    <remote-ports>
                       <start>0</start>
                       <end>0</end>
                    </remote-ports>
                </traffic-selector>
                <protocol-parameters>esp</protocol-parameters>
                <mode>transport</mode>
                <esp-sa>
                  <encryption>
                     <!-- //ENCR_AES_CBC -->
                     <encryption-algorithm>12</encryption-algorithm>
                     <key>01:23:45:67:89:AB:CE:DF</key>
                     <iv>01:23:45:67:89:AB:CE:DF</iv>
                  </encryption>
                  <integrity>
                     <!-- //AUTH_HMAC_SHA1_96 -->
                     <integrity-algorithm>2</integrity-algorithm>
                     <key>01:23:45:67:89:AB:CE:DF</key>
                  </integrity>
                </esp-sa>
            </ipsec-sa-config>
          </sad-entry>
          <sad-entry>
             <name>in/trans/2001:DB8:123::200/2001:DB8:123::100</name>
             <reqid>1</reqid>
             <ipsec-sa-config>
                 <spi>34502</spi>
                 <ext-seq-num>true</ext-seq-num>
                 <seq-number-counter>100</seq-number-counter>
```

```
                 <seq-overflow>true</seq-overflow>
                 <anti-replay-window>32</anti-replay-window>
                 <traffic-selector>
                    <local-subnet>2001:DB8:123::200/128</local-subnet>
                    <remote-subnet>2001:DB8:123::100/128</remote-subnet>
                    <inner-protocol>any</inner-protocol>
                    <local-ports>
                       <start>0</start>
                       <end>0</end>
                    </local-ports>
                    <remote-ports>
                       <start>0</start>
                       <end>0</end>
                    </remote-ports>
                 </traffic-selector>
                 <protocol-parameters>esp</protocol-parameters>
                 <mode>transport</mode>
                 <esp-sa>
                    <encryption>
                       <!-- //ENCR_AES_CBC -->
                       <encryption-algorithm>12</encryption-algorithm>
                       <key>01:23:45:67:89:AB:CE:DF</key>
                       <iv>01:23:45:67:89:AB:CE:DF</iv>
                    </encryption>
                    <integrity>
                       <!-- //AUTH_HMAC_SHA1_96 -->
                       <integrity-algorithm>2</integrity-algorithm>
                       <key>01:23:45:67:89:AB:CE:DF</key>
                    </integrity>
                  </esp-sa>
                  <sa-lifetime-hard>
                     <bytes>2000000</bytes>
                     <packets>2000</packets>
                     <time>60</time>
                     <idle>120</idle>
                  </sa-lifetime-hard>
                  <sa-lifetime-soft>
                     <bytes>1000000</bytes>
                     <packets>1000</packets>
                     <time>30</time>
                     <idle>60</idle>
                     <action>replace</action>
                  </sa-lifetime-soft>
            </ipsec-sa-config>
          </sad-entry>
       </sad>
   </ipsec-ikeless>
```

Appendix F. Examples of notifications.

Below we show several XML files that represent different types of notifications defined in the IKE-less YANG model, which are sent by the NSF to the Security Controller. The notifications happen in the IKE-less case.


```
<sadb-expire xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
<ipsec-sa-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
</ipsec-sa-name>
    <soft-lifetime-expire>true</soft-lifetime-expire>
       <lifetime-current>
          <bytes>1000000</bytes>
          <packets>1000</packets>
          <time>30</time>
          <idle>60</idle>
       </lifetime-current>
</sadb-expire>
```


Figure 9: Example of sadb-expire notification.


```
<sadb-acquire xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
    <ipsec-policy-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
    </ipsec-policy-name>
    <traffic-selector>
        <local-subnet>2001:DB8:123::200/128</local-subnet>
        <remote-subnet>2001:DB8:123::100/128</remote-subnet>
        <inner-protocol>any</inner-protocol>
         <local-ports>
              <start>0</start>
              <end>0</end>
         </local-ports>
         <remote-ports>
              <start>0</start>
              <end>0</end>
         </remote-ports>
    </traffic-selector>
</sadb-acquire>
```


Figure 10: Example of sadb-acquire notification.

```
   <sadb-seq-overflow
       xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
         <ipsec-sa-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
         </ipsec-sa-name>
   </sadb-seq-overflow>
```


Figure 11: Example of sadb-seq-overflow notification.


```
   <sadb-bad-spi
            xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
           <spi>666</spi>
   </sadb-bad-spi>
```


Figure 12: Example of sadb-bad-spi notification.

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
EMail: rafa@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 04
EMail: gabilm@um.es

Fernando Pereniguez-Garcia
University Defense Center
Spanish Air Force Academy, MDE-UPCT
San Javier (Murcia) 30720
Spain

Phone: +34 968 18 99 46
EMail: fernando.pereniguez@cud.upct.es

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 18, 2018

J. Jeong
E. Kim
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei

November 14, 2017

I2NSF Consumer-Facing Interface YANG Data Model
draft-jeong-i2nsf-consumer-facing-interface-dm-05

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 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.

```
module: ietf-i2nsf-cf-interface
```

```
    +--rw ietf-i2nsf-consumer-facing-interface
       +--rw policy
       |  +--rw rule* [rule-id]
       |  |  +--rw rule-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?
       |  |  |  +-- start-time         yang:date-and-time
       |  |  |  +-- end-time           yang:date-and-time
       |  |  +--rw event-map-group?    -> /ietf-i2nsf-consumer-facing-interface/
       |  |  |                            threat-feed/threat-feed/
       |  |  |                            threat-feed-id
       |  |  +--rw enable?             boolean
       |  +--rw condition* [condition-id]
       |  |  +--rw condition-id       string
       |  |  +--rw source?            -> /ietf-i2nsf-consumer-facing-interface/
       |  |  |                            threat-feed/threat-feed/
       |  |  |                            threat-feed-id
       |  |  +--rw destination?       -> /ietf-i2nsf-consumer-facing-interface/
       |  |  |                            threat-feed/threat-feed/
       |  |  |                            custom-list-id
       |  |  +--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 authentication-method    string
       |  +--rw policy-tenant* [policy-tenant-id]
       |  |  +--rw policy-tenant-id*   uint16
       |  |  +--rw name                string
       |  |  +--rw date                yang:date-and-time
```

```
       |  |  +--rw domain              string
       |  +--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-mgmt-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         string
       |     +--rw mutual-authentication         boolean
       |     +--rw token-server                  string
       |     +--rw certificate-server            string
       |     +--rw single-sing-on-server         string
       +--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?     string
       |  |  +--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?         string
       |  |  +--rw meta-data-server?   string
       |  |  +--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?         string
       |  |  +--rw meta-data-server?   string
       |  |  +--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?             string
     |  |  +--rw meta-data-server?       string
     |  |  +--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?          string
     |     +--rw meta-data-server?    string
     |     +--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?      string
     |  |  +--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?        string
     |  |  +--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?                string
          |  +--rw nsf-access-parameters?      string
          |  +--rw nsf-access-credentials?     string
          |  +--rw collection-interval?        uint16
          |  +--rw collection-method?          enumeration
          |  +--rw heartbeat-interval?         uint16
          |  +--rw qos-marking?                uint8
          +--rw telemetry-destination* [telemetry-destination-id]
             +--rw telemetry-destination-id      uint16
             +--rw name?                         string
             +--rw date?                         yang:date-and-time
             +--rw collector-state?              string
             +--rw collector-credentials?        string
             +--rw collector-source?             string
             +--rw data-encoding?                string
             +--rw data-transport?               string
```

Figure 1: Generic Data Model for cf Interface

5. YANG Data Model 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 "ietf-i2nsf-cf-interface.yang"
module ietf-i2nsf-cf-interface {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-cf-interface";
  prefix
    cf-interface;

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

     description
       "This module defines a YANG data module for consumer-facing
        interface to security controller.";

     revision "2017-11-14"{
       description "Fifth revision";
       reference
         "draft-kumar-i2nsf-client-facing-interface-im-04";
     }

     //Groupings
     container ietf-i2nsf-consumer-facing-interface {
     description
     "grouping Policy";
     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";
             }

           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.";
             }
              list time-information {
               key "time-information-id";
               leaf time-information-id{
                 type string;
                 description
                 "this is a time information id.";
```

```
               }
               leaf start-time {
                 type yang:date-and-time;
                 description
                   "start time information.";
               }
               leaf end-time {
                 type yang:date-and-time;
                 description
                 "end time information.";
               }
                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 string;
               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 enumeration{
                    enum one-direction{
                        value 0;
                    description
                    "one direction traffic.";
                    }
                    enum both-direction{
                         value 1;
                         description
                         "both direction traffic.";
                    }
                }
                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 enumeration{
                  enum permit{
                  value 0;
                  description
                  "permit.";
                  }
                  enum deny{
                  value 1;
                  description
                  "deny.";
                  }
                  enum rate-limit{
                  value 2;
                  description
```

```
                  "rate-limit.";
                  }
                  enum traffic-class{
                  value 3;
                  description
                  "traffic-class.";
                  }
                  enum authenticate-session{
                  value 4;
                  description
                  "authenticate-session";
                  }
                  enum ips{
                  value 5;
                  description
                  "ips.";
                  }
                  enum app-firewall{
                  value 6;
                  description
                  "app-firewall.";
                  }
              }
              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 enumeration{
                  enum log{
                  value 0;
                  description
                  "log.";
                  }
                  enum syslog{
                  value 1;
                  description
                  "syslog.";
                  }
                  enum session-log{
                  value 2;
                  description
                  "session-log.";
                  }
              }
```

```
                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.";
             }

             leaf authentication-method {
               type string;
               mandatory true;
               description
                 "The description of authentication method;
                 token-based, password, certificate,
                 single-sign-on";
             }
           }

           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 string;
               mandatory true;
```

```
              description
              "This field identifies the domain to which this
              tenant belongs. This should be reference to a
              'Policy-Domain' 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-mgmt-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 string;
              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 string;
               mandatory true;
               description
                 "The token-server information if the
                 authentication method is token-based";
             }

             leaf certificate-server {
               type string;
               mandatory true;
               description
                 "The certificate-server information if
                 the authentication method is certificate-based";
             }

             leaf single-sing-on-server {
               type string;
               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 string;
              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 string;
               description
                 "This describes the group type; User-tag,
                 User-name or IP-address.";
             }

             leaf meta-data-server {
               type string;
               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 string;
              description
                "This describes the group type; device-tag,
                device-name or IP-address.";
            }

            leaf meta-data-server {
              type string;
              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 string;
               description
                 "This identifies the group type;
                 application-tag, application-name or
                 IP-address.";
             }

             leaf meta-data-server {
               type string;
               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 string;
               description
                 "This identifies the group type;
                 location-tag, location-name or
                 IP-address.";
             }

             leaf meta-data-server {
               type string;
               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 string;
              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 string;
              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 string;
              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-access-parameters {
              type string;
              description
                "This field contains information such as
                IP address and protocol (UDP or TCP) port
```

```
              number of the NSF providing telemetry data.";
          }

          leaf nsf-access-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 uint8;
              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-state {
              type string;
              description
                "This describes collector state information.";
            }
            leaf collector-credentials {
              type string;
              description
                "iThis field contains the username and
               password for the collector.";
            }
```

```
            leaf collector-source {
              type string;
              description
                "This field contains information such as
                IP address and protocol (UDP or TCP) port
                number for the collector's destination.";
            }

            leaf data-encoding {
              type string;
              description
              "This field contains the telemetry data encoding
              in the form of schema.";
            }

            leaf data-transport {
              type string;
              description
              "This field contains streaming telemetry data
              protocols. This could be gRPC, protocol
              buffer over UDP, etc.";
            }
        }
      }
      }
    }
  }
}
<CODE ENDS>
```

Figure 2: YANG for cf_interface

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).

8. Contributors

I2NSF is a group effort. This document has greatly benefited from inputs by Mahdi F. Dachmehchi, Jinyong Tim Kim, and Daeyoung Hyun. I2NSF has a number of contributing authors. The following 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-04

The following changes have been made from draft-jeong-i2nsf-consumer-facing-interface-dm-04:

- o Sections 4 and 5 have been revised to produce a data tree model and YANG data model according to the information model and Event-Condition-Action (ECA) based policy generation 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 An example XML format output has been modified 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: ietf-i2nsf-cf-interface-voip
    +--rw ietf-i2nsf-consumer-facing-interface
       +--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?
          |   |     +-- start-time         yang:date-and-time
          |   |     +-- end-time           yang:date-and-time;
          |   |  +--rw event-map-group?      -> /ietf-i2nsf-consumer-facing-inte
rface/
          |   |  |                              threat-prevention/threat-feed/
          |   |  |                              threat-feed-id
          |   |  +--rw enable?               boolean
          |   +--rw condition* [condition-id]
          |   |  +--rw condition-id          string
          |   |  +--rw source-caller?        -> /ietf-i2nsf-consumer-facing-inte
rface/
          |   |  |                              threat-prevention/custom-list/
          |   |  |                              custom-list-id
          |   |  +--rw destination-callee?   -> /ietf-i2nsf-consumer-facing-inte
rface/
          |   |  |                              end-group/user-group/
          |   |  |                              user-group-id
          |   |  +--rw match?                boolean
          |   |  +--rw match-direction?      enum
          |   |  +--rw exception?            string
          |   +--rw action* [action-id]
          |   |  +--rw action-id             string
          |   |  +--rw name?                 string
          |   |  +--rw date?                 yang:date-and-time
          |   |  +--rw primary-action?       enum
          |   |  +--rw secondary-action?     enum
          |   +--rw precedence?              uint8
          +--rw owner?                       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"

  container ietf-i2nsf-consumer-facing-interface {
    description
    "grouping Policy-VoIP";
    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-list 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
                 .";
             }
              list time-information {
               key "time-information-id";
                 leaf start-time {
                    type yang:date-and-time;
                 description
                   "start time information.";
                 }
                 leaf end-time {
                    type yang:date-and-time;
                    description
                      "end time information.";
                 }
                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 "/ietf-i2nsf-consumer-facing-interface/
                 threat-prevention/threat-feed/threat-feed-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.
                There can be a negation rule, such that an
                action to be applied when there is no event";
            }
          }
          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 "/ietf-i2nsf-consumer-facing-interface/
                threat-prevention/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 destination-callee {
              type leafref {
                path "/ietf-i2nsf-consumer-facing-interface/
                end-group/user-group/user-group-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 uint8;
               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 action {
             key "owner-id";
             leaflist 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.";
             }
         }
       }
     }
   }


   <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: May 3, 2018

J. Kim
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
Q. Lin
Huawei

October 30, 2017

I2NSF Network Security Functions-Facing Interface YANG Data Model
draft-kim-i2nsf-nsf-facing-interface-data-model-04

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

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 caluse for generic network security function policy
- o configuration of condition caluse for generic network security function policy
- o configuration of action caluse 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. Objectives

4.1. I2NSF Security Policy Rule

This shows a identification of policy for generic network security functions. These objects are defined as policy information and rule information. This includes ECA Policy Rule, Event Clause Objects, Condition Clause Objects, and Action Clause Objects, Resolution Strategy, Default Action.

4.2. Event Caluse

This shows a event caluse 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. 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.

4.3. Condition Caluse

This shows a condition caluse 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 user security event, device security event, system security event, and time security event. 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.

4.4. Action Caluse

This shows a action caluse 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. These objects are defined as ingress action, egress action, and apply profile action. These objects can be extended according to specific vendor action features.

5. Data Model Structure

This section shows an following mapped features of a data model structure tree of generic network security functions, as 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 identification of network security policy has the following structure:

```
module: ietf-i2nsf-nsf-facing-interface
+--rw generic-nsf
|  +--rw i2nsf-security-policy* [policy-name]
|     +--rw policy-name            string
|     +--rw time-zone
|     |  +--rw start-time?   yang:date-and-time
|     |  +--rw end-time?     yang:date-and-time
|     +--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 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-nsf-facing-interface
+--rw generic-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-nsf-facing-interface
+--rw generic-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 (condition-type)?
  |        +--:(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-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-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
  |        +--:(packet-payload-condition)
```

```
   |         |  +--rw packet-payload-manual?            string
   |         |  +--rw pkt-payload-content*              string
   |         +--:(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
   |         +--:(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
   |         +--:(context-condition)
   |         |  +--rw context-manual?                   string
   |         +--:(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-nsf-facing-interface
+--rw generic-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 (action-type)?
         +--:(ingress-action)
         |  +--rw ingress-manual?                    string
         |  +--rw ingress-action-type?               ingress-action
         +--:(egress-action)
         |  +--rw egress-manual?                     string
         |  +--rw egress-action-type?                egress-action
         +--:(apply-profile)
            +--rw profile-manual?                    string
            +--rw (apply-profile-action-type)?
               +--:(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
               +--:(attack-mitigation-control)
                  +--rw (attack-mitigation-control-type)?
                     +--:(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
                        +--:(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-nsf-facing-interface@2017-10-30.yang"

module ietf-i2nsf-nsf-facing-interface {
  yang-version 1.1;
```

```
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-facing-interface";
  prefix
    nsf-facing-interface;

  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 "2017-10-30"{
    description "The third 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).";
      }

 }
```

```
 container generic-nsf {
  description
    "Configuration for Generic Network Security Functions.";

  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.";
      }
      container time-zone {
        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";
        }
      }


      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 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.";

          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-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-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-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.";
              }
            }
          }

          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-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.";
            }
```

```
          }
          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";
            }

            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.";
              }
            }
          }
          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 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.";
                    }
                  }
                }
              }

            }
            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-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.";


      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";
          }
          leaf ingress-action-type {
            type ingress-action;
            description
              "Ingress action type: permit, deny, and 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";
            }
            leaf egress-action-type {
              type egress-action;
              description
                "Egress-action-type: invoke-signaling,
                 tunnel-encapsulation, and forwarding.";
            }
          }
          case apply-profile {
            leaf profile-manual {
              type string;
              description
                "This is manual for apply profile action.
                Vendors can write instructions for apply
                profile action that vendor made";
            }

            choice apply-profile-action-type {
              description
                "Advanced action types: Content Security Control
                 and Attack Mitigation Control.";

              case 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.";
                }
              }
            }

            case attack-mitigation-control {
              description
                "This category of security capabilities is
                 specially used to detect and mitigate various
                 types of network attacks.";

              choice attack-mitigation-control-type {
                description
                  "Attack-mitigation types: DDoS-attack and
                   Single-packet attack.";

                case 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.";
                }
              }
            }
          }
        }

        case 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 [i2nsf-framework].

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 Jung-Soo Park (ETRI)
- 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.

10.2. 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-08 (work in progress), October 2017.

[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., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-08 (work in progress), July 2017.

[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. draft-kim-i2nsf-nsf-facing-interface-data-model-03

The following changes are made from draft-kim-i2nsf-nsf-facing-interface-data-model-03:

1. Event/Condition/Action Policies are changed to Event/Condition/Action Clauses.

2. Resolution Strategy mechanism is added 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.

3. Default Action mechanism is added to specify a predefined action when no other alternative action was matched by the currently executing I2NSF Policy Rule.

4. Introduction stating is added that the data model structure can be mapped to draft-ietf-i2nsf-capability.

5. Identities are added for combining the overlaped attributes as one "Identity" so that only one "Identity" is appearing.

6. Aggregations for Event, Condition, and Action Object are added for reusing the objects.

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

EMail: linqiushi@huawei.com

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 18, 2019

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei
J. Jeong
Sungkyunkwan University
July 17, 2018

Information Model for Consumer-Facing Interface to Security Controller
draft-kumar-i2nsf-client-facing-interface-im-07

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. Figure 1 shows a high-level abstraction of Consumer-Facing Interface. 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.

```
                           +-----------------+       +-----------------+
                           |                 |       |                 |
                           | Consumer-Facing +------>+ Consumer Facing |
                           |    Interface    |       |    Interface    |
                           |Information Model|       |    Data Model   |
                           +--------+--------+       +-----------------+
                                    ^
                                    |
                                    |
                      +-------------+-------------+
                      |                           |
                      |       Policy-general      |
                      |                           |
                      +-------------+-------------+
                                    ^
                                    |
         +------------+-------------+------------+--------------+
         |            |             |            |              |
   +-----+----+  +----+-----+  +----+----+  +----+----+  +------+-----+
   |          |  |          |  |         |  |          |  |            |
   |  Multi   |  | Endpoint |  |  Policy |  | Threat   |  | Telemetry  |
   | tenancy  |  |  groups  |  |         |  |  feed    |  |    data    |
   +----------+  +----------+  +----+----+  +----------+  +------------+
                                    ^
                                    |
                                    |
                             +------+------+
                             |             |
                             |     Rule    |
                             |             |
                             +------+------+
                                    ^
                                    |
                   +----------------+----------------+
                   |                |                |
            +------+------+  +------+------+  +------+------+
            |             |  |             |  |             |
            |    Event    |  |  Condition  |  |    Action   |
            |             |  |             |  |             |
            +-------------+  +-------------+  +-------------+
```

Figure 1: Diagram for High-level Abstraction of Consumer-Facing Interface

2. Conventions used in the 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.

The condition object is made of condition clauses. Each condition clause consists of three tuples; variable, operator and value.

The variable and value can be source and destination IP address, for example, and they have logical operator in between to check whether they match the condition criteria set by a security admin. For Example: If condition A AND B is true: THEN execute actions ENDIF where A denotes a destination address, and B denotes a blacklisted IP address. The operator AND is the logical AND operation.

```
                                                     1..n
                                                     +-----------------+
                                                     |                 |
                                       +------------>+  Policy rule    |
                                       |             |                 |
                              1..n     |             +-----------------+
                              +--------+--------+
                              |                 |
                              +Condition clause +
                              |                 |
                              +--------+--------+
                                   ^   ^   ^
                                   |   |   |
                    +--------------+   |   +--------------+
           1..n     |         1..n     |          1..n    |
           +--------+-------+ +--------+--------+ +-------+-------+
           |                | |                 | |               |
           |    Variable    | |    Operator     | |     Value     |
           |                | |                 | |               |
           +----------------+ +-----------------+ +---------------+
```

Figure 2: Condition Clause Diagram

The semantics used in a condition clause is also used in the clauses in the Event-submodel and Action sub-model.

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", "DROP-CONNECTION", "AUTHENTICATE-CONNECTION", "MIRROR", "REDIRECT", "NETFLOW", "COUNT", "ENCRYPT", "DECRYPT", "THROTTLE", "MARK", or "INSTANTIATE-NSF".

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. Below diagram shows an example of multi-tenancy in an Enterprise domain.

```
                                +--------------------+
       (Multi-Tenancy)          |       Domain       |
                                |(e.g., Enterprise)  |
                                +---------+----------+
                                          ^
                                          |
                     +--------------------+--------------------+
                     |                    |                    |
            +--------+-------+  +---------+--------+  +--------+--------+
            |  Department 1  |  |   Department 2   |  |  Department n   |
            +--------+-------+  +---------+--------+  +--------+--------+
                     ^                    ^                    ^
                     |                    |                    |
            +--------+--------+  +--------+--------+  +--------+--------+
            | Sub-domain 1..n |  | Sub-domain 1..n |  | Sub-domain 1..n |
            +--------+--------+  +--------+--------+  +--------+--------+
                     ^                    ^                    ^
                     |                    |                    |
            +--------+--------+  +--------+--------+  +--------+--------+
            |   Tenant 1..n   |  |   Tenant 1..n   |  |   Tenant 1..n   |
            +-----------------+  +-----------------+  +-----------------+
```

Figure 3: Multi-tenancy Diagram

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.

```
                          +-------------------+
                          |  Endpoint Group   |
                          +---------+---------+
                                    ^
                                    |
               +------------+-------+-----+---------------+
       1..n    |      1..n  |     1..n    |       1..n    |
        +-----+----+  +----+---+  +------+------+  +-----+----+
        |   User   |  | Device |  | Application |  | Location |
        +----------+  +--------+  +-------------+  +----------+
```

Figure 4: Endpoint Group Diagram

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.

```
                          +---------------------+
                          |  Threat Prevention  |
                          +----------+----------+
                                     ^
                                     |
               +---------------------+---------------------+
 1..n          |          1..n       |          1..n       |
  +----------+---------+ +---------+---------+ +----------+---------+
  |     Threat feed    | |    Custom list    | | Malware scan group |
  +--------------------+ +-------------------+ +--------------------+
```

Figure 5: Threat Prevention Diagram

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. Role-Based Acess Control (RBAC)

Role-Based Access Control (RBAC) provides a powerful and centralized control within a network. It is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC, such as role-permissions, user-role and role-role relationships, make it simple to perform user assignments.

```
+--------------+
|              |
|    User 1    + (has many)
|              |\
+--------------+ \     +---------------+           +-------------+
       .          \    |               | (has many) |             |
       .           --->+ List of roles +----------->+ Permissions |
+--------------+  /    |               |            |             |
|              | /     +---------------+            +-------------+
|    User n    +/
|              | (has many)
+--------------+
```

Figure 6: RBAC Diagram

As shown in Figure 6, a role represents a collection of permissions (e.g., accessing a file server or other particular resources). A role may be assigned to one or multiple users. Both roles and permissions can be organized in a hirarchy. A role may consists of other roles and permissions.

Following are the steps required to build RBAC.

1. Defining roles and permissions.

2. Establishing relations among roles and permissions.

3. Defining users.

4. Associating rules with roles and permissions.

5. assigning roles to users.

9. 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.

10. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

11. 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).

12. 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:

- Eunsoo Kim (Sungkyunkwan University)
- Seungjin Lee (Sungkyunkwan University)
- Hyoungshick Kim (Sungkyunkwan University)

13. 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-02 (work in progress), July 2018.

[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-05 (work in progress), May 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-06 (work in progress), July 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-06

The following changes have been made from draft-kumar-i2nsf-client-facing-interface-im-06:

- o In Section 1, Figure 1 is added to show a diagram for a high-level abstraction of Consumer-Facing Interface.

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

I2NSF
Internet-Draft
Intended status: Standard Track
Expires: January 5, 2018

L. Xia
J. Strassner
Huawei
C. Basile
PoliTO
D. Lopez
TID

July 3, 2017

Information Model of NSFs Capabilities
draft-xibassnez-i2nsf-capability-02.txt

Abstract

This document defines the concept of an NSF (Network Security Function) Capability, as well as its information model. Capabilities are a set of features that are available from a managed entity, and are represented as data that unambiguously characterizes an NSF. Capabilities enable management entities to determine the set offer features from available NSFs that will be used, and simplify the management of NSFs.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

1. Introduction

The rapid development of virtualized systems requires advanced security protection in various scenarios. Examples include network devices in an enterprise network, User Equipment in a mobile network, devices in the Internet of Things, or residential access users [I-D.draft-ietf-i2nsf-problem-and-use-cases].

NSFs produced by multiple security vendors provide various security Capabilities to customers. Multiple NSFs can be combined together to provide security services over the given network traffic, regardless of whether the NSFs are implemented as physical or virtual functions.

Security Capabilities describe the set of network security-related features that are available to use for security policy enforcement purposes. Security Capabilities are independent of the actual security control mechanisms that will implement them. Every NSF registers the set of Capabilities it offers. Security Capabilities are a market enabler, providing a way to define customized security protection by unambiguously describing the security features offered by a given NSF. Moreover, Security Capabilities enable security functionality to be described in a vendor-neutral manner. That is, it is not required to refer to a specific product when designing the network; rather, the functionality characterized by their Capabilities are considered.

According to [I-D.draft-ietf-i2nsf-framework], there are two types of I2NSF interfaces available for security policy provisioning:

- o Interface between I2NSF users and applications, and a security controller (Consumer-Facing Interface): this is a service-oriented interface that provides a communication channel between consumers of NSF data and services and the network operator's security controller. This enables security information to be exchanged between various applications (e.g., OpenStack, or various BSS/OSS components) and the security controller. The design goal of the Consumer-Facing Interface is to decouple the specification of security services from their implementation.

- o Interface between NSFs (e.g., firewall, intrusion prevention, or anti-virus) and the security controller (NSF-Facing Interface): The NSF-Facing Interface is used to decouple the security management scheme from the set of NSFs and their various implementations for this scheme, and is independent of how the NSFs are implemented (e.g., run in Virtual Machines or physical appliances). This document defines an object-oriented information model for network security, content security, and attack mitigation Capabilities, along with associated I2NSF Policy objects.

This document is organized as follows. Section 2 defines conventions and acronyms used. Section 3 discusses the design principles for the I2NSF Capability information model and related policy model objects. Section 4 defines the structure of the information model, which describes the policy and capability objects design; details of the model elements are contained in the appendices.

2. Conventions used in this document

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

This document uses terminology defined in [I-D.draft-ietf-i2nsf-terminology] for security related and I2NSF scoped terminology.

2.1. Acronyms

```
AAA:     Access control, Authorization, Authentication
ACL:     Access Control List
(D)DoD:  (Distributed) Denial of Service (attack)
ECA:     Event-Condition-Action
FMR:     First Matching Rule (resolution strategy)
FW:      Firewall
GNSF:    Generic Network Security Function
HTTP:    HyperText Transfer Protocol
I2NSF:   Interface to Network Security Functions
IPS:     Intrusion Prevention System
LMR:     Last Matching Rule (resolution strategy)
MIME:    Multipurpose Internet Mail Extensions
NAT:     Network Address Translation
NSF:     Network Security Function
RPC:     Remote Procedure Call
SMA:     String Matching Algorithm
URL:     Uniform Resource Locator
VPN:     Virtual Private Network
```

3. Information Model Design

The starting point of the design of the Capability information model is the categorization of types of security functions. For instance, experts agree on what is meant by the terms "IPS", "Anti-Virus", and "VPN concentrator". Network security experts unequivocally refer to "packet filters" as stateless devices able to allow or deny packet forwarding based on various conditions (e.g., source and destination IP addresses, source and destination ports, and IP protocol type fields) [Alshaer].

However, more information is required in case of other devices, like stateful firewalls or application layer filters. These devices filter packets or communications, but there are differences in the packets and communications that they can categorize and the states they maintain. Analogous considerations can be applied for channel protection protocols, where we all understand that they will protect packets by means of symmetric algorithms whose keys could have been negotiated with asymmetric cryptography, but they may work at different layers and support different algorithms and protocols. To ensure protection, these protocols apply integrity, optionally confidentiality, anti-reply protections, and authenticate peers.

3.1. Capability Information Model Overview

This document defines a model of security Capabilities that provides the foundation for automatic management of NSFs. This includes enabling the security controller to properly identify and manage NSFs, and allow NSFs to properly declare their functionality, so that they can be used in the correct way.

Some basic design principles for security Capabilities and the systems that have to manage them are:

- o Independence: each security Capability should be an independent function, with minimum overlap or dependency on other Capabilities. This enables each security Capability to be utilized and assembled together freely. More importantly, changes to one Capability will not affect other Capabilities. This follows the Single Responsibility Principle [Martin] [OODSRP].
- o Abstraction: each Capability should be defined in a vendor-independent manner, and associated to a well-known interface to provide a standardized ability to describe and report its processing results. This facilitates multi-vendor interoperability.
- o Automation: the system must have the ability to auto-discover, auto-negotiate, and auto-update its security Capabilities (i.e., without human intervention). These features are especially useful for the management of a large number of NSFs. They are essential to add smart services (e.g., analysis,

refinement, Capability reasoning, and optimization) for the security scheme employed. These features are supported by many design patterns, including the Observer Pattern [OODOP], the Mediator Pattern [OODMP], and a set of Message Exchange Patterns [Hohpe].

- o Scalability: the management system must have the Capability to scale up/down or scale in/out. Thus, it can meet various performancerequirements derived from changeable network traffic or service requests. In addition, security Capabilities that are affected by scalability changes must support reporting statistics to the security controller to assist its decision on whether it needs to invoke scaling or not. However, this requirement is for information only, and is beyond the scope of this document.

Based on the above principles, a set of abstract and vendor-neutral Capabilities with standard interfaces is defined. This provides a Capability model that enables a set of NSFs that are required at a given time to be selected, as well as the unambiguous definition of the security offered by the set of NSFs used. The security controller can compare the requirements of users and applications to the set of Capabilities that are currently available in order to choose which NSFs are needed to meet those requirements. Note that this choice is independent of vendor, and instead relies specifically on the Capabilities (i.e., the description) of the functions provided. The security controller may also be able to customize the functionality of selected NSFs.

Furthermore, when an unknown threat (e.g., zero-day exploits and unknown malware) is reported by a NSF, new Capabilities may be created, and/or existing Capabilities may be updated (e.g., by updating its signature and algorithm). This results in enhancing existing NSFs (and/or creating new NSFs) to address the new threats. New Capabilities may be sent to and stored in a centralized repository, or stored separately in a vendor’s local repository. In either case, a standard interface facilitates the update process.

Note that most systems cannot dynamically create a new Capability without human interaction. This is an area for further study.

3.2. ECA Policy Model Overview

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules; definitions of all I2NSF policy-related terms are also defined in [I-D.draft-ietf-i2nsf-terminology]:

- o Event: 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.

Examples of an I2NSF Event include time and user actions (e.g., logon, logoff, and actions that violate an ACL).

- o Condition: 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.
- o Action: 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.

An I2NSF Policy Rule is made up of three Boolean clauses: an Event clause, a Condition clause, and an Action clause. A Boolean clause is a logical statement that evaluates to either TRUE or FALSE. It may be made up of one or more terms; if more than one term, then a Boolean clause connects the terms using logical connectives (i.e., AND, OR, and NOT). It has the following semantics:

```
IF <event-clause> is TRUE
   IF <condition-clause> is TRUE
      THEN execute <action-clause>
   END-IF
END-IF
```

Technically, the "Policy Rule" is really a container that aggregates the above three clauses, as well as metadata.

The above ECA policy model is very general and easily extensible, and can avoid potential constraints that could limit the implementation of generic security Capabilities.

3.3. Relation with the External Information Model

Note: the symbology used from this point forward is taken from section 3.3 of [I-D.draft-ietf-supa-generic-policy-info-model].

The I2NSF NSF-Facing Interface is in charge of selecting and managing the NSFs using their Capabilities. This is done using the following approach:

1) Each NSF registers its Capabilities with the management system when it "joins", and hence makes its Capabilities available to the management system;
2) The security controller selects the set of Capabilities required to meet the needs of the security service from all available NSFs that it manages;

3) The security controller uses the Capability information model to match chosen Capabilities to NSFs, independent of vendor;
4) The security controller takes the above information and creates or uses one or more data models from the Capability information model to manage the NSFs;
5) Control and monitoring can then begin.

This assumes that an external information model is used to define the concept of an ECA Policy Rule and its components (e.g., Event, Condition, and Action objects). This enables I2NSF Policy Rules [I-D.draft-ietf-i2nsf-terminology] to be subclassed from an external information model.

Capabilities are defined as classes (e.g., a set of objects that exhibit a common set of characteristics and behavior [I-D.draft-ietf-supa-generic-policy-info-model].

Each Capability is made up of at least one model element (e.g., attribute, method, or relationship) that differentiates it from all other objects in the system. Capabilities are, generically, a type of metadata (i.e., information that describes, and/or prescribes, the behavior of objects); hence, it is also assumed that an external information model is used to define metadata (preferably, in the form of a class hierarchy). Therefore, it is assumed that Capabilities are subclassed from an external metadata model.

The Capability sub-model is used for advertising, creating, selecting, and managing a set of specific security Capabilities independent of the type and vendor of device that contains the NSF. That is, the user of the NSF-Facing Interface does not care whether the NSF is virtualized or hosted in a physical device, who the vendor of the NSF is, and which set of entities the NSF is communicating with (e.g., a firewall or an IPS). Instead, the user only cares about the set of Capabilities that the NSF has, such as packet filtering or deep packet inspection. The overall structure is illustrated in the figure below:

```
+-------------------------+ 0..n        0..n +---------------+
|                         |/ \             \|   External    |
| External ECA Info Model + A ----------------+   Metadata    |
|                         |\ /  Aggregates   /|  Info Model   |
+-----------+-----------+      Metadata      +-------+-------+
            |                                       / \
            |                                        |
           / \                                       |
Subclasses derived for I2NSF                   +-----+------+
     Security Policies                         | Capability |
                                               | Sub-Model  |
                                               +------------+
```

Figure 1. The Overall I2NSF Information Model Design

This draft defines a set of extensions to a generic, external, ECA Policy Model to represent various NSF ECA Security Policy Rules. It also defines the Capability Sub-Model; this enables ECA Policy Rules to control which Capabilities are seen by which actors, and used by the I2NSF system. Finally, it places requirements on what type of extensions are required to the generic, external, ECA information model and metadata models, in order to manage the lifecycle of I2NSF Capabilities.

Both of the external models shown in Figure 1 could, but do not have to, be based on the SUPA information model [I-D.draft-ietf-supa-generic-policy-info-model]. Note that classes in the Capability Sub-Model will inherit the AggregatesMetadata aggregation from the External Metadata Information Model.

The external ECA Information Model supplies at least a set of classes that represent a generic ECA Policy Rule, and a set of classes that represent Events, Conditions, and Actions that can be aggregated by the generic ECA Policy Rule. This enables I2NSF to reuse this generic model for different purposes, as well as refine it (i.e., create new subclasses, or add attributes and relationships) to represent I2NSF-specific concepts.

It is assumed that the external ECA Information Model has the ability to aggregate metadata. Capabilities are then sub-classed from an appropriate class in the external Metadata Information Model; this enables the ECA objects to use the existing aggregation between them and Metadata to add Metadata to appropriate ECA objects.

Detailed descriptions of each portion of the information model are given in the following sections.

3.4. I2NSF Capability Information Model: Theory of Operation

Capabilities are typically used to represent NSF functions that can be invoked. Capabilities are objects, and hence, can be used in the event, condition, and/or action clauses of an I2NSF ECA Policy Rule. The I2NSF Capability information model refines a predefined metadata model; the application of I2NSF Capabilities is done by refining a predefined ECA Policy Rule information model that defines how to use, manage, or otherwise manipulate a set of Capabilities. In this approach, an I2NSF Policy Rule is a container that is made up of three clauses: an event clause, a condition clause, and an action clause. When the I2NSF policy engine receives a set of events, it matches those events to events in active ECA Policy Rules. If the event matches, then this triggers the evaluation of the condition clause of the matched I2NSF Policy Rule. The condition clause is then evaluated; if it matches, then the set of actions in the matched I2NSF Policy Rule MAY be executed.

This document defines additional important extensions to both the external ECA Policy Rule model and the external Metadata model that are used by the I2NSF Information Model; examples include resolution strategy, external data, and default action. All these extensions come from the geometric model defined in [Bas12]. A more detailed description is provided in Appendix E; a summary of the important points follows.

Formally, given a set of actions in an I2NSF Policy Rule, the resolution strategy maps all the possible subsets of actions to an outcome. In other words, the resolution strategy is included in the I2NSF Policy Rule to decide how to evaluate all the actions in a particular I2NSF Policy Rule. This is then extended to include all possible I2NSF Policy Rules that can be applied in a particular scenario. Hence, the final action set from all I2NSF Policy Rules is deduced.

Some concrete examples of resolution strategy are the First Matching Rule (FMR) or Last Matching Rule (LMR) resolution strategies. When no rule matches a packet, the NSFs may select a default action, if they support one.

Resolution strategies may use, besides intrinsic rule data (i.e., event, condition, and action clauses), "external data" associated to each rule, such as priority, identity of the creator, and creation time. Two examples of this are attaching metadata to the policy action and/or policy rule, and associating the policy rule with another class to convey such information.

3.4.1. I2NSF Condition Clause Operator Types

After having analyzed the literature and some existing NSFs, the types of selectors are categorized as exact-match, range-based, regex-based, and custom-match [Bas15][Lunt].

Exact-match selectors are (unstructured) sets: elements can only be checked for equality, as no order is defined on them. As an example, the protocol type field of the IP header is an unordered set of integer values associated to protocols. The assigned protocol numbers are maintained by the IANA (http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml).

In this selector, it is only meaningful to specify condition clauses that use either the "equals" or "not equals" operators:

```
proto = tcp, udp       (protocol type field equals to TCP or UDP)
proto != tcp           (protocol type field different from TCP)
```

No other operators are allowed on exact-match selectors. For example, the following is an invalid condition clause, even if protocol types map to integers:

```
proto < 62             (invalid condition)
```

Range-based selectors are ordered sets where it is possible to naturally specify ranges as they can be easily mapped to integers. As an example, the ports in the TCP protocol may be represented with a range-based selector (e.g., 1024-65535). As another example, the following are examples of valid condition clauses:

```
source_port = 80
source_port < 1024
source_port < 30000 && source_port >= 1024
```

We include, in range-based selectors, the category of selectors that have been defined by Al-Shaer et al. as "prefix-match" [Alshaer]. These selectors allow the specification of ranges of values by means of simple regular expressions. The typical case is the IP address selector (e.g., 10.10.1.*).

There is no need to distinguish between prefix match and range-based selectors; for example, the address range "10.10.1.*" maps to "[10.10.1.0,10.10.1.255]".

Another category of selector types includes those based on regular expressions. This selector type is used frequently at the application layer, where data are often represented as strings of text. The regex-based selector type also includes string-based selectors, where matching is evaluated using string matching algorithms (SMA) [Cormen]. Indeed, for our purposes, string matching can be mapped to regular expressions, even if in practice SMA are much faster. For instance, Squid (http://www.squid-cache.org/), a popular Web caching proxy that offers various access control Capabilities, allows the definition of conditions on URLs that can be evaluated with SMA (e.g., dstdomain) or regex matching (e.g., dstdom_regex).

As an example, the condition clause:

```
"URL = *.website.*"
```

matches all the URLs that contain a subdomain named website and the ones whose path contain the string ".website.". As another example, the condition clause:

```
"MIME_type = video/*"
```

matches all MIME objects whose type is video.

Finally, the idea of a custom check selector is introduced. For instance, malware analysis can look for specific patterns, and returns a Boolean value if the pattern is found or not.

In order to be properly used by high-level policy-based processing systems (such as reasoning systems and policy translation systems), these custom check selectors can be modeled as black-boxes (i.e., a function that has a defined set of inputs and outputs for a particular state), which provide an associated Boolean output.

More examples of custom check selectors will be presented in the next versions of the draft. Some examples are already present in Section 6.

3.4.2. Capability Selection and Usage

Capability selection and usage are based on the set of security traffic classification and action features that an NSF provides; these are defined by the Capability model. If the NSF has the classification features needed to identify the packets/flows required by a policy, and can enforce the needed actions, then that particular NSF is capable of enforcing the policy.

NSFs may also have specific characteristics that automatic processes or administrators need to know when they have to generate configurations, like the available resolution strategies and the possibility to set default actions.

The Capability information model can be used for two purposes: describing the features provided by generic security functions, and describing the features provided by specific products. The term Generic Network Security Function (GNSF) refers to the classes of security functions that are known by a particular system. The idea is to have generic components whose behavior 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. GNSF examples include packet filter, URL filter, HTTP filter, VPN gateway, anti-virus, anti-malware, content filter, monitoring, and anonymity proxy; these will be described later in a revision of this draft as well as in an upcoming data model contribution.

The next section will introduce the algebra to define the information model of Capability registration. This associates NSFs to Capabilities, and checks whether a NSF has the Capabilities needed to enforce policies.

3.4.3. Capability Algebra

We introduce a Capability Algebra to ensure that the actions of different policy rules do not conflict with each other.

Formally, two I2NSF Policy Actions conflict with each other if:

- o the event clauses of each evaluate to TRUE
- o the condition clauses of each evaluate to TRUE
- o the action clauses affect the same object in different ways

For example, if we have two Policies:

P1: During 8am-6pm, if traffic is external, then run through FW
P2: During 7am-8pm, conduct anti-malware investigation

There is no conflict between P1 and P2, since the actions are different. However, consider these two policies:

P3: During 8am-6pm, John gets GoldService
P4: During 10am-4pm, FTP from all users gets BronzeService

P3 and P4 are now in conflict, because between the hours of 10am and 4pm, the actions of P3 and P4 are different and apply to the same user (i.e., John).

Let us define the concept of a "matched" policy rule as one in which its event and condition clauses both evaluate to true. This enables the actions in this policy rule to be evaluated. Then, the conflict matrix is defined by a 5-tuple {Ac, Cc, Ec, RSc, Dc}, where:

- o Ac is the set of Actions currently available from the NSF;
- o Cc is the set of Conditions currently available from the NSF;
- o Ec is the set of Events the NSF is able to respond to. Therefore, the event clause of an I2NSF ECA Policy Rule that is written for an NSF will only allow a set of designated events in Ec. For compatibility purposes, we will assume that if Ec={} (that is, Ec is empty), the NSF only accepts CA policies.
- o RSc is the set of Resolution Strategies that 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;
- o Dc defines the notion of a Default action that 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. This field of the Capability algebra can take the following values:
  - An explicit action (that has been predefined; typically, this means that it is fixed and not configurable), denoted as Dc ={a}. In this case, the NSF will always use the action as as the default action.
  - A set of explicit actions, denoted Dc={a1,a2, ...}; typically, this means that any **one** action can be used as the default action. This enables the policy writer to choose one of a predefined set of actions {a1, a2, ...} to serve as the default action.

- A fully configurable default action, denoted as Dc={F}. Here, F is a dummy symbol (i.e., a placeholder value) that can be used to indicate that the default action can be freely selected by the policy editor from the actions Ac available at the NSF. In other words, one of the actions Ac may be selected by the policy writer to act as the default action.
- No default action, denoted as Dc={}, for cases where the NSF does not allow the explicit selection of a default action.

```
*** Note to WG: please review the following paragraphs
*
*  Interesting Capability concepts that could be considered for a next
*  version of the Capability model and algebra include:
*
*     o Event clause representation (e.g., conjunctive vs. disjunctive
*       normal form for Boolean clauses)
*     o Event clause evaluation function, which would enable more
*       complex expressions than simple Boolean expressions to be used
*
*
*     o Condition clause representation (e.g., conjunctive vs.
*       disjunctive normal form for Boolean clauses)
*     o Condition clause evaluation function, which would enable more
*       complex expressions than simple Boolean expressions to be used
*     o Action clause evaluation strategies (e.g., execute first
*       action only, execute last action only, execute all actions,
*       execute all actions until an action fails)
*     o The use of metadata, which can be associated to both an I2NSF
*       Policy Rule as well as objects contained in the I2NSF Policy
*       Rule (e.g., an action), that describe the object and/or
*       prescribe behavior. Descriptive examples include adding
*       authorship information and defining a time period when an NSF
*       can be used to be defined; prescriptive examples include
*       defining rule priorities and/or ordering.
*
*  Given two sets of Capabilities, denoted as
*
*     cap1=(Ac1,Cc1,Ec1,RSc1,Dc1) and
*     cap2=(Ac2,Cc2,Ec2,RSc2,Dc2),
*
*  two set operations are defined for manipulating Capabilities:
*
*     o Capability addition:
*          cap1+cap2 = {Ac1 U Ac2, Cc1 U Cc2, Ec1 U Ec2, RSc1, Dc1}
*     o Capability subtraction:
*          cap1-cap2 = {Ac1 \ Ac2, Cc1 \ Cc2, Ec1 \ Ec2, RSc1, Dc1}
*
*  In the above formulae, "U" is the set union operator and "\" is the
*  set difference operator.
```

```
*  The addition and subtraction of Capabilities are defined as the
*  addition (set union) and subtraction (set difference) of both the
*  Capabilities and their associated actions. Note that **only** the
*  leftmost (in this case, the first matched policy rule) Resolution
*  Strategy and Default Action are used.
*
*  Note: actions, events, and conditions are **symmetric**. This means
*  that when two matched policy rules are merged, the resultant actions
*  and Capabilities are defined as the union of each individual matched
*  policy rule. However, both resolution strategies and default actions
*  are **asymmetric** (meaning that in general, they can **not** be
*  combined, as one has to be chosen). In order to simplify this, we
*  have chosen that the **leftmost** resolution strategy and the
*  **leftmost** default action are chosen. This enables the developer
*  to view the leftmost matched rule as the "base" to which other
*  elements are added.
*
*  As an example, assume that a packet filter Capability, Cpf, is
*  defined. Further, assume that a second Capability, called Ctime,
*  exists, and that it defines time-based conditions. Suppose we need
*  to construct a new generic packet filter, Cpfgen, that adds
*  time-based conditions to Cpf.
*
*
*  Conceptually, this is simply the addition of the Cpf and Ctime
*  Capabilities, as follows:
*     Apf   =  {Allow, Deny}
*     Cpf   =  {IPsrc,IPdst,Psrc,Pdst,protType}
*     Epf   =  {}
*     RSpf  =  {FMR}
*     Dpf   =  {A1}
*
*     Atime =  {Allow, Deny, Log}
*     Ctime =  {timestart, timeend, datestart, datestop}
*     Etime =  {}
*     RStime = {LMR}
*     Dtime =  {A2}
*
*  Then, Cpfgen is defined as:
*     Cpfgen = {Apf U Atime, Cpf U Ctime, Epf U Etime, RSpf, Dpf}
*            = {Allow, Deny, Log},
*              {{IPsrc, IPdst, Psrc, Pdst, protType} U
*               {timestart, timeend, datestart, datestop}},
*              {},
*              {FMR},
*              {A1}
*
*  In other words, Cpfgen provides three actions (Allow, Deny, Log),
*  filters traffic based on a 5-tuple that is logically ANDed with a
*  time period, and uses FMR; it provides A1 as a default action, and
*  it does not react to events.
```

* Note: We are investigating, for a next revision of this draft, the
* possibility to add further operations that do not follow the
* symmetric vs. asymmetric properties presented in the previous note.
* We are looking for use cases that may justify the complexity added
* by the availability of more Capability manipulation operations.
*
*** End Note to WG

3.5. Initial NSFs Capability Categories

The following subsections define three common categories of Capabilities: network security, content security, and attack mitigation. Future versions of this document may expand both the number of categories as well as the types of Capabilities within a given category.

3.5.1. Network Security Capabilities

Network security is a category that describes the inspecting and processing of network traffic based on the use of pre-defined security policies.

The inspecting portion may be thought of as a packet-processing engine that inspects packets traversing networks, either directly or in the context of flows with which the packet is associated. From the perspective of packet-processing, implementations differ in the depths of packet headers and/or payloads they can inspect, the various flow and context states they can maintain, and the actions that can be applied to the packets or flows.

3.5.2. Content Security Capabilities

Content security is another category of security Capabilities applied to the application layer. Through analyzing traffic contents carried in, for example, the application layer, content security Capabilities can be used to identify various security functions that are required. These include defending against intrusion, inspecting for viruses, filtering malicious URL or junk email, blocking illegal web access, or preventing malicious data retrieval.

Generally, each type of threat in the content security category has a set of unique characteristics, and requires handling using a set of methods that are specific to that type of content. Thus, these Capabilities will be characterized by their own content-specific security functions.

3.5.3. Attack Mitigation Capabilities

This category of security Capabilities is used to detect and mitigate various types of network attacks. Today's common network attacks can be classified into the following sets:

- o DDoS attacks:
  - Network layer DDoS attacks: Examples include SYN flood, UDP flood, ICMP flood, IP fragment flood, IPv6 Routing header attack, and IPv6 duplicate address detection attack;
  - Application layer DDoS attacks: Examples include HTTP flood, https flood, cache-bypass HTTP floods, WordPress XML RPC floods, and ssl DDoS.
- o Single-packet attacks:
  - Scanning and sniffing attacks: IP sweep, port scanning, etc.
  - malformed packet attacks: Ping of Death, Teardrop, etc.
  - special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc.

Each type of network attack has its own network behaviors and packet/flow characteristics. Therefore, each type of attack needs a special security function, which is advertised as a set of Capabilities, for detection and mitigation. The implementation and management of this category of security Capabilities of attack mitigation control is very similar to the content security control category. A standard interface, through which the security controller can choose and customize the given security Capabilities according to specific requirements, is essential.

4. Information Sub-Model for Network Security Capabilities

The purpose of the Capability Information Sub-Model is to define the concept of a Capability, and enable Capabilities to be aggregated to appropriate objects. The following sections present the Network Security, Content Security, and Attack Mitigation Capability sub-models.

4.1. Information Sub-Model for Network Security

The purpose of the Network Security Information Sub-Model is to define how network traffic is defined, and determine if one or more network security features need to be applied to the traffic or not. Its basic structure is shown in the following figure:

```
                                      +---------------------+
     +---------------+ 1..n      1..n |                     |
     |               |/ \           \| A Common Superclass |
     | ECAPolicyRule + A -------------+   for ECA Objects   |
     |               |\ /           /|                     |
     +-------+-------+                +---------+-----------+
            / \                                / \
             |                                  |
             |                                  |
  (subclasses to define Network        (subclasses of Event,
    Security ECA Policy Rules      Condition, and Action Objects
    extensibly, so that other          for Network Security
    Policy Rules can be added)             Policy Rules)
```

Figure 2. Network Security Information Sub-Model Overview

In the above figure, the ECAPolicyRule, along with the Event, Condition, and Action Objects, are defined in the external ECA Information Model. The Network Security Sub-Model extends all of these objects in order to define security-specific ECA Policy Rules, as well as extensions to the (generic) Event, Condition, and Action objects.

An I2NSF Policy Rule is a special type of Policy Rule that is in event-condition-action (ECA) form. It consists of the Policy Rule, components of a Policy Rule (e.g., events, conditions, actions, and some extensions like resolution policy, default action and external data), and optionally, metadata. It can be applied to both uni- and bi-directional traffic across the NSF.

Each rule is triggered by one or more events. If the set of events evaluates to true, then a set of conditions are evaluated and, if true, enable a set of actions to be executed. This takes the following conceptual form:

```
IF <event-clause> is TRUE
   IF <condition-clause> is TRUE
      THEN execute <action-clause>
   END-IF
END-IF
```

In the above example, the Event, Condition, and Action portions of a Policy Rule are all **Boolean Clauses**. Hence, they can contain combinations of terms connected by the three logical connectives operators (i.e., AND, OR, NOT). An example is:

```
((SLA==GOLD) AND ((numPackets>burstRate) OR NOT(bwAvail<minBW)))
```

Note that Metadata, such as Capabilities, can be aggregated by I2NSF ECA Policy Rules.

4.1.1. Network Security Policy Rule Extensions

Figure 3 shows an example of more detailed design of the ECA Policy Rule subclasses that are contained in the Network Security Information Sub-Model, which just illustrates how more specific Network Security Policies are inherited and extended from the SecurityECAPolicyRule class. Any new kinds of specific Network Security Policy can be created by following the same pattern of class design as below.

```
                            +---------------+
                            |    External   |
                            | ECAPolicyRule |
                            +-------+-------+
                                   / \
                                    |
                                    |
                       +------------+----------+
                       | SecurityECAPolicyRule |
                       +------------+----------+
                                    |
                                    |
          +----+-----+--------+-----+----+---------+---------+--- ...
          |          |        |          |         |         |
          |          |        |          |         |         |
   +------+-------+  |  +-----+-------+  |  +------+------+  |
   |Authentication|  |  |  Accounting |  |  |ApplyProfile |  |
   |ECAPolicyRule |  |  |ECAPolicyRule|  |  |ECAPolicyRule|  |
   +--------------+  |  +-------------+  |  +-------------+  |
                     |                   |                   |
              +------+------+     +------+------+     +--------------+
              |Authorization|     |   Traffic   |     |ApplySignature|
              |ECAPolicyRule|     | Inspection  |     |ECAPolicyRule |
              +-------------+     |ECAPolicyRule|     +--------------+
                                  +-------------+
```

Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

The SecurityECAPolicyRule is the top of the I2NSF ECA Policy Rule hierarchy. It inherits from the (external) generic ECA Policy Rule, and represents the specialization of this generic ECA Policy Rule to add security-specific ECA Policy Rules. The SecurityECAPolicyRule contains all of the attributes, methods, and relationships defined in its superclass, and adds additional concepts that are required for Network Security (these will be defined in the next version of this draft). The six SecurityECAPolicyRule subclasses extend the SecurityECAPolicyRule class to represent six different types of Network Security ECA Policy Rules. It is assumed that the (external) generic ECAPolicyRule class defines basic information in the form of attributes, such as an unique object ID, as well as a description and other necessary information.

*** Note to WG
*
*   The design in Figure 3 represents the simplest conceptual design
*   for network security. An alternative model would be to use a
*   software pattern (e.g., the Decorator pattern); this would result
*   in the SecurityECAPolicyRule class being "wrapped" by one or more
*   of the six subclasses shown in Figure 3. The advantage of such a
*   pattern is to reduce the number of active objects at runtime, as
*   well as offer the ability to combine multiple actions of different
*   policy rules (e.g., inspect traffic and then apply a filter) into
*   one. The disadvantage is that it is a more complex software design.
*   The design team is requesting feedback from the WG regarding this.
*
*** End of Note to WG

It is assumed that the (external) generic ECA Policy Rule is abstract; the SecurityECAPolicyRule is also abstract. This enables data model optimizations to be made while making this information model detailed but flexible and extensible. For example, abstract classes may be collapsed into concrete classes.

The SecurityECAPolicyRule defines network security policy as a container that aggregates Event, Condition, and Action objects, which are described in Section 4.1.3, 4.1.4, and 4.1.5, respectively. Events, Conditions, and Actions can be generic or security-specific.

Brief class descriptions of these six ECA Policy Rules are provided in Appendix A.

4.1.2. Network Security Policy Rule Operation

A Network Security Policy consists of one or more ECA Policy Rules formed from the information model described above. In simpler cases, where the Event and Condition clauses remain unchanged, then the action of one Policy Rule may invoke additional network security actions from other Policy Rules. Network security policy examines and performs basic processing of the traffic as follows:

1. The NSF evaluates the Event clause of a given SecurityECAPolicyRule (which can be generic or specific to security, such as those in Figure 3). It may use security Event objects to do all or part of this evaluation, which are defined in section 4.1.3. If the Event clause evaluates to TRUE, then the Condition clause of this SecurityECAPolicyRule is evaluated; otherwise, the execution of this SecurityECAPolicyRule is stopped, and the next SecurityECAPolicyRule (if one exists) is evaluated.

2. The Condition clause is then evaluated. It may use security Condition objects to do all or part of this evaluation, which are defined in section 4.1.4. If the Condition clause evaluates to TRUE, it is defined as "matching" the SecurityECAPolicyRule; otherwise, execution of this SecurityECAPolicyRule is stopped, and the next SecurityECAPolicyRule (if one exists) is evaluated.
3. The set of actions to be executed are retrieved, and then the resolution strategy is used to define their execution order. This process includes using any optional external data associated with the SecurityECAPolicyRule.
4. Execution then takes one of the following three forms:
   a. If one or more actions is selected, then the NSF may perform those actions as defined by the resolution strategy. For example, the resolution strategy may only allow a single action to be executed (e.g., FMR or LMR), or it may allow all actions to be executed (optionally, in a particular order). In these and other cases, the NSF Capability MUST clearly define how execution will be done. It may use security Action objects to do all or part of this execution, which are defined in section 4.1.5. If the basic Action is permit or mirror, the NSF firstly performs that function, and then checks whether certain other security Capabilities are referenced in the rule. If yes, go to step 5. If no, the traffic is permitted.
   b. If no actions are selected, and if a default action exists, then the default action is performed. Otherwise, no actions are performed.
   c. Otherwise, the traffic is denied.
5. If other security Capabilities (e.g., the conditions and/or actions implied by Anti-virus or IPS profile NSFs) are referenced in the action set of the SecurityECAPolicyRule, the NSF can be configured to use the referenced security Capabilities (e.g., check conditions or enforce actions). Execution then terminates.

One policy or rule can be applied multiple times to different managed objects (e.g., links, devices, networks, VPNS). This not only guarantees consistent policy enforcement, but also decreases the configuration workload.

4.1.3. Network Security Event Sub-Model

Figure 4 shows a more detailed design of the Event subclasses that are contained in the Network Security Information Sub-Model.

The four Event classes shown in Figure 4 extend the (external) generic Event class to represent Events that are of interest to Network Security. It is assumed that the (external) generic Event class defines basic Event information in the form of attributes, such as a unique event ID, a description, as well as the date and time that the event occurred.

```
                                     +---------------------+
        +---------------+ 1..n   1..n|                     |
        |               |/ \        \| A Common Superclass |
        | ECAPolicyRule + A ---------+   for ECA Objects   |
        |               |\ /        /|                     |
        +---------------+            +---------+-----------+
                                              / \
                                               |
                                               |
                   +---------------+-----------+------+
                   |               |                  |
                   |               |                  |
             +-----+----+   +------+------+     +-----+-----+
             | An Event |   | A Condition |     | An Action |
             |   Class  |   |    Class    |     |   Class   |
             +-----+----+   +-------------+     +-----------+
                  / \
                   |
                   |
             +-----+---------+----------------+--------------+-- ...
             |               |                |              |
             |               |                |              |
     +-------+----+ +--------+-----+ +--------+-----+ +------+-----+
     |UserSecurity| |    Device    | |    System    | |TimeSecurity|
     |   Event    | | SecurityEvent| | SecurityEvent| |     Event  |
     +------------+ +--------------+ +--------------+ +------------+
```

Figure 4. Network Security Info Sub-Model Event Class Extensions

The following are assumptions that define the functionality of the generic Event class. If desired, these could be defined as attributes in a SecurityEvent class (which would be a subclass of the generic Event class, and a superclass of the four Event classes shown in Figure 4). However, this makes it harder to use any generic Event model with the I2NSF events. Assumptions are:

- All four SecurityEvent subclasses are concrete
- The generic Event class uses the composite pattern, so individual Events as well as hierarchies of Events are available (the four subclasses in Figure 4 would be subclasses of the Atomic Event class); otherwise, a mechanism is needed to be able to group Events into a collection
- The generic Event class has a mechanism to uniquely identify the source of the Event
- The generic Event class has a mechanism to separate header information from its payload
- The generic Event class has a mechanism to attach zero or more metadata objects to it

```
*** Note to WG:
*
*   The design in Figure 4 represents the simplest conceptual design
*   design for describing Security Events. An alternative model would
*   be to use a software pattern (e.g., the Decorator pattern); this
*   would result in the SecurityEvent class being "wrapped" by one or
*   more of the four subclasses shown in Figure 4. The advantage of
*   such a pattern is to reduce the number of active objects at runtime,
*   as well as offer the ability to combine multiple events of different
*   types into one. The disadvantage is that it is a more complex
*   software design.
*
*** End of Note to WG
```

Brief class descriptions are provided in Appendix B.

4.1.4. Network Security Condition Sub-Model

Figure 5 shows a more detailed design of the Condition subclasses that are contained in the Network Security Information Sub-Model. The six Condition classes shown in Figure 5 extend the (external) generic Condition class to represent Conditions that are of interest to Network Security. It is assumed that the (external) generic Condition class is abstract, so that data model optimizations may be defined. It is also assumed that the generic Condition class defines basic Condition information in the form of attributes, such as a unique object ID, a description, as well as a mechanism to attach zero or more metadata objects to it. While this could be defined as attributes in a SecurityCondition class (which would be a subclass of the generic Condition class, and a superclass of the six Condition classes shown in Figure 5), this makes it harder to use any generic Condition model with the I2NSF conditions.

```
*** Note to WG:
*
*   The design in Figure 5 represents the simplest conceptual design
*   for describing Security Conditions. An alternative model would be
*   to use a software pattern (e.g., the Decorator pattern); this would
*   result in the SecurityCondition class being "wrapped" by one or
*   more of the six subclasses shown in Figure 5. The advantage of such
*   a pattern is to reduce the number of active objects at runtime, as
*   well as offer the ability to combine multiple conditions of
*   different types into one. The disadvantage is that it is a more
*   complex software design.
*   The design team is requesting feedback from he WG regarding this.
*
*** End of Note to WG
```

```
                                     +---------------------+
     +---------------+ 1..n     1..n |                     |
     |               |/ \          \| A Common Superclass |
     | ECAPolicyRule+ A -------------+   for ECA Objects   |
     |               |\ /          /|                     |
     +-------+-------+               +-----------+---------+
                                                / \
                                                 |
                                                 |
                       +--------------+----------+----+
                       |              |               |
                       |              |               |
                 +-----+----+  +------+------+  +-----+-----+
                 | An Event |  | A Condition |  | An Action |
                 |   Class  |  |    Class    |  |   Class   |
                 +----------+  +------+------+  +-----------+
                                     / \
                                      |
                                      |
           +--------+----------+------+---+---------+--------+--- ...
           |        |          |          |         |        |
           |        |          |          |         |        |
     +-----+-----+  |  +-------+-------+  |  +------+-----+  |
     |   Packet  |  |  | PacketPayload |  |  |    Target  |  |
     |  Security |  |  |    Security   |  |  |  Security  |  |
     | Condition |  |  |   Condition   |  |  |  Condition |  |
     +-----------+  |  +---------------+  |  +------------+  |
                    |                     |                  |
             +------+-------+  +----------+------+  +--------+-------+
             | UserSecurity |  | SecurityContext |  | GenericContext |
             |   Condition  |  |    Condition    |  |    Condition   |
             +--------------+  +-----------------+  +----------------+
```

Figure 5. Network Security Info Sub-Model Condition Class Extensions

Brief class descriptions are provided in Appendix C.

4.1.5. Network Security Action Sub-Model

Figure 6 shows a more detailed design of the Action subclasses that are contained in the Network Security Information Sub-Model.

The four Action classes shown in Figure 6 extend the (external) generic Action class to represent Actions that perform a Network Security Control function.

The three Action classes shown in Figure 6 extend the (external) generic Action class to represent Actions that are of interest to Network Security. It is assumed that the (external) generic Action class is abstract, so that data model optimizations may be defined.

```
                                  +---------------------+
      +---------------+ 1..n     1..n |                     |
      |               |/ \          \| A Common Superclass |
      | ECAPolicyRule+ A -------------+   for ECA Objects   |
      |               |\ /          /|                     |
      +---------------+               +-----------+---------+
                                                 / \
                                                  |
                                                  |
                    +--------------+--------+------+
                    |              |               |
                    |              |               |
              +-----+----+  +------+------+  +-----+-----+
              | An Event |  | A Condition |  | An Action |
              |   Class  |  |    Class    |  |   Class   |
              +----------+  +-------------+  +-----+-----+
                                                  / \
                                                   |
                                                   |
                 +------------------+---------------+------- ...
                 |                  |               |
                 |                  |               |
             +---+-----+       +----+---+     +------+-------+
             | Ingress |       | Egress |     | ApplyProfile |
             | Action  |       | Action |     |    Action    |
             +---------+       +--------+     +--------------+
```

Figure 6. Network Security Info Sub-Model Action Extensions

It is also assumed that the generic Action class defines basic Action information in the form of attributes, such as a unique object ID, a description, as well as a mechanism to attach zero or more metadata objects to it. While this could be defined as attributes in a SecurityAction class (which would be a subclass of the generic Action class, and a superclass of the six Action classes shown in Figure 6), this makes it harder to use any generic Action model with the I2NSF actions.

```
*** Note to WG
*   The design in Figure 6 represents the simplest conceptual design
*   for describing Security Actions. An alternative model would be to
*   use a software pattern (e.g., the Decorator pattern); this would
*   result in the SecurityAction class being "wrapped" by one or more
*   of the three subclasses shown in Figure 6. The advantage of such a
*   pattern is to reduce the number of active objects at runtime, as
*   well as offer the ability to combine multiple actions of different
*   types into one. The disadvantage is that it is a more complex
*   software design.
*   The design team is requesting feedback from the WG regarding this.
*** End of Note to WG
```

Brief class descriptions are provided in Appendix D.

4.2. Information Model for I2NSF Capabilities

The I2NSF Capability Model is made up of a number of Capabilities that represent various content security and attack mitigation functions. Each Capability protects against a specific type of threat in the application layer. This is shown in Figure 7.

```
+-------------------------+ 0..n         0..n +---------------+
|                         |/ \               \|   External    |
| External ECA Info Model + A ----------------+   Metadata    |
|                         |\ /  Aggregates   /|  Info Model   |
+----+--------------------+      Metadata     +-----+---------+
     |                                             / \
     |                                              |
    / \                                             |
 Subclasses    +------------------------------------+-----------+
  derived      | Capability                      |              |
 for I2NSF     | Sub-Model            +----------+---------+    |
Policy Rules   |                      | SecurityCapability |    |
               |                      +----------+---------+    |
               |                                 |              |
               |                                 |              |
               |          +----------------------+---+          |
               |          |                          |          |
               | +--------+---------+     +---------+---------+ |
               | | Content Security |     | Attack Mitigation | |
               | |   Capabilities   |     |    Capabilities   | |
               | +------------------+     +-------------------+ |
               +------------------------------------------------+
```

Figure 7. I2NSF Security Capability High-Level Model

Figure 7 shows a common I2NSF Security Capability class, called SecurityCapability. This enables us to add common attributes, relationships, and behavior to this class without affecting the design of the external metadata information model. All I2NSF Security Capabilities are then subclassed from the SecuritCapability class.

Note: the SecurityCapability class will be defined in the next version of this draft, after feedback from the WG is obtained.

4.3. Information Model for Content Security Capabilities

Content security is composed of a number of distinct security Capabilities; each such Capability protects against a specific type of threat in the application layer. Content security is a type of Generic Network Security Function (GNSF), which summarizes a well-defined set of security Capabilities, and was shown in Figure 7.

Figure 8 shows exemplary types of the content security GNSF.

```
     +--------------------------------------------------------------+
     |                             +--------------------+           |
     |   Capability                | SecurityCapability |           |
     |   Sub-Model:                +---------+----------+           |
     | Content Security                    / \                      |
     |                                      |                       |
     |                                      |                       |
     |       +-------+----------+----------+---------------+        |
     |       |       |          |                          |        |
     | +-----+----+  |  +-------+----+             +-------+------+ |
     | |Anti-Virus|  |  | Intrusion  |             |    Attack    | |
     | |Capability|  |  | Prevention |             |  Mitigation  | |
     | +----------+  |  | Capability |             | Capabilities | |
     |               |  +------------+             +--------------+ |
     |               |                                              |
     |      +--------+----+------------+-----------+--------+       |
     |      |             |            |           |        |       |
     | +----+-----+ +-----+----+ +-----+----+ +----+-----+  |       |
     | |   URL    | |   Mail   | |   File   | |  Data    |  |       |
     | |Filtering | |Filtering | |Filtering | |Filtering |  |       |
     | |Capability| |Capability| |Capability| |Capability|  |       |
     | +----------+ +----------+ +----------+ +----------+  |       |
     |                                                      |       |
     |             +----------------+------------------+----+       |
     |             |                |                  |            |
     |      +------+------+  +------+------+ +---------+---------+  |
     |      |PacketCapture|  |FileIsolation| |ApplicationBehavior|  |
     |      | Capability  |  | Capability  | |    Capability     |  |
     |      +-------------+  +-------------+ +-------------------+  |
     +--------------------------------------------------------------+
```

Figure 8. Network Security Capability Information Model

The detailed description about a standard interface, and the parameters for all the security Capabilities of this category, will be defined in a future version of this document.

4.4. Information Model for Attack Mitigation Capabilities

Attack mitigation is composed of a number of GNSFs; each one protects against a specific type of network attack. Attack Mitigation security is a type of GNSF, which summarizes a well-defined set of security Capabilities, and was shown in Figure 7. Figure 9 shows exemplary types of Attack Mitigation GNSFs.

```
     +---------------------------------------------------------------+
     |                           +--------------------+              |
     |    Capability             | SecurityCapability |              |
     |    Sub-Model:             +---------+----------+              |
     | Attack Mitigation                  / \                        |
     |                                     |                         |
     |                                     |                         |
     |       +-------+--------+------------+-------------+           |
     |       |       |        |                          |           |
     | +-----+----+  |  +-----+----+             +-------+------+    |
     | | SSLDDoS  |  |  | PortScan |             |    Content   |    |
     | |Capability|  |  |Capability|             |   Security   |    |
     | +----------+  |  +----------+             | Capabilities |    |
     |               |                           +--------------+    |
     |               |                                               |
     |      +--------+----+------------+-----------+--------+        |
     |      |             |            |           |        |        |
     | +----+-----+ +-----+----+ +-----+----+ +----+-----+  |        |
     | | SYNFlood | | UDPFlood | |ICMPFlood | | WebFlood |  |        |
     | |Capability| |Capability| |Capability| |Capability|  |        |
     | +----------+ +----------+ +----------+ +----------+  |        |
     |                                                      |        |
     |         +-----------------+--------------+-----------+        |
     |         |                 |              |                    |
     | +-------+-------+ +-------+------+ +-----+-----+ +-----+----+ |
     | |IPFragmentFlood| |DNSAmplication| |PingOfDeath| | IPSweep  | |
     | |  Capability   | |  Capability  | |Capability | |Capability| |
     | +---------------+ +--------------+ +-----------+ +----------+ |
     +---------------------------------------------------------------+
```

Figure 9. Attack Mitigation Capability Information Model

The detailed description about a standard interface, and the parameters for all the security Capabilities of this category, will be defined in a future version of this document.

5. Security Considerations

The security Capability policy information sent to NSFs should be protected by a secure communication channel, to ensure its confidentiality and integrity. Note that the NSFs and security controller can all be spoofed, which leads to undesirable results (e.g., security policy leakage from security controller, or a spoofed security controller sending false information to mislead the NSFs). Hence, mutual authentication MUST be supported to protected against this kind of threat. The current mainstream security technologies (i.e., TLS, DTLS, and IPSEC) can be employed to protect against the above threats.

In addition, to defend against DDoS attacks caused by a hostile security controller sending too many configuration messages to the NSFs, rate limiting or similar mechanisms should be considered.

6. IANA Considerations

TBD

7. Contributors

The following people contributed to creating this document, and are listed below in alphabetical order:

Antonio Lioy (Politecnico di Torino)
Dacheng Zhang (Huawei)
Edward Lopez (Fortinet)
Fulvio Valenza (Politecnico di Torino)
Kepeng Li (Alibaba)
Luyuan Fang (Microsoft)
Nicolas Bouthors (QoSmos)

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.

[RFC3539]
Aboba, B., and Wood, J., "Authentication, Authorization, and Accounting (AAA) Transport Profile", RFC 3539, June 2003.

8.2. Informative References

[RFC2975]
Aboba, B., et al., "Introduction to Accounting Management", RFC 2975, October 2000.

[I-D.draft-ietf-i2nsf-problem-and-use-cases]
Hares, S., et.al., "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-16, May 2017.

[I-D.draft-ietf-i2nsf-framework]
Lopez, E., et.al., "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-06, July, 2017.

[I-D.draft-ietf-i2nsf-terminology]
Hares, S., et.al., "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03, March, 2017

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

[Alshaer]
Al Shaer, E. and H. Hamed, "Modeling and management of firewall policies", 2004.

[Bas12]
Basile, C., Cappadonia, A., and A. Lioy, "Network-Level Access Control Policy Analysis and Transformation", 2012.

[Bas15]
Basile, C. and Lioy, A., "Analysis of application-layer filtering policies with application to HTTP", IEEE/ACM Transactions on Networking, Vol 23, Issue 1, February 2015.

[Cormen]
Cormen, T., "Introduction to Algorithms", 2009.

[Hohpe]
Hohpe, G. and Woolf, B., "Enterprise Integration Patterns", Addison-Wesley, 2003, ISBN 0-32-120068-3

[Lunt]
van Lunteren, J. and T. Engbersen, "Fast and scalable packet classification", IEEE Journal on Selected Areas in Communication, vol 21, Issue 4, September 2003.

[Martin]
Martin, R.C., "Agile Software Development, Principles, Patterns, and Practices", Prentice-Hall, 2002, ISBN: 0-13-597444-5

[OODMP]
http://www.oodesign.com/mediator-pattern.html

[OODOP]
http://www.oodesign.com/observer-pattern.html

[OODSRP]
http://www.oodesign.com/single-responsibility-principle.html

Appendix A. Network Security Capability Policy Rule Definitions

Six exemplary Network Security Capability Policy Rules are introduced in this Appendix to clarify how to create different kinds of specific ECA policy rules to manage Network Security Capabilities.

Note that there is a common pattern that defines how these ECAPolicyRules operate; this simplifies their implementation. All of these six ECA Policy Rules are concrete classes.

In addition, none of these six subclasses define attributes. This enables them to be viewed as simple object containers, and hence, applicable to a wide variety of content. It also means that the content of the function (e.g., how an entity is authenticated, what specific traffic is inspected, or which particular signature is applied) is defined solely by the set of events, conditions, and actions that are contained by the particular subclass. This enables the policy rule, with its aggregated set of events, conditions, and actions, to be treated as a reusable object.

A.1. AuthenticationECAPolicyRule Class Definition

The purpose of an AuthenticationECAPolicyRule is to define an I2NSF ECA Policy Rule that can verify whether an entity has an attribute of a specific value. A high-level conceputal figure is shown below.

```
                                                    +----------------+
   +----------------+ 1..n                  1...n |                |
   |                |/ \  HasAuthenticationMethod  \| Authentication |
   | Authentication + A ----------+-----------------+     Method     |
   | ECAPolicyRule  |\ /          ^                /|                |
   |                |             |                 +----------------+
   +----------------+             |
                                  |
                     +------------+-------------+
                     | AuthenticationRuleDetail |
                     +------------+-------------+
                                 / \ 0..n
                                  |
                                  | PolicyControlsAuthentication
                                  |
                                 / \
                                  A
                                 \ / 0..n
                       +----------+-------------+
                       | ManagementECAPolicyRule |
                       +-------------------------+
```

Figure 10. Modeling Authentication Mechanisms

This class does NOT define the authentication method used. This is because this would effectively "enclose" this information within the AuthenticationECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Authentication class(es) could not; they would have to associate with the AuthenticationECAPolicyRule class, and those other classes would not likely be interested in the AuthenticationECAPolicyRule. Second, the evolution of new authentication methods should be independent of the AuthenticationECAPolicyRule; this cannot happen if the Authentication class(es) are embedded in the AuthenticationECAPolicyRule.

This document only defines the AuthenticationECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 10 defines an aggregation between an external class, which defines one or more authentication methods, and an AuthenticationECAPolicyRule. This decouples the implementation of authentication mechanisms from how authentication mechanisms are managed and used.

Since different AuthenticationECAPolicyRules can use different authentication mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AuthenticationRuleDetail) to be used to define how a given AuthenticationMethod is used by a particular AuthenticationECAPolicyRule.

Similarly, the PolicyControlsAuthentication aggregation defines Policy Rules to control the configuration of the AuthenticationRuleDetail association class. This enables the entire authentication process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AuthenticationECAPolicyRule class (e.g., called authenticationMethodCurrent and authenticationMethodSupported), to represent the HasAuthenticationMethod aggregation and its association class. The former would be a string attribute that defines the current authentication method used by this AuthenticationECAPolicyRule, while the latter would define a set of authentication methods, in the form of an authentication Capability, which this AuthenticationECAPolicyRule can advertise.

A.2. AuthorizationECAPolicyRuleClass Definition

The purpose of an AuthorizationECAPolicyRule is to define an I2NSF ECA Policy Rule that can determine whether access to a resource should be given and, if so, what permissions should be granted to the entity that is accessing the resource.

This class does NOT define the authorization method(s) used. This is because this would effectively "enclose" this information within the AuthorizationECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Authorization class(es) could not; they would have to associate with the AuthorizationECAPolicyRule class, and those other classes would not likely be interested in the AuthorizationECAPolicyRule. Second, the evolution of new authorization methods should be independent of the AuthorizationECAPolicyRule; this cannot happen if the Authorization class(es) are embedded in the AuthorizationECAPolicyRule. Hence, this document recommends the following design:

```
                                                 +----------------+
 +----------------+ 1..n                   1...n |                |
 |                |/ \   HasAuthorizationMethod \| Authorization  |
 | Authorization  + A ----------+----------------+     Method     |
 | ECAPolicyRule  |\ /          ^               /|                |
 |                |             |                +----------------+
 +----------------+             |
                                |
                   +------------+-----------+
                   | AuthorizationRuleDetail |
                   +------------+-----------+
                               / \ 0..n
                                |
                                | PolicyControlsAuthorization
                                |
                               / \
                                A
                               \ / 0..n
                     +----------+--------------+
                     | ManagementECAPolicyRule |
                     +-------------------------+
```

Figure 11. Modeling Authorization Mechanisms

This document only defines the AuthorizationECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 11 defines an aggregation between the AuthorizationECAPolicyRule and an external class that defines one or more authorization methods. This decouples the implementation of authorization mechanisms from how authorization mechanisms are managed and used.

Since different AuthorizationECAPolicyRules can use different authorization mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AuthorizationRuleDetail) to be used to define how a given AuthorizationMethod is used by a particular AuthorizationECAPolicyRule.

Similarly, the PolicyControlsAuthorization aggregation defines Policy Rules to control the configuration of the AuthorizationRuleDetail association class. This enables the entire authorization process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AuthorizationECAPolicyRule class, called (for example) authorizationMethodCurrent and authorizationMethodSupported, to represent the HasAuthorizationMethod aggregation and its association class. The former is a string attribute that defines the current authorization method used by this AuthorizationECAPolicyRule, while the latter defines a set of authorization methods, in the form of an authorization Capability, which this AuthorizationECAPolicyRule can advertise.

A.3. AccountingECAPolicyRuleClass Definition

The purpose of an AccountingECAPolicyRule is to define an I2NSF ECA Policy Rule that can determine which information to collect, and how to collect that information, from which set of resources for the purpose of trend analysis, auditing, billing, or cost allocation [RFC2975] [RFC3539].

This class does NOT define the accounting method(s) used. This is because this would effectively "enclose" this information within the AccountingECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Accounting class(es) could not; they would have to associate with the AccountingECAPolicyRule class, and those other classes would not likely be interested in the AccountingECAPolicyRule. Second, the evolution of new accounting methods should be independent of the AccountingECAPolicyRule; this cannot happen if the Accounting class(es) are embedded in the AccountingECAPolicyRule. Hence, this document recommends the following design:

```
                                                   +-------------+
      +----------------+ 1..n                 1...n |             |
      |                |/ \   HasAccountingMethod  \| Accounting  |
      |  Accounting    + A ----------+--------------+   Method    |
      | ECAPolicyRule  |\ /          ^             /|             |
      |                |             |              +-------------+
      +----------------+             |
                                     |
                          +----------+----------+
                          | AccountingRuleDetail |
                          +----------+----------+
                                    / \ 0..n
                                     |
                                     | PolicyControlsAccounting
                                     |
                                    / \
                                     A
                                    \ / 0..n
                          +----------+-------------+
                          | ManagementECAPolicyRule |
                          +-------------------------+
```

Figure 12. Modeling Accounting Mechanisms

This document only defines the AccountingECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 12 defines an aggregation between the AccountingECAPolicyRule and an external class that defines one or more accounting methods. This decouples the implementation of accounting mechanisms from how accounting mechanisms are managed and used.

Since different AccountingECAPolicyRules can use different accounting mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AccountingRuleDetail) to be used to define how a given AccountingMethod is used by a particular AccountingECAPolicyRule.

Similarly, the PolicyControlsAccounting aggregation defines Policy Rules to control the configuration of the AccountingRuleDetail association class. This enables the entire accounting process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AccountingECAPolicyRule class, called (for example) accountingMethodCurrent and accountingMethodSupported, to represent the HasAccountingMethod aggregation and its association class.

The former is a string attribute that defines the current accounting method used by this AccountingECAPolicyRule, while the latter defines a set of accounting methods, in the form of an accounting Capability, which this AccountingECAPolicyRule can advertise.

A.4. TrafficInspectionECAPolicyRuleClass Definition

The purpose of a TrafficInspectionECAPolicyRule is to define an I2NSF ECA Policy Rule that, based on a given context, can determine which traffic to examine on which devices, which information to collect from those devices, and how to collect that information.

This class does NOT define the traffic inspection method(s) used. This is because this would effectively "enclose" this information within the TrafficInspectionECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the TrafficInspection class(es) could not; they would have to associate with the TrafficInspectionECAPolicyRule class, and those other classes would not likely be interested in the TrafficInspectionECAPolicyRule. Second, the evolution of new traffic inspection methods should be independent of the TrafficInspectionECAPolicyRule; this cannot happen if the TrafficInspection class(es) are embedded in the TrafficInspectionECAPolicyRule. Hence, this document recommends the following design:

```
                                               +------------------+
   +-------------------+1..n                1..n|                  |
   |                   |/ \ HasTrafficInspection \|      Traffic     |
   | TrafficInspection + A ----------+-------------+ InspectionMethod |
   |   ECAPolicyRule   |\ /          ^           / |                  |
   |                   |             |            +------------------+
   +-------------------+             |
                                     |
                        +------------+-----------+
                        | TrafficInspectionDetail |
                        +------------+-----------+
                                   / \ 0..n
                                    |
                                    | PolicyControlsTrafficInspection
                                    |
                                   / \
                                    A
                                   \ / 0..n
                          +----------+-------------+
                          | ManagementECAPolicyRule |
                          +------------------------+
```

Figure 13. Modeling Traffic Inspection Mechanisms

This document only defines the TrafficInspectionECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 13 defines an aggregation between the TrafficInspectionECAPolicyRule and an external class that defines one or more traffic inspection mechanisms. This decouples the implementation of traffic inspection mechanisms from how traffic inspection mechanisms are managed and used.

Since different TrafficInspectionECAPolicyRules can use different traffic inspection mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., TrafficInspectionDetail) to be used to define how a given TrafficInspectionMethod is used by a particular TrafficInspectionECAPolicyRule.

Similarly, the PolicyControlsTrafficInspection aggregation defines Policy Rules to control the configuration of the TrafficInspectionDetail association class. This enables the entire traffic inspection process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the TrafficInspectionECAPolicyRule class, called (for example) trafficInspectionMethodCurrent and trafficInspectionMethodSupported, to represent the HasTrafficInspectionMethod aggregation and its association class. The former is a string attribute that defines the current traffic inspection method used by this TrafficInspectionECAPolicyRule, while the latter defines a set of traffic inspection methods, in the form of a traffic inspection Capability, which this TrafficInspectionECAPolicyRule can advertise.

A.5. ApplyProfileECAPolicyRuleClass Definition

The purpose of an ApplyProfileECAPolicyRule is to define an I2NSF ECA Policy Rule that, based on a given context, can apply a particular profile to specific traffic. The profile defines the security Capabilities for content security control and/or attack mitigation control; these will be described in sections 4.4 and 4.5, respectively.

This class does NOT define the set of Profiles used. This is because this would effectively "enclose" this information within the ApplyProfileECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Profile class(es) could not; they would have to associate with the

ApplyProfileECAPolicyRule class, and those other classes would not likely be interested in the ApplyProfileECAPolicyRule. Second, the evolution of new Profile classes should be independent of the ApplyProfileECAPolicyRule; this cannot happen if the Profile class(es) are embedded in the ApplyProfileECAPolicyRule. Hence, this document recommends the following design:

```
                                                       +-------------+
      +-------------------+ 1..n                  1..n |             |
      |                   |/ \  ProfileApplied        \|             |
      | ApplyProfile      + A -----------+-------------+   Profile   |
      |   ECAPolicyRule   |\ /           ^            /|             |
      |                   |              |             +-------------+
      +-------------------+              |
                                         |
                            +------------+---------+
                            | ProfileAppliedDetail |
                            +------------+---------+
                                        / \ 0..n
                                         |
                                         |
        PolicyControlsProfileApplication |
                                         |
                                        / \
                                         A
                                        \ / 0..n
                              +----------+-------------+
                              | ManagementECAPolicyRule |
                              +-------------------------+
```

Figure 14. Modeling Profile ApplicationMechanisms

This document only defines the ApplyProfileECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 14 defines an aggregation between the ApplyProfileECAPolicyRule and an external Profile class. This decouples the implementation of Profiles from how Profiles are used.

Since different ApplyProfileECAPolicyRules can use different Profiles in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., ProfileAppliedDetail) to be used to define how a given Profile is used by a particular ApplyProfileECAPolicyRule.

Similarly, the PolicyControlsProfileApplication aggregation defines policies to control the configuration of the ProfileAppliedDetail association class. This enables the application of Profiles to be managed and used by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the ApplyProfileECAPolicyRuleclass, called (for example) profileAppliedCurrent and profileAppliedSupported, to represent the ProfileApplied aggregation and its association class. The former is a string attribute that defines the current Profile used by this ApplyProfileECAPolicyRule, while the latter defines a set of Profiles, in the form of a Profile Capability, which this ApplyProfileECAPolicyRule can advertise.

A.6. ApplySignatureECAPolicyRuleClass Definition

The purpose of an ApplySignatureECAPolicyRule is to define an I2NSF ECA Policy Rule that, based on a given context, can determine which Signature object (e.g., an anti-virus file, or aURL filtering file, or a script) to apply to which traffic. The Signature object defines the security Capabilities for content security control and/or attack mitigation control; these will be described in sections 4.4 and 4.5, respectively.

This class does NOT define the set of Signature objects used. This is because this would effectively "enclose" this information within the ApplySignatureECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Signature object class(es) could not; they would have to associate with the ApplySignatureECAPolicyRule class, and those other classes would not likely be interested in the ApplySignatureECAPolicyRule. Second, the evolution of new Signature object classes should be independent of the ApplySignatureECAPolicyRule; this cannot happen if the Signature object class(es) are embedded in the ApplySignatureECAPolicyRule. Hence, this document recommends the following design:

This document only defines the ApplySignatureECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 15 defines an aggregation between the ApplySignatureECAPolicyRule and an external Signature object class. This decouples the implementation of signature objects from how Signature objects are used.

Since different ApplySignatureECAPolicyRules can use different Signature objects in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., SignatureAppliedDetail) to be used to define how a given Signature object is used by a particular ApplySignatureECAPolicyRule.

```
                                             +-------------+
    +---------------+ 1..n                1..n |             |
    |               |/ \   SignatureApplied     \|             |
    | ApplySignature+ A ----------+--------------+  Signature  |
    | ECAPolicyRule |\ /          ^             /|             |
    |               |             |              +-------------+
    +---------------+             |
                                  |
                     +------------+----------+
                     | SignatureAppliedDetail |
                     +------------+----------+
                                / \ 0..n
                                 |
                                 | PolicyControlsSignatureApplication
                                 |
                                / \
                                 A
                                \ / 0..n
                       +----------+-------------+
                       | ManagementECAPolicyRule |
                       +-------------------------+
```

Figure 15. Modeling Sginature Application Mechanisms

Similarly, the PolicyControlsSignatureApplication aggregation defines policies to control the configuration of the SignatureAppliedDetail association class. This enables the application of the Signature object to be managed by policy.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the ApplySignatureECAPolicyRule class, called (for example) signature signatureAppliedCurrent and signatureAppliedSupported, to represent the SignatureApplied aggregation and its association class. The former is a string attribute that defines the current Signature object used by this ApplySignatureECAPolicyRule, while the latter defines a set of Signature objects, in the form of a Signature Capability, which this ApplySignatureECAPolicyRule can advertise.

Appendix B. Network Security Event Class Definitions

This Appendix defines a preliminary set of Network Security Event classes, along with their attributes.

B.1. UserSecurityEvent Class Description

The purpose of this class is to represent Events that are initiated by a user, such as logon and logoff Events. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include user identification data and the type of connection used by the user.

The UserSecurityEvent class defines the following attributes.

B.1.1. The usrSecEventContent Attribute

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 the string "hrAdmin", with the usrSecEventFormat set to 1 (GUID) and the usrSecEventType attribute set to 5 (new logon).

B.1.2. The usrSecEventFormat Attribute

This is a mandatory non-negative 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 the string "hrAdmin", with the usrSecEventFormat attribute set to 1 (GUID) and the usrSecEventType attribute set to 5 (new logon). Values include:

0: unknown
1: GUID (Generic Unique IDentifier)
2: UUID (Universal Unique IDentifier)
3: URI (Uniform Resource Identifier)
4: FQDN (Fully Qualified Domain Name)
5: FQPN (Fully Qualified Path Name)

B.1.3. The usrSecEventType Attribute

This is a mandatory non-negative 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 the string "hrAdmin", with the usrSecEventFormat attribute set to 1 (GUID), and the usrSecEventType attribute set to 5 (new logon). Values include:

0: unknown
1: new user created
2: new user group created
3: user deleted
4: user group deleted
5: user logon
6: user logoff
7: user access request
8: user access granted
9: user access violation

B.2. DeviceSecurityEvent Class Description

The purpose of a DeviceSecurityEvent is to represent Events that provide information from the Device that are important to I2NSF Security. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include alarms and various device statistics (e.g., a type of threshold that was exceeded), which may signal the need for further action.

The DeviceSecurityEvent class defines the following attributes.

B.2.1. The devSecEventContent Attribute

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).

B.2.2. The devSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the devSecEventContent attribute. Values include:

0: unknown
1: GUID (Generic Unique IDentifier)
2: UUID (Universal Unique IDentifier)
3: URI (Uniform Resource Identifier)
4: FQDN (Fully Qualified Domain Name)
5: FQPN (Fully Qualified Path Name)

B.2.3. The devSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that was generated by this device. Values include:

0: unknown
1: communications alarm
2: quality of service alarm
3: processing error alarm
4: equipment error alarm
5: environmental error alarm

Values 1-5 are defined in X.733. Additional types of errors may also be defined.

B.2.4. The devSecEventTypeInfo[0..n] Attribute

This is an optional array of strings, which is used to provide additional information describing the specifics of the Event generated by this Device. For example, this attribute could contain probable cause information in the first array, trend information in the second array, proposed repair actions in the third array, and additional information in the fourth array.

B.2.5. The devSecEventTypeSeverity Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the perceived severity of the Event generated by this Device. Values (which are defined in X.733) include:

0: unknown
1: cleared
2: indeterminate
3: critical
4: major
5: minor
6: warning

B.3. SystemSecurityEvent Class Description

The purpose of a SystemSecurityEvent is to represent Events that are detected by the management system, instead of Events that are generated by a user or a device. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include an event issued by an analytics system that warns against a particular pattern of unknown user accesses, or an Event issued by a management system that represents a set of correlated and/or filtered Events.

The SystemSecurityEvent class defines the following attributes.

B.3.1. The sysSecEventContent Attribute

This is a mandatory string that contains the content of the SystemSecurityEvent. The format of the content is specified in the sysSecEventFormat class attribute, and the type of Event is defined in the sysSecEventType class attribute. An example of the sysSecEventContent attribute is the string "sysadmin3", with the sysSecEventFormat attribute set to 1 (GUID), and the sysSecEventType attribute set to 2 (audit log cleared).

B.3.2. The sysSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the sysSecEventContent attribute. Values include:

0: unknown
1: GUID (Generic Unique IDentifier)
2: UUID (Universal Unique IDentifier)
3: URI (Uniform Resource Identifier)
4: FQDN (Fully Qualified Domain Name)
5: FQPN (Fully Qualified Path Name)

B.3.3. The sysSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that involves this device. Values include:

0: unknown
1: audit log written to
2: audit log cleared
3: policy created
4: policy edited
5: policy deleted
6: policy executed

B.4. TimeSecurityEvent Class Description

The purpose of a TimeSecurityEvent is to represent Events that are temporal in nature (e.g., the start or end of a period of time). Time events signify an individual occurrence, or a time period, in which a significant event happened. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include issuing an Event at a specific time to indicate that a particular resource should not be accessed, or that different authentication and authorization mechanisms should now be used (e.g., because it is now past regular business hours).

The TimeSecurityEvent class defines the following attributes.

B.4.1. The timeSecEventPeriodBegin Attribute

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).

B.4.2. The timeSecEventPeriodEnd Attribute

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 occurence, and not a time period when the Event can occur, then the timeSecEventPeriodEnd attribute may be ignored.

B.4.3. The timeSecEventTimeZone Attribute

This is a mandatory string attribute, and defines the time zone that this Event occurred in using the format specified in ISO8601.

Appendix C. Network Security Condition Class Definitions

This Appendix defines a preliminary set of Network Security Condition classes, along with their attributes.

C.1. PacketSecurityCondition

The purpose of this Class is to represent 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. This class is abstract, and serves as the superclass of more detailed conditions that act on different types of packet formats. Its subclasses are shown in Figure 16, and are defined in the following sections.

```
                           +-------------------------+
                           | PacketSecurityCondition |
                           +------------+------------+
                                       / \
                                        |
                                        |
               +---------+----------+---+-----+----------+
               |         |          |         |          |
               |         |          |         |          |
      +--------+-------+ | +--------+-------+ | +--------+-------+
      | PacketSecurity | | | PacketSecurity | | | PacketSecurity |
      |  MACCondition  | | | IPv4Condition  | | | IPv6Condition  |
      +----------------+ | +----------------+ | +----------------+
                         |                    |
                +--------+-------+   +--------+-------+
                |  TCPCondition  |   |  UDPCondition  |
                +----------------+   +----------------+
```

Figure 16. Network Security Info Sub-Model PacketSecurityCondition
           Class Extensions

C.1.1. PacketSecurityMACCondition

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 executed or not. This class is concrete, and defines the following attributes.

C.1.1.1. The pktSecCondMACDest Attribute

This is a mandatory string attribute, and defines the MAC destination address (6 octets long).

C.1.1.2. The pktSecCondMACSrc Attribute

This is a mandatory string attribute, and defines the MAC source address (6 octets long).

C.1.1.3. The pktSecCondMAC8021Q Attribute

This is an optional string attribute, and defines the 802.1Q tag value (2 octets long). This defines VLAN membership and 802.1p priority values.

C.1.1.4. The pktSecCondMACEtherType Attribute

This is a mandatory string attribute, and defines 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.

C.1.1.5. The pktSecCondMACTCI Attribute

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).

C.1.2. PacketSecurityIPv4Condition

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. This class is concrete, and defines the following attributes.

C.1.2.1. The pktSecCondIPv4SrcAddr Attribute

This is a mandatory string attribute, and defines the IPv4 Source Address (32 bits).

C.1.2.2. The pktSecCondIPv4DestAddr Attribute

This is a mandatory string attribute, and defines the IPv4 Destination Address (32 bits).

C.1.2.3. The pktSecCondIPv4ProtocolUsed Attribute

This is a mandatory string attribute, and defines the protocol used in the data portion of the IP datagram (8 bits).

C.1.2.4. The pktSecCondIPv4DSCP Attribute

This is a mandatory string attribute, and defines the Differentiated Services Code Point field (6 bits).

C.1.2.5. The pktSecCondIPv4ECN Attribute

This is an optional string attribute, and defines the Explicit Congestion Notification field (2 bits).

C.1.2.6. The pktSecCondIPv4TotalLength Attribute

This is a mandatory string attribute, and defines the total length of the packet (including header and data) in bytes (16 bits).

C.1.2.7. The pktSecCondIPv4TTL Attribute

This is a mandatory string attribute, and defines the Time To Live in seconds (8 bits).

C.1.3. PacketSecurityIPv6Condition

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. This class is concrete, and defines the following attributes.

C.1.3.1. The pktSecCondIPv6SrcAddr Attribute

This is a mandatory string attribute, and defines the IPv6 Source Address (128 bits).

C.1.3.2. The pktSecCondIPv6DestAddr Attribute

This is a mandatory string attribute, and defines the IPv6 Destination Address (128 bits).

C.1.3.3. The pktSecCondIPv6DSCP Attribute

This is a mandatory string attribute, and defines the Differentiated Services Code Point field (6 bits). It consists of the six most significant bits of the Traffic Class field in the IPv6 header.

C.1.3.4. The pktSecCondIPv6ECN Attribute

This is a mandatory string attribute, and defines the Explicit Congestion Notification field (2 bits). It consists of the two least significant bits of the Traffic Class field in the IPv6 header.

C.1.3.5. The pktSecCondIPv6FlowLabel Attribute

This is a mandatory string attribute, and defines an IPv6 flow label. This, in combination with the Source and Destination Address fields, enables efficient IPv6 flow classification by using only the IPv6 main header fields (20 bits).

C.1.3.6. The pktSecCondIPv6PayloadLength Attribute

This is a mandatory string attribute, and defines the total length of the packet (including the fixed and any extension headers, and data) in bytes (16 bits).

C.1.3.7. The pktSecCondIPv6NextHeader Attribute

This is a mandatory string attribute, and defines the type of the next header (e.g., which extension header to use) (8 bits).

C.1.3.8. The pktSecCondIPv6HopLimit Attribute

This is a mandatory string attribute, and defines the maximum number of hops that this packet can traverse (8 bits).

C.1.4. PacketSecurityTCPCondition

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. This class is concrete, and defines the following attributes.

C.1.4.1. The pktSecCondTPCSrcPort Attribute

This is a mandatory string attribute, and defines the Source Port number (16 bits).

C.1.4.2. The pktSecCondTPCDestPort Attribute

This is a mandatory string attribute, and defines the Destination Port number (16 bits).

C.1.4.3. The pktSecCondTCPSeqNum Attribute

This is a mandatory string attribute, and defines the sequence number (32 bits).

C.1.4.4. The pktSecCondTCPFlags Attribute

This is a mandatory string attribute, and defines the nine Control bit flags (9 bits).

C.1.5. PacketSecurityUDPCondition

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. This class is concrete, and defines the following attributes.

C.1.5.1.1. The pktSecCondUDPSrcPort Attribute

This is a mandatory string attribute, and defines the UDP Source Port number (16 bits).

C.1.5.1.2. The pktSecCondUDPDestPort Attribute

This is a mandatory string attribute, and defines the UDP Destination Port number (16 bits).

C.1.5.1.3. The pktSecCondUDPLength Attribute

This is a mandatory string attribute, and defines the length in bytes of the UDP header and data (16 bits).

C.2. PacketPayloadSecurityCondition

The purpose of this Class is to represent packet payload data 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. Examples include a specific set of bytes in the packet payload.

C.3. TargetSecurityCondition

The purpose of this Class is to represent information about different targets of this policy (i.e., entities to which this Policy Rule should be applied), which 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. Examples include whether the targeted entities are playing the same role, or whether each device is administered by the same set of users, groups, or roles. This Class has several important subclasses, including:

a. ServiceSecurityContextCondition is the superclass for all information that can be used in an ECA Policy Rule that specifies data about the type of service to be analyzed (e.g., the protocol type and port number)
b. ApplicationSecurityContextCondition is the superclass for all information that can be used in a ECA Policy Rule that specifies data that identifies a particular application (including metadata, such as risk level)
c. DeviceSecurityContextCondition is the superclass for all information that can be used in a ECA Policy Rule that specifies data about a device type and/or device OS that is being used

C.4. UserSecurityCondition

The purpose of this Class is to represent data about the user or group referenced in this ECA Policy Rule that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include the user or group id used, the type of connection used, whether a given user or group is playing a particular role, or whether a given user or group has failed to login a particular number of times.

C.5. SecurityContextCondition

The purpose of this Class is to represent security conditions that are part of a specific context, which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include testing to determine if a particular pattern of security-related data have occurred, or if the current session state matches the expected session state.

C.6. GenericContextSecurityCondition

The purpose of this Class is to represent generic contextual information in which this ECA Policy Rule is being executed, which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include geographic location and temporal information.

Appendix D. Network Security Action Class Definitions

This Appendix defines a preliminary set of Network Security Action classes, along with their attributes.

D.1. IngressAction

The purpose of this Class is to represent actions performed on packets that enter an NSF. Examples include pass, dropp, or mirror traffic.

D.2. EgressAction

The purpose of this Class is to represent actions performed on packets that exit an NSF. Examples include pass, drop, or mirror traffic, signal, and encapsulate.

D.3. ApplyProfileAction

The purpose of this Class is to define the application of a profile to packets to perform content security and/or attack mitigation control.

Appendix E. Geometric Model

The geometric model defined in [Bas12] is summarized here. Note that our work has extended the work of [Bas12] to model ECA Policy Rules, instead of just condition-action Policy Rules. However, the geometric model in this Appendix is simplified in this version of this I-D, and is used to define just the CA part of the ECA model.

All the actions available to the security function are well known and organized in an action set A.

For filtering controls, the enforceable actions are either Allow or Deny, thus A={Allow,Deny}. For channel protection controls, they may be informally written as "enforce confidentiality", "enforce data authentication and integrity", and "enforce confidentiality and data authentication and integrity". However, these actions need to be instantiated to the technology used. For example, AH-transport mode and ESP-transport mode (and combinations thereof) are a more precise definition of channel protection actions.

Conditions are typed predicates concerning a given selector. A selector describes the values that a protocol field may take. For example, the IP source selector is the set of all possible IP addresses, and it may also refer to the part of the packet where the values come from (e.g., the IP source selector refers to the IP source field in the IP header). Geometrically, a condition is the subset of its selector for which it evaluates to true. A condition on a given selector matches a packet if the value of the field referred to by the selector belongs to the condition. For instance, in Figure 17 the conditions are s1 <= S1 (read as s1 subset of or equal to S1) and s2 <= S2 (s2 subset of or equal to S2), both s1 and s2 match the packet x1, while only s2 matches x2.

To consider conditions in different selectors, the decision space is extended using the Cartesian product because distinct selectors refer to different fields, possibly from different protocol headers. Hence, given a policy-enabled element that allows the definition of conditions on the selectors S1, S2,..., Sm (where m is the number of Selectors available at the security control we want to model), its selection space is:

    S=S1 X S2 X ... X Sm

To consider conditions in different selectors, the decision space is extended using the Cartesian product because distinct selectors refer to different fields, possibly from different protocol headers.

```
        S2 ^ Destination port
           |
           |      x2
           +......o
           |      .
           |      .
         --+.............+------------------------------------+
         | |      .      |                                    |
         s |      .      |                                    |
         e |      .      |            (rectangle)             |
         g |      .      |        condition clause (c)        |
         m |      .      |   here the action a is applied     |
         e |      .      |                                    |
         n |      .      |             x1=point=packet        |
         t +.............|.............o                      |
         | |      .      |             .                      |
         --+.............+------------------------------------+
           |      .      .             .                      .
           |      .      .             .                      .
     +------------+------+-------------+----------------------+------>
           |             |---- segment = condition in S1 -----|     S1
           +                                                 IP source
```

Figure 17: Geometric representation of a rule r=(c,a) that matches x1, but does not match x2.

Accordingly, the condition clause c is a subset of S:

c = s1 X s2 X ... X sm <= S1 X S2 X ... X Sm = S

S represents the totality of the packets that are individually selectable by the security control to model when we use it to enforce a policy. Unfortunately, not all its subsets are valid condition clauses: only hyper-rectangles, or the union of hyper-rectangles (as they are Cartesian product of conditions), are valid. This is an intrinsic constraint of the policy language, as it specifies rules by defining a condition for each selector. Languages that allow specification of conditions as relations over more fields are modeled by the geometric model as more complex geometric shapes determined by the equations. However, the algorithms to compute intersections are much more sophisticated than intersection hyper-rectangles. Figure 17 graphically represents a condition clause c in a two-dimensional selection space.

In the geometric model, a rule is expressed as r=(c,a), where c <= S (the condition clause is a subset of the selection space), and the action a belongs to A. Rule condition clauses match a packet (rules match a packet), if all the conditions forming the clauses match the packet. In Figure 17, the rule with condition clause c matches the packet x1 but not x2.

The rule set R is composed of n rules ri=(ci,ai).

The decision criteria for the action to apply when a packet matches two or more rules is abstracted by means of the resolution strategy

RS: Pow(R) -> A

where Pow(R) is the power set of rules in R.

Formally, given a set of rules, the resolution strategy maps all the possible subsets of rules to an action a in A. When no rule matches a packet, the security controls may select the default action d in A, if they support one.

Resolution strategies may use, besides intrinsic rule data (i.e., condition clause and action clause), also external data associated to each rule, such as priority, identity of the creator, and creation time. Formally, every rule ri is associated by means of the function e(.):

e(ri) = (ri,f1(ri),f2(ri),...)

where E={fj:R -> Xj} (j=1,2,...) is the set that includes all functions that map rules to external attributes in Xj. However, E, e, and all the Xj are determined by the resolution strategy used.

A policy is thus a function p: S -> A that connects each point of the selection space to an action taken from the action set A according to the rules in R. By also assuming RS(0)=d (where 0 is the empty-set) and RS(ri)=ai, the policy p can be described as:

p(x)=RS(match{R(x)}).

Therefore, in the geometric model, a policy is completely defined by the 4-tuple (R,RS,E,d): the rule set R, the resolution function RS, the set E of mappings to the external attributes, and the default action d.

Note that, the geometric model also supports ECA paradigms by simply modeling events like an additional selector.

Authors' Addresses
Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China
Email: Frank.xialiang@huawei.com

John Strassner
Huawei
Email: John.sc.Strassner@huawei.com

Cataldo Basile
Politecnico di Torino
Corso Duca degli Abruzzi, 34
Torino, 10129
Italy
Email: cataldo.basile@polito.it

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid,   28010
Spain
Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com