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.

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

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

5. 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:

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