Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 21, 2017

J. Jeong
J. Kim
D. Hyun
Sungkyunkwan University
J. Park
ETRI
T. Ahn
Korea Telecom

July 20, 2016

YANG Data Model of Interface to Network Security Functions Capability Interface
draft-jeong-i2nsf-capability-interface-yang-02

Abstract

This document defines a data model corresponding to the information model for Interface to Network Security Functions (I2NSF) capability interface (i.e., NSF facing interface). It describes a data model for three security capabilities (i.e., network security functions), such as network security control, content security control, and attack mitigation control, as defined in the information model for the NSF facing interface.

Status of This Memo

Table of Contents

1. Introduction   3
2. Requirements Language   3
3. Terminology   3
  3.1. Tree Diagrams   3
4. Model Structure   4
5. YANG Model 11
6. Security Considerations 47
7. Acknowledgements 47
8. References 47
  8.1. Normative References 47
  8.2. Informative References 47

1. Introduction

This document defines a YANG [RFC6020] model for security services with the information model of Interface to Network Security Functions (I2NSF) capability interface (i.e., NSF facing interface). It provides a specific information model and the corresponding data model for three security capabilities (i.e., network security functions), such as network security control, content security control, and attack mitigation control, as defined in [i2nsf-cap-interface-im].

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-cap-interface-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. Model Structure

Figure 1 shows an overview of a model structure tree of network security control, content security control, and attack mitigation control, as defined in the [i2nsf-cap-interface-im].

```
module : ietf-i2nsf-nsf-facing-interface
  +--rw policy
    +--rw policy-name  string
    +--rw policy-id  string
    +--rw rule  *[rule-id]
      +--rw rule-name  string
      +--rw rule-id  uint 8
      +--rw event
      |  +--rw time-event-list?  *[time-id]
      |  |  +--rw event-time?  yang:date-and-time
      |  |  +--rw time-id  uint 8
      |  +--rw user-action?
      |     +--rw login  boolean
      |     +--rw logoff  boolean
      |     +--rw originating  boolean
      |     +--rw terminating  boolean
      +--rw condition
      |  +--rw packet-content-values
      |  |  +--rw layer-234-packet-header *[packet-header-id]
      |  |  |  +--rw packet-header-id uint 8
      |  |  |  +--rw address-scope?
      |  |  |  |  +--rw (route-type)?
      |  |  |  |  |  +--: (ipv4)
      |  |  |  |  |  |  +--rw (ip-route-match-type)?
      |  |  |  |  |  |     +--: (dest-ipv4-address)
      |  |  |  |  |  |     |  +--rw dest-ipv4-prefix  inet:ipv4-prefix
      |  |  |  |  |  |     +--: (src-ipv4-address)
      |  |  |  |  |  |     |  +--rw src-ipv4-prefix  inet:ipv4-prefix
      |  |  |  |  |  |     +--: (dest-src-ipv4-address)
      |  |  |  |  |  |        +--rw dest-src-ipv4-address
      |  |  |  |  |  |           +--rw dest-ipv4-prefix inet:ipv4-prefix
      |  |  |  |  |  |           +--rw src-ipv4-prefix inet:ipv4-prefix
      |  |  |  |  |  +--: (ipv6)
      |  |  |  |  |  |  +--rw (ip-route-match-type)?
      |  |  |  |  |  |     +--: (dest-ipv6-address)
      |  |  |  |  |  |     |  +--rw dest-ipv6-prefix inet:ipv6-prefix
```

```
      |  |  |  |  |  |     +--: (src-ipv6-address)
      |  |  |  |  |  |     |  +--rw src-ipv6-prefix inet:ipv6-prefix
      |  |  |  |  |  |     +--: (dest-src-ipv6-address)
      |  |  |  |  |  |        +--rw dest-src-ipv6-address
      |  |  |  |  |  |           +--rw dest-ipv6-prefix inet:ipv6-prefix
      |  |  |  |  |  |           +--rw src-ipv6-prefix inet:ipv6-prefix
      |  |  |  |  |  +--: (mpls-route)
      |  |  |  |  |  |  +--rw mpls-label uint32
      |  |  |  |  |  +--: (mac-route)
      |  |  |  |  |  |  +--rw mac-address uint32
      |  |  |  |  |  +--: (interface-route)
      |  |  |  |  |  |  +--rw interface-identifier if:interface-ref
      |  |  |  +--rw (layer-type)?
      |  |  |     +--: (layer-2-header)
      |  |  |     |  +--rw src-mac-address yang:phys-address
      |  |  |     |  +--rw dest-mac-address yang:phys-address
      |  |  |     +--: (layer-3-header)
      |  |  |     |  +--rw (packet-type)?
      |  |  |     |    +--: (ipv4)
      |  |  |     |    |  +--rw src-ipv4-address inet:ipv4-address
      |  |  |     |    |  +--rw dest-ipv4-address inet:ipv4-address
      |  |  |     |    |  +--rw protocol uint 8
      |  |  |     |    |  +--rw ttl uint 8
      |  |  |     |    |  +--rw dscp uint 8
      |  |  |     |    +--: (ipv6)
      |  |  |     |    |  +--rw src-ipv6-address inet:ipv6-address
      |  |  |     |    |  +--rw dest-ipv6-address inet:ipv6-address
      |  |  |     |    |  +--rw next-header uint 8
      |  |  |     |    |  +--rw traffic-class uint 8
      |  |  |     |    |  +--rw flow-label uint 16
      |  |  |     |    |  +--rw hop-limit uint 8
      |  |  |     +--: (layer-4-header)
      |  |  |       +--rw src-port inet:port-number
      |  |  |       +--rw dest-port inet:port-number
      |  |  +--rw packet-payload *[packet-payload-id]
      |  |    +--rw packet-payload-id uint 8
      |  +--rw context-values
      |     +--rw user *[user-id]
      |     |  +--rw user-id uint 8
      |     |  +--rw (user-name)?
      |     |     +--: (tenant)
      |     |     |  +--rw tenant uint 8
      |     |     +--: (vn-id)
      |     |        +--rw vn-id uint 8
      |     +--rw schedule
      |     |  +--rw name string
      |     |  +--rw (schedule-type)?
      |     |  |  +--: (once)
```

```
      |     |  |  |  +--rw once boolean
      |     |  |  +--: (periodic)
      |     |  |  |  +--rw periodic boolean
      |     |  +--rw start-time? yang:date-and-time
      |     |  +--rw end-time? yang:date-and-time
      |     |  +--rw weekly-validity-time yang:date-and-time
      |     +--rw region *[ip-address-region]
      |     |  +--rw ip-address-region uint 8
      |     +--rw target
      |     |  +--rw service
      |     |  |  +--rw name string
      |     |  |  +--rw id uint 8
      |     |  |  +--rw protocol
      |     |  |  |  +--rw TCP? boolean
      |     |  |  |  +--rw UDP? boolean
      |     |  |  |  +--rw ICMP? boolean
      |     |  |  |  +--rw ICMPv6? boolean
      |     |  |  |  +--rw IP? boolean
      |     |  |  +--rw src-port? inet:port-number
      |     |  |  +--rw dest-port? inet:port-number
      |     |  +--rw application
      |     |     +--rw name string
      |     |     +--rw id uint 8
      |     |     +--rw category
      |     |     |  +--rw business-system?  boolean
      |     |     |  +--rw entertainment?  boolean
      |     |     |  +--rw internet?  boolean
      |     |     |  +--rw network?  boolean
      |     |     |  +--rw general?  boolean
      |     |     +--rw subcategory
      |     |     |  +--rw finance?  boolean
      |     |     |  +--rw email?  boolean
      |     |     |  +--rw game?  boolean
      |     |     |  +--rw media-sharing?  boolean
      |     |     |  +--rw social-network?  boolean
      |     |     |  +--rw web-posting?  boolean
      |     |     +--rw data-transmission-model
      |     |     |  +--rw client-server?  boolean
      |     |     |  +--rw browser-based?  boolean
      |     |     |  +--rw networking?  boolean
      |     |     |  +--rw peer-to-peer?  boolean
      |     |     |  +--rw unassigned?  boolean
      |     |     +--rw risk-level
      |     |        +--rw exploitable?  boolean
      |     |        +--rw productivity-loss?  boolean
      |     |        +--rw evasive?  boolean
      |     |        +--rw data-loss?  boolean
      |     |        +--rw malware-vehicle?  boolean
```

```
      |     |        +--rw bandwidth-consuming?  boolean
      |     |        +--rw tunneling?  boolean
      |     +--rw device
      |     |  +--rw pc?  boolean
      |     |  +--rw mobile-phone?  boolean
      |     |  +--rw tablet?  boolean
      |     |  +--rw voip-phone  boolean
      |     +--rw (state)?
      |     |  +--: (session-state)
      |     |  |  +--rw tcp-session-state
      |     |  |     +--rw new?  boolean
      |     |  |     +--rw established?  boolean
      |     |  |     +--rw related?  boolean
      |     |  |     +--rw invalid?  boolean
      |     |  |     +--rw untracked?  boolean
      |     |  +--: (session-aaa-state)
      |     |  |  +--rw session-sip-state
      |     |  |     +--rw auth-state?  boolean
      |     |  |     +--rw call-state?  boolean
      |     |  +--: (access-mode)
      |     |  |  +--rw access-mode  string
      |     +--rw direction
      |        +--rw request?  boolean
      |        +--rw response?  boolean
      +--rw action
         +--rw (action-type)?
            +--: (ingress-action)
            |  +--rw (ingress-action-type)?
            |    +--: (permit)
            |    |  +--rw permit boolean
            |    +--: (deny)
            |    |  +--rw deny boolan
            |    +--: (mirror)
            |       +--rw mirror boolean
            +--: (egress-action)
            |  +--rw (egress-action-type)?
            |    +--: (invoke-signaling)
            |    |  +--rw invoke-signaling boolean
            |    +--: (tunnel-encapsulation)
            |    |  +--rw tunnel-encapsulation boolean
            |    +--: (forwarding)
            |       +--rw forwarding  boolean
            +--: (advanced-action)
               +--rw (advanced-action-type)?
                 +--: (content-security-control)
                 |  +--rw (content-security-control-type)?
                 |     +--: (antivirus)
                 |     |  +--rw antivirus?  boolean
```

```
                |     +--: (ips)
                |     |  +--rw ips?  boolean
                |     +--: (url-filtering)
                |     |  +--rw url-filtering?  boolean
                |     +--: (file-blocking)
                |     |  +--rw file-blocking?  boolean
                |     +--: (data-filtering)
                |     |  +--rw data-filtering?  boolean
                |     +--: (application-control)
                |     |  +--rw application-control?  boolean
                |     +--: (voip-volte)
                |        +--rw voip-volte-rule  *[voip-volte-rule-id]
                |           +--rw voip-volte-rule-id  uint 8
                |           +--rw event
                |           |  +--rw called-voip  boolean
                |           |  +--rw called-volte  boolean
                |           +--rw condition
                |           |  +--rw sip-header?  *[sip-header-uri]
                |           |  |  +--rw sip-header-uri string
                |           |  |  +--rw sip-header-method string
                |           |  |  +--rw expire-time yang:date-and-time
                |           |  |  +--rw sip-header-user-agent uint32
                |           |  +--rw cell-region? *[cell-id-region]
                |           |     +--rw cell-id-region uint 32
                |           +--rw action
                |              +--rw (action-type)?
                |                 +--: (ingress-action)
                |                 |  +--rw (ingress-action-type)?
                |                 |    +--: (permit)
                |                 |    |  +--rw permit boolean
                |                 |    +--: (deny)
                |                 |    |  +--rw deny boolean
                |                 |    +--: (mirror)
                |                 |       +--rw mirror boolean
                |                 +--: (egress-action)
                |                    +--: (egress-action-type)?
                |                      +--: (redirection)
                |                         +--rw redirection? boolean
                +--: (attack-mitigation-control)
                   +--rw (attack-mitigation-control-type)?
                      +--: (ddos-attack)
                      |  +--rw (ddos-attack-type)?
                      |     +--: (network-layer-ddos-attack)
                      |     |  +--rw (network-layer-ddos-attack-type)?
                      |     |     +--: (syn-flood-attack)
                      |     |     |  +--rw syn-flood    boolean
                      |     |     +--: (udp-flood-attack)
                      |     |     |  +--rw udp-flood    boolean
```

```
                       |     |     +--: (icmp-flood-attack)
                       |     |     |  +--rw icmp-flood    boolean
                       |     |     +--: (ip-fragment-flood-attack)
                       |     |     |  +--rw ip-fragment-flood boolean
                       |     |     +--: (ipv6-related-attacks)
                       |     |        +--rw ipv6-related    boolean
                       |     +--: (app-layer-ddos-attack)
                       |        +--rw (app-layer-ddos-attack-type)?
                       |           +--: (http-flood-attack)
                       |           |  +--rw http-flood    boolean
                       |           +--: (https-flood-attack)
                       |           |  +--rw https-flood    boolean
                       |           +--: (dns-flood-attack)
                       |           |  +--rw dns-flood    boolean
                       |           +--: (dns-amp-flood-attack)
                       |           |  +--rw dns-amp-flood boolean
                       |           +--: (ssl-ddos-attack)
                       |              +--rw ssl-ddos    boolean
                       +--: (single-packet-attack)
                          +--rw (single-packet-attack-type)?
                             +--: (scan-and-sniff-attack)
                             |  +--rw (scan-and-sniff-attack-type)?
                             |  |  +--: (ip-sweep-attack)
                             |  |  |  +--rw ip-sweep    boolean
                             |  |  +--: (port-scanning-attack)
                             |  |  |  +--rw port-scanning    boolean
                             +--: (malformed-packet-attack)
                             |  +--rw (malformed-packet-attack-type)?
                             |  |  +--: (ping-of-death-attack)
                             |  |  |  +--rw ping-of-death    boolean
                             |  |  +--: (teardrop-attack)
                             |  |  |  +--rw teardrop    boolean
                             +--: (special-packet-attack)
                                +--rw (special-packet-attack-type)?
                                   +--: (oversized-icmp-attack)
                                   | +--rw oversized-icmp    boolean
                                   +--: (tracert-attack)
                                      +--rw tracert    boolean
```

Figure 1: Model Structure of NSF Facing Interface

```
   rpcs:
     +---x time-event-add
     |  +---w input
     |  |  +---w event-time?  yang:date-and-time
     |  |  +---w time-id  uint 8
     |  +--ro output
     |     +--ro result  boolean
     |     +--ro reason  string
     |
     +---x time-event-delete
     |  +---w input
     |  |  +---w time-id  uint 8
     |  +--ro output
     |     +--ro result  boolean
     |     +--ro reason  string
     |
     +---x user-add
     |  +---w input
     |  |  +---w user-id  uint 8
     |  |  +---w (user-name)?
     |  |  |  +--: (tenant)
     |  |  |  |  +---w tenant  uint 8
     |  |  |  +--: (vn-id)
     |  |  |    +---w vn-id  uint 8
     |  +--ro output
     |     +--ro result  boolean
     |     +--ro reason  string
     |
     +---x user-delete
     |  +---w input
     |  |  +---w user-id  uint 8
     |  +--ro output
     |     +--ro result  boolean
     |     +--ro reason  string
     |
     +---x region-add
     |  +---w input
     |  |  +---w ip-address-region  uint 8
     |  +--ro output
     |     +--ro result  boolean
     |     +--ro reason  string
     |
     +---x region-delete
        +---w input
        |  +---w ip-address-region  uint 8
        +--ro output
           +--ro result  boolean
           +--ro reason  string
```

Figure 2: Model of Remote Procedure Calls

5. YANG Model

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

```
<CODE BEGINS> file "ietf-i2nsf-nsf-facing-interface@2016-07-20.yang"

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

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

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

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

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

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

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

  description
    "This module defines a YANG data module for network security
     functions.";
  revision "2016-07-20"{
    description "Initial revision";
```

```
    reference
      "draft-xia-i2nsf-capability-interface-im-05";
  }

  //Groupings
  grouping policy {
    description
      "policy is a grouping
       including a set of security rules according to certain logic,
       i.e., their similarity or mutual relations, etc. The network
       security policy is able to apply over both the unidirectional
       and bidirectional traffic across the NSF.";

      leaf policy-name {
        type string;
        mandatory true;
        description
          "The name of the policy.
           This must be unique.";
      }
      leaf policy-id {
        type string;
        mandatory true;
        description
          "The ID of the policy.
           This must be unique.";
      }

      list rule {
        key "rule-id";
        description
          "This is a rule for network security control.";

          leaf rule-name {
            type string;
            mandatory true;
            description
              "The name of the rule.
               This must be unique.";
          }

          leaf rule-id {
            type uint8;
            mandatory true;
            description
              "The ID of the rule.
               This is key for rule-list.
               This must be unique.";
```

```
          }

          container event {
            description
              " 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.).";

              list time-event-list {
                key "time-id";
                description
                  "TBD";

                leaf event-time {
                  type yang:date-and-time;
                  mandatory true;
                  description
                    "TBD";
                }

                leaf time-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the time-event.
                     This is the key for time-event-list.
                     This must be unique.";
                }
              }

              container user-action {
                description
                  "User related actions such as login, logoff,
                   and etc.";
                leaf login {
                  type boolean;
                  description
                    "When users login, this event occurs.";
                }
                leaf logoff {
                  type boolean;
                  description
                    "When users logoff, this event occurs.";
```

```
                }
                leaf originating {
                  type boolean;
                  description
                    "When users originate voip-volte,
                     this event occurs.";
                }
                leaf terminating {
                  type boolean;
                  description
                    "When users terminate voip-volte,
                     this event occurs.";
                }
              }
          }

          container condition {
            description
              "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 make a decision. When used in
               the context of policy rules for flow-based NSFs, it is
               used to determine whether or not the set of Actions in
               that Policy Rule can be executed or not. The following
               contents (which are not exhausted) of the received
               packets can be used in the Condition.";

            container packet-content-values {
              description
                "Refer to the kind of information or
                 attributes acquired directly from the packet headers
                 or payloads that can be used in the security policy
                 directly. It can be any fields or attributes in the
                 packet L2/L3/L4 header, or special segment of bytes
                 in the packet payload.";

              list layer-234-packet-header {
                key "packet-header-id";
                description
                  "All meaningful and useful attributes in
                   L2/L3/L4 packet header, for example:
                   src/dest address, or src/dest port.";

                leaf packet-header-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the packet-header.
```

```
                     This is key for packet-header-list.
                     This must be unique.";
                }
                container address-scope {
                  description
                    "Address Scope.";
                  choice route-type {
                    description
                     "Route types: IPv4, IPv6, MPLS, MAC etc.";
                    case ipv4 {
                      description
                        "IPv4 route case.";
                      container ipv4 {
                        description
                          "IPv4 route match.";
                        choice ip-route-match-type {
                          description
                            "IP route match type options:
                             match source, or
                             match destination, or
                             match source and destination.";
                          case dest-ipv4-address {
                            leaf dest-ipv4-prefix {
                              type inet:ipv4-prefix;
                              mandatory true;
                              description
                                "An IPv4 destination address
                                 as the match.";
                            }
                          }
                          case src-ipv4-address {
                            leaf src-ipv4-prefix {
                              type inet:ipv4-prefix;
                              mandatory true;
                              description
                                "An IPv4 source address as the match.";
                            }
                          }
                          case dest-src-ipv4-address {
                            container dest-src-ipv4-address {
                              description
                                "A combination of an IPv4 source and
                                 an IPv4 destination address
                                 as the match.";
                              leaf dest-ipv4-prefix {
                                type inet:ipv4-prefix;
                                mandatory true;
                                description
```

```
                                  "An IPv4 destination address of
                                   the match.";
                              }
                              leaf src-ipv4-prefix {
                                type inet:ipv4-prefix;
                                mandatory true;
                                description
                                  "An IPv4 destination address of
                                   the match.";
                              }
                            }
                          }
                        }
                      }
                    }
                    case ipv6 {
                      description
                        "IPv6 route case.";
                      container ipv6 {
                        description
                          "IPv6 route match.";
                        choice ip-route-match-type {
                          description
                            "IP route match type options:
                             match source, or
                             match destination, or
                             match source and destination.";
                          case dest-ipv6-address {
                            leaf dest-ipv6-prefix {
                              type inet:ipv6-prefix;
                              mandatory true;
                              description
                                "An IPv6 destination address
                                 as the match.";
                            }
                          }
                          case src-ipv6-address {
                            leaf src-ipv6-prefix {
                              type inet:ipv6-prefix;
                              mandatory true;
                              description
                                "An IPv6 source address
                                 as the match.";
                            }
                          }
                          case dest-src-ipv6-address {
                            container dest-src-ipv6-address {
                              description
```

```
                  "A combination of an IPv6 source and
                   an IPv6 destination address as the
                   match.";
                leaf dest-ipv6-prefix {
                  type inet:ipv6-prefix;
                  mandatory true;
                  description
                    "An IPv6 destination address of
                     the match.";
                }
                leaf src-ipv6-prefix {
                  type inet:ipv6-prefix;
                  mandatory true;
                  description
                    "An IPv6 destination address of
                     the match.";
                }
              }
            }
          }
        }
      }
      case mpls-route {
        description
          "MPLS route case.";
        leaf mpls-label {
          type uint32;
          mandatory true;
          description
            "The label used for matching.";
        }
      }
      case mac-route {
        description
          "MAC route case.";
        leaf mac-address {
          type uint32;
          mandatory true;
          description
            "The MAC address used for matching.";
        }
      }
      case interface-route {
        description
          "Interface route case.";
        leaf interface-identifier {
          type if:interface-ref;
          mandatory true;
```

```
                           description
                             "The interface used for matching.";
                         }
                       }
                     }
                   }
                   choice layer-type {
                     description
                       "Layer Types: Layer 2, Layer 3, and Layer 4.";
                     case layer-2-header {
                       description
                         "A header about Layer 2.";
                       leaf src-mac-address {
                         type yang:phys-address;
                         mandatory true;
                         description
                           "Source MAC Address.";
                       }
                       leaf dest-mac-address {
                         type yang:phys-address;
                         mandatory true;
                         description
                           "Destination MAC Address.";
                       }
                     }
                     case layer-3-header {
                       description
                         "A header about Layer 3.";
                       choice packet-type {
                         description
                           "Packet Types: IPv4 and IPv6.";
                           case ipv4 {
                             uses ipv4-header;
                           }
                           case ipv6{
                             uses ipv6-header;
                           }
                       }
                     }
                     case layer-4-header {
                       description
                         "A header about Layer 4.";
                       leaf src-port {
                         type inet:port-number;
                         mandatory true;
                         description
                           "source port number.";
                       }
```

```
                  leaf dest-port {
                    type inet:port-number;
                    mandatory true;
                    description
                      "destination port number.";
                  }
                }
              }
            }
            list packet-payload {
              key "packet-payload-id";
              description
                "TBD.";

              leaf packet-payload-id {
                type uint8;
                mandatory true;
                description
                  "TBD.";
              }
            }
            container context-values {
              description
                "Refer to the context information for the
                 received packets.";
              list user{
                key "user-id";
                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)";

                leaf user-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the user.
                     This is key for user-list.
                     This must be unique.";
                }
                choice user-name {
```

```
                    description
                      "The name of the user.
                       This must be unique.";
                    case tenant {
                      description
                        "Tenant information.";
                      leaf tenant {
                        type uint8;
                        mandatory true;
                        description
                          "User's tenant information.";
                      }
                    }
                    case vn-id {
                      description
                        "VN-ID information.";
                      leaf vn-id {
                        type uint8;
                        mandatory true;
                        description
                          "User's VN-ID information.";
                      }
                    }
                  }
                }
                container schedule {
                  description
                    "A schedule defines time range. A rule can
                     reference a schedule to filter traffic that
                     passes through the NSF within the schedule.
                     A schedule can be a periodic schedule, or a
                     one-time schedule.";
                  leaf name {
                    type string;
                    mandatory true;
                    description
                      "The name of the schedule.
                       This must be unique.";
                  }
                  choice schedule-type {
                    description
                      "We can configure a schedule either once or
                       periodic.";
                    case once {
                      description
                        "If we need to configure only once rule,
                         we can use the once option.";
                      leaf once {
```

```
                type boolean;
                mandatory true;
                description
                  "The once option of a schedule.";
              }
            }
            case periodic {
              description
                "If we need to configure periodic rule,
                 we can use the periodic option.";
              leaf periodic {
                type boolean;
                mandatory true;
                description
                  "The periodic option of a schedule.";
              }
            }
          }
          leaf start-time {
            type yang:date-and-time;
            mandatory true;
            description
              "This is start time to take effect on rules.";
          }
          leaf end-time {
            type yang:date-and-time;
            mandatory true;
            description
              "This is end time to take effect on rules.";
          }
          leaf weekly-validity-time {
            type yang:date-and-time;
            mandatory true;
            description
              "This is weekly validity time.";
          }
        }

        list region {
          key "ip-address-region";
          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 ip-address-region {
            type uint8;
```

```
                    mandatory true;
                    description
                      "This is mapped to ip address. We can acquire
                       region through ip address stored the database.";
                  }
                }

                container target {
                  description
                    "Under the circumstances of network, it mainly
                     refers to the service, application, and device.";
                  container service{
                    description
                      "A service is an application identified by a
                       protocol type and port number, such as TCP,
                       UDP, ICMP, and IP.";
                    leaf name {
                      type string;
                      mandatory true;
                      description
                        "The name of the service.
                         This must be unique.";
                    }
                    leaf id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the service.
                         This must be unique.";
                    }
                    container protocol {
                      description
                        "Protocol types:
                           TCP, UDP, ICMP, ICMPv6, IP, and etc.";
                      leaf tcp  {
                        type boolean;
                        mandatory true;
                        description
                          "TCP protocol type.";
                      }
                      leaf udp {
                        type boolean;
                        mandatory true;
                        description
                          "UDP protocol type.";
                      }
                      leaf icmp {
                        type boolean;
```

```
                 mandatory true;
                 description
                   "ICMP protocol type.";
               }
               leaf icmpv6 {
                 type boolean;
                 mandatory true;
                 description
                   "ICMPv6 protocol type.";
               }
               leaf ip {
                 type boolean;
                 mandatory true;
                 description
                   "IP protocol type.";
               }
             }
             leaf src-port{
               type inet:port-number;
               description
                 "It can be used for finding programs.";
             }
             leaf dest-port{
               type inet:port-number;
               description
                 "It can be used for finding programs.";
             }
           }
           container application {
             description
               "An application is a computer program for
                a specific task or purpose. It provides
                a finer granularity than service in matching
                traffic.";
             leaf name{
               type string;
               mandatory true;
               description
                 "The name of the application.
                  This must be unique.";
             }
             leaf id{
               type uint8;
               mandatory true;
               description
                 "The ID of the application.
                  This must be unique.";
             }
```

```
                  container category{
                    description
                      "Category types: Business system, Entertainment,
                       Interest, Network, General, and etc.";
                    leaf business-system {
                      type boolean;
                      description
                        "Business system category.";
                    }
                    leaf entertainment {
                      type boolean;
                      description
                        "Entertainment category.";
                    }
                    leaf interest {
                      type boolean;
                      description
                        "Interest category.";
                    }
                    leaf network {
                      type boolean;
                      description
                        "Network category.";
                    }
                    leaf general {
                      type boolean;
                      description
                        "General category.";
                    }
                  }
                  container subcategory{
                    description
                      "Subcategory types: Finance, Email, Game,
                       Media sharing, Social network, Web posting,
                       and etc.";
                    leaf finance {
                      type boolean;
                      description
                        "Finance subcategory.";
                    }
                    leaf email {
                      type boolean;
                      description
                        "Email subcategory.";
                    }
                    leaf game {
                      type boolean;
                      description
```

```
                            "Game subcategory.";
                        }
                        leaf media-sharing {
                          type boolean;
                          description
                            "Media sharing subcategory.";
                        }
                        leaf social-network {
                          type boolean;
                          description
                            "Social network subcategory.";
                        }
                        leaf web-posting {
                          type boolean;
                          description
                            "Web posting subcategory.";
                        }
                      }
                      container data-transmission-model{
                        description
                          "Data transmission model types: Client-server,
                           Browser-based, Networking, Peer-to-Peer,
                           Unassigned, and etc.";
                        leaf client-server {
                          type boolean;
                          description
                            "client-server data transmission model.";
                        }
                        leaf browser-based {
                          type boolean;
                          description
                            "Browser-based data transmission model.";
                        }
                        leaf networking {
                            type boolean;
                            description
                            "Networking data transmission model.";
                        }
                        leaf peer-to-peer {
                          type boolean;
                          description
                            "Peer-to-Peer data transmission model.";
                        }
                        leaf unassigned {
                          type boolean;
                          description
                            "Unassigned data transmission model.";
                        }
```

```
                    }
                    container risk-level{
                      description
                        "Risk level types: Exploitable,
                         Productivity loss, Evasive, Data loss,
                         Malware vehicle, Bandwidth consuming,
                         Tunneling, and etc.";
                      leaf exploitable {
                        type boolean;
                        description
                          "Exploitable risk level.";
                      }
                      leaf productivity-loss {
                        type boolean;
                        description
                          "Productivity loss risk level.";
                      }
                      leaf evasive {
                        type boolean;
                        description
                          "Evasive risk level.";
                      }
                      leaf data-loss {
                        type boolean;
                        description
                          "Data loss risk level.";
                      }
                      leaf malware-vehicle {
                        type boolean;
                        description
                          "Malware vehicle risk level.";
                      }
                      leaf bandwidth-consuming {
                        type boolean;
                        description
                          "Bandwidth consuming risk level.";
                      }
                      leaf tunneling {
                        type boolean;
                        description
                          "Tunneling risk level.";
                      }
                    }
                  }
                  container device {
                    description
                      "The device attribute that can identify a device,
                       including the device type (i.e., router, switch,
```

```
                          pc, ios, or android) and the device's owner as
                          well.";
                       leaf pc {
                         type boolean;
                         description
                           "If type of a device is PC.";
                       }
                       leaf mobile-phone {
                         type boolean;
                         description
                           "If type of a device is mobile-phone.";
                       }
                       leaf tablet {
                         type boolean;
                         description
                           "If type of a device is tablet.";
                       }
                       leaf voip-volte-phone {
                         type boolean;
                         description
                           "If type of a device is voip-volte-phone.";
                       }

                     }

                     choice state {
                       description
                         "It refers to various states with which the
                          network flow is associated. It can be either
                          the TCP session state (i.e., new, established,
                          related, invalid, or untracked),
                          the session AAA state or the access mode of
                          the devices (i.e., wire, wireless, or vpn).";
                       case session-state {
                         description "TBD.";
                         container tcp-session-state {
                           description
                             "TCP session state types: new,
                              established, related, invalid,
                              and untracked.";
                           leaf new {
                             type boolean;
                             description "New state.";
                           }
                           leaf established {
                             type boolean;
                             description "Established state.";
                           }
```

```
                  leaf related {
                    type boolean;
                    description "Related state.";
                  }
                  leaf invalid {
                    type boolean;
                    description "Invalid state.";
                  }
                  leaf untracked {
                    type boolean;
                    description "untracked state.";
                  }
                }
              }
              case session-aaa-state {
                description "TBD.";
                container session-sip-state {
                  description "TBD.";
                  leaf auth-state {
                    type boolean;
                    description "untracked state.";
                  }
                  leaf call-state {
                    type boolean;
                    description "untracked state.";
                  }
                }
              }
              case access-mode {
                description "TBD.";
                leaf access-mode {
                  type string;
                  mandatory true;
                  description "TBD.";
                }
              }
            }

            container direction {
              description "The direction of the network flow.";
              leaf request {
                type boolean;
                description "TBD.";
              }
              leaf response {
                type boolean;
                description "TBD.";
              }
```

```
                  }
                }
              }
            }
          }
          choice action-type {
            description
              "The flow-based NSFs realize the network security
               functions by executing various Actions, which at least
               includes ingress-action, egress-action, and
               advanced-action.";
            case ingress-action {
              description
                "The ingress actions consist of permit, deny,
                 and mirror.";
              choice ingress-action-type {
                description
                  "Ingress action type: permit, deny, and mirror.";
                case permit {
                  description
                    "Permit case.";
                  leaf permit {
                    type boolean;
                    mandatory true;
                    description
                      "Packet flow is permitted.";
                  }
                }
                case deny {
                  description
                    "Deny case.";
                  leaf deny {
                    type boolean;
                    mandatory true;
                    description
                      "Packet flow is denied.";
                  }
                }
                case mirror {
                  description
                    "Mirror case.";
                  leaf mirror {
                    type boolean;
                    mandatory true;
                    description
                      "Packet flow is mirroried.";
                  }
                }
```

```
              }
            }
            case egress-action {
              description
                "The egress actions consist of invoke-signaling,
                tunnel-encapsulation, and forwarding.";
              choice egress-action-type {
                description
                  "Egress-action-type: invoke-signaling,
                   tunnel-encapsulation, and forwarding.";
                case invoke-signaling {
                  description
                    "Invoke-signaling case.";
                  leaf invoke-signaling {
                    type boolean;
                    mandatory true;
                    description
                      "TBD.";
                  }
                }
                case tunnel-encapsulation {
                  description
                    "tunnel-encapsulation case.";
                  leaf tunnel-encapsulation {
                    type boolean;
                    mandatory true;
                    description
                      "TBD.";
                  }
                }
                case forwarding {
                  description
                    "forwarding case.";
                  leaf forwarding {
                    type boolean;
                    mandatory true;
                    description
                      "TBD.";
                  }
                }
              }
            }
            case advanced-action {
              description
                "Applying a specific Functional Profile or signature
                 - e.g., an IPS Profile, a signature file, an
                 anti-virus file, or a URL filtering file. The
                 functional profile or signature file corresponds to
```

```
                 the security capability for the content security
                 control and attack mitigation control which will be
                 described afterwards. It is one of the key properties
                 that determine the effectiveness of the NSF, and is
                 mostly vendor specific today. One goal of I2NSF is
                 to standardize the form and functional interface of
                 those security capabilities while supporting vendor-
                 specific implementations of each.";
              choice advanced-action-type {
                description
                  "Advanced action types: Content Security Control
                   and Attack Mitigation Control.";
                case content-security-control {
                  description
                    "Content security control is another category of
                    security capabilities applied to application layer.
                    Through detecting the contents carried over the
                    traffic in application layer, these capabilities
                    can realize various security purposes, such as
                    defending against intrusion, inspecting virus,
                    filtering malicious URL or junk email, and blocking
                    illegal web access or data retrieval.";
                  choice content-security-control-type {
                    description
                     "Content Security types: Antivirus, IPS,
                      url-filtering file-blocking, data-filtering,
                      application-control, and voip-volte.";
                    case antivirus {
                      leaf antivirus {
                        type boolean;
                        description
                          "Antivirus is computer software used to
                           prevent, detect and remove malicious
                           software.";
                      }
                    }
                    case ips {
                      leaf ips {
                        type boolean;
                        description
                          "Intrusion prevention systems (IPS) are
                           network security appliances that monitor
                           network and/or system activities for
                           malicious activities.";
                      }
                    }
                    case url-filtering {
                      leaf url-filtering {
```

```
                    type boolean;
                    description
                      "URL filtering security service.";
                  }
                }
                case file-blocking {
                  leaf file-blocking {
                    type boolean;
                    description
                      "File blocking security service.";
                  }
                }
                case data-filtering {
                  leaf data-filtering {
                    type boolean;
                    description
                      "Data filtering security service.";
                  }
                }
                case application-control {
                  leaf application-control {
                    type boolean;
                    description
                      "Application control security service.";
                  }
                }
                case voip-volte {
                  list voip-volte-rule {
                    key "voip-volte-rule-id";
                    description
                      "For the VoIP/VoLTE security system, a VoIP/
                       VoLTE security system can monitor each
                       VoIP/VoLTE flow and manage VoIP/VoLTE
                       security rules controlled by a centralized
                       server for VoIP/VoLTE security service
                       (called VoIP IPS). The VoIP/VoLTE security
                       system controls each switch for the
                       VoIP/VoLTE call flow management by
                       manipulating the rules that can be added,
                       deleted, or modified dynamically.";
                    leaf voip-volte-rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the voip-volte-rule.
                         This is the key for voip-volte-rule-list.
                         This must be unique.";
                    }
```

```
                        container event {
                          description
                            "Event types: VoIP and VoLTE.";
                          leaf called-voip {
                            type boolean;
                            mandatory true;
                            description
                              "If content-security-control-type is
                               voip.";
                          }
                          leaf called-volte {
                            type boolean;
                            mandatory true;
                            description
                              "If content-security-control-type is
                               volte.";
                          }
                        }
                        container condition {
                          description
                            "TBD.";
                          list sip-header {
                            key "sip-header-uri";
                            description
                              "TBD.";
                            leaf sip-header-uri {
                              type string;
                              mandatory true;
                              description
                                "SIP header URI.";
                            }
                            leaf sip-header-method {
                              type string;
                              mandatory true;
                              description
                                "SIP header method.";
                            }
                            leaf sip-header-expire-time {
                              type yang:date-and-time;
                              mandatory true;
                              description
                                "SIP header expire time.";
                            }
                            leaf sip-header-user-agent {
                              type uint32;
                              mandatory true;
                              description
                                "SIP header user agent.";
```

```
                            }
                          }
                          list cell-region {
                            key "cell-id-region";
                            description
                              "TBD.";
                            leaf cell-id-region {
                              type uint32;
                              mandatory true;
                              description
                                "Cell region.";
                            }
                          }
                        }
                        container action {
                          description
                            "The flow-based NSFs realize the security
                             functions by executing various Actions.";
                          choice action-type {
                            description
                              "Action type: ingress action and
                               egress action.";
                            case ingress-action {
                              description
                                "The ingress actions consist of permit,
                                 deny, and mirror.";
                              choice ingress-action-type {
                                description
                                  "Ingress-action-type: permit, deny,
                                   and mirror.";
                                case permit {
                                  description
                                    "Permit case.";
                                  leaf permit {
                                    type boolean;
                                    mandatory true;
                                    description
                                      "Packet flow is permitted.";
                                  }
                                }
                                case deny {
                                  description
                                    "Deny case.";
                                  leaf deny {
                                    type boolean;
                                    mandatory true;
                                    description
                                      "Packet flow is denied.";
```

```
                                  }
                                }
                                case mirror {
                                  description
                                    "Mirror case.";
                                  leaf mirror {
                                    type boolean;
                                    mandatory true;
                                    description
                                      "Packet flow is mirrored.";
                                  }
                                }
                              }
                            }
                            case egress-action {
                              description
                                "The engress actions consist of
                                 mirror and etc.";
                              choice egress-action-type {
                                description
                                  "Engress-action-type: redirection,
                                   and etc.";
                                case redirection {
                                  description
                                    "Redirection case.";
                                  leaf redirection {
                                    type boolean;
                                    mandatory true;
                                    description "TBD.";
                                  }
                                }
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
                case attack-mitigation-control {
                  description
                    "This category of security capabilities is
                     specially used to detect and mitigate various
                     types of network attacks.";
                  choice attack-mitigation-control-type {
                    description
                      "Attack-mitigation types: DDoS-attack and
                       Single-packet attack.";
```

```
                    case ddos-attack {
                      description
                        "A distributed-denial-of-service (DDoS) is
                         where the attack source is more than one,
                         often thousands of unique IP addresses.";
                      choice ddos-attack-type {
                        description
                          "DDoS-attack types: Network Layer DDoS Attacks
                           and Application Layer DDoS Attacks.";
                        case network-layer-ddos-attack {
                          description
                            "Network layer DDoS-attack.";
                          choice network-layer-ddos-attack-type {
                            description
                              "Network layer DDoS attack types:
                               Syn Flood Attack, UDP Flood Attack,
                               ICMP Flood Attack, IP Fragment Flood,
                               IPv6 Related Attacks, and etc";
                            case syn-flood-attack {
                              description
                                "If the network layer DDoS-attack is
                                 a syn flood attack.";
                              leaf syn-flood {
                                type boolean;
                                mandatory true;
                                description
                                  "Syn Flood Attack.";
                              }
                            }
                            case udp-flood-attack {
                              description
                                "If the network layer DDoS-attack is
                                 a udp flood attack.";
                              leaf udp-flood {
                                type boolean;
                                mandatory true;
                                description
                                  "UDP Flood Attack.";
                              }
                            }
                            case icmp-flood-attack {
                              description
                                "If the network layer DDoS-attack is
                                 an icmp flood attack.";
                              leaf icmp-flood {
                                type boolean;
                                mandatory true;
                                description
```

```
                                   "ICMP Flood Attack.";
                               }
                             }
                             case ip-fragment-flood-attack {
                               description
                                 "If the network layer DDoS-attack is
                                  an ip fragment flood attack.";
                               leaf ip-fragment-flood {
                                 type boolean;
                                 mandatory true;
                                 description
                                   "IP Fragment Flood.";
                               }
                             }
                             case ipv6-related-attacks {
                               description
                                 "If the network layer DDoS-attack is
                                  ipv6 related attacks.";
                               leaf ipv6-related {
                                 type boolean;
                                 mandatory true;
                                 description
                                   "IPv6 Related Attacks.";
                               }
                             }
                           }
                         }
                         case app-layer-ddos-attack {
                           description
                             "Application layer DDoS-attack.";
                           choice app-ddos-attack-type {
                             description
                               "Application layer DDoS-attack types:
                                Http Flood Attack, Https Flood Attack,
                                DNS Flood Attack, and
                                DNS Amplification Flood Attack,
                                SSL DDoS Attack, and etc.";
                             case http-flood-attack {
                               description
                                 "If the application layer DDoS-attack is
                                  a http flood attack.";
                               leaf http-flood {
                                 type boolean;
                                 mandatory true;
                                 description
                                   "Http Flood Attack.";
                               }
                             }
```

```
                      case https-flood-attack {
                        description
                          "If the application layer DDoS-attack is
                           a https flood attack.";
                        leaf https-flood {
                          type boolean;
                          mandatory true;
                          description
                            "Https Flood Attack.";
                        }
                      }
                      case dns-flood-attack {
                        description
                          "If the application layer DDoS-attack is
                           a dns flood attack.";
                        leaf dns-flood {
                          type boolean;
                          mandatory true;
                          description
                            "DNS Flood Attack.";
                        }
                      }
                      case dns-amp-flood-attack {
                        description
                          "If the application layer DDoS-attack is
                           a dns amplification flood attack.";
                        leaf dns-amp-flood {
                          type boolean;
                          mandatory true;
                          description
                            "DNS Amplification Flood Attack.";
                        }
                      }
                      case ssl-ddos-attack {
                        description
                          "If the application layer DDoS-attack is
                           an ssl DDoS attack.";
                        leaf ssl-ddos {
                          type boolean;
                          mandatory true;
                          description
                            "SSL Flood Attack.";
                        }
                      }
                    }
                  }
                }
              }
```

```
                    case single-packet-attack {
                      description
                        "Single Packet Attacks.";
                      choice single-packet-attack-type {
                        description
                          "DDoS-attack types: Scanning Attack,
                           Sniffing Attack, Malformed Packet Attack,
                           Special Packet Attack, and etc.";
                        case scan-and-sniff-attack {
                          description
                            "Scanning and Sniffing Attack.";
                          choice scan-and-sniff-attack-type {
                            description
                              "Scanning and sniffing attack types:
                               IP Sweep attack, Port Scanning,
                               and etc.";
                            case ip-sweep-attack {
                              description
                                "If the scanning and sniffing attack is
                                 an ip sweep attack.";
                              leaf ip-sweep {
                                type boolean;
                                mandatory true;
                                description
                                  "IP Sweep Attack.";
                              }
                            }
                            case port-scanning-attack {
                              description
                                "If the scanning and sniffing attack is
                                 a port scanning attack.";
                              leaf port-scanning {
                                type boolean;
                                mandatory true;
                                description
                                  "Port Scanning Attack.";
                              }
                            }
                          }
                        }
                        case malformed-packet-attack {
                          description
                            "Malformed Packet Attack.";
                          choice malformed-packet-attack-type {
                            description
                              "Malformed packet attack types:
                               Ping of Death Attack, Teardrop Attack,
                               and etc.";
```

```
                     case ping-of-death-attack {
                       description
                         "If the malformed packet attack is
                          a ping of death attack.";
                       leaf ping-of-death {
                         type boolean;
                         mandatory true;
                         description
                           "Ping of Death Attack.";
                       }
                     }
                     case teardrop-attack {
                       description
                         "If the malformed packet attack is
                          a teardrop attack.";
                       leaf teardrop {
                         type boolean;
                         mandatory true;
                         description
                           "Teardrop Attack.";
                       }
                     }
                   }
                 }
                 case special-packet-attack {
                   description
                     "special Packet Attack.";
                   choice special-packet-attack-type {
                     description
                       "Special packet attack types:
                        Oversized ICMP Attack, Tracert Attack,
                        and etc.";
                     case oversized-icmp-attack {
                       description
                         "If the special packet attack is
                          an oversized icmp attack.";
                       leaf oversized-icmp {
                         type boolean;
                         mandatory true;
                         description
                           "Oversize ICMP Attack.";
                       }
                     }
                     case tracert-attack {
                       description
                         "If the special packet attack is
                          a tracert attack.";
                       leaf tracert {
```

```
                                type boolean;
                                mandatory true;
                                description
                                  "Tracrt Attack.";
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }

      /* grouping */
      grouping ipv4-header {
        description
          "The IPv4 header encapsulation information.";
        leaf src-ipv4-address {
          type inet:ipv4-address;
          mandatory true;
          description
            "The source ip address of the header.";
        }
        leaf dest-ipv4-address {
          type inet:ipv4-address;
          mandatory true;
          description
            "The destination ip address of the header.";
        }
        leaf protocol {
          type uint8;
          mandatory true;
          description
            "The Protocol id of the header.";
        }
        leaf ttl {
          type uint8;
          description
            "The TTL of the header.";
        }
      }

      grouping ipv6-header {
```

```
    description "The IPv6 header encapsulation information.";
    leaf src-ipv6-address {
      type inet:ipv6-address;
      mandatory true;
      description
        "The source ip address of the header.";
      }
      leaf dest-ipv6-address {
        type inet:ipv6-address;
        mandatory true;
        description
          "The destination ip address of the header.";
      }
      leaf next-header {
        type uint8;
        mandatory true;
        description
          "The next header of the IPv6 header.";
      }
      leaf traffic-class {
        type uint8;
        description
          "The traffic class value of the header.";
      }
      leaf flow-label {
        type uint16;
        description
          "The flow label of the header.";
      }
      leaf hop-limit {
        type uint8;
        description
          "The hop limit the header.";
      }
  }


  /* RPC Operations */
  rpc time-event-add {
    description
      "To add a rule of time event or a list of time event.";
    input {
      leaf event-time {
        type yang:date-and-time;
        description
          "TBD.";
      }
      leaf time-id {
```

```
          type uint8;
          description
            "The ID of the time-event.
             This is the key for time-event-list.
             This must be unique.";
        }
      }
      output {
        leaf result {
          type  boolean;
          description
            "Return the result of the time-event-add operation.
             true - success
             false - failed";
        }
        leaf reason {
          type   string;
          description
            "The specific reason that causes the failure.";
        }
      }
    }

    rpc time-event-delete {
      description
        "To delete the rule of a user or a list of users.";
      input {
        leaf time-id {
          type  uint8;
          description
            "The ID of the time-event.
             This is key for time-event-list.
             This must be unique.";
        }
      }
      output {
        leaf result {
          type  boolean;
          description
            "Return the result of the time-event-delete operation.
             true - success
             false - failed.";
        }
        leaf reason {
          type   string;
          description
            "The specific reason that causes the failure.";
        }
```

```
    }
  }

  rpc user-add {
    description
      "To add a rule of user or a list of users.";
    input {
      leaf user-id {
        type uint8;
        mandatory true;
        description
          "The ID of the user.
           This is the key for user-list.
           This must be unique.";
      }
      choice user-name {
        description
          "The name of the user.
           This must be unique.";
        case tenant {
          description
            "Tenant information";
          leaf tenant {
            type uint8;
            mandatory true;
            description
              "User's tenant information";
          }
        }
        case vn-id {
          description
            "VN-ID information.";
          leaf vn-id {
            type uint8;
            mandatory true;
            description
              "User's VN-ID information.";
           }
        }
      }
    }
    output {
      leaf result {
        type  boolean;
        description
          "Return the result of the user-add operation.
           true - success
           false - failed.";
```

```
        }
        leaf reason {
          type   string;
          description
            "The specific reason that causes the failure.";
        }
      }
    }

    rpc user-delete {
      description
        "To delete a rule of user or a list of users.";
      input {
        leaf user-id {
          type  uint8;
          description
            "The ID of the user.
             This is the key for user-list.
             This must be unique.";
        }
      }
      output {
        leaf result {
          type  boolean;
          description
            "Return the result of the user-delete operation.
             true - success
             false - failed.";
        }
        leaf reason {
          type   string;
          description
            "The specific reason that causes the failure.";
        }
      }
    }

    rpc region-add {
      description
        "To add a rule of region or a list of regions.";
      input {
        leaf ip-address-region {
          type  uint8;
          description
            "This is mapped to ip address.
             We can acquire region through ip address stored the
             database.";
        }
```

```
    }
    output {
      leaf result {
        type  boolean;
        description
          "Return the result of the region-add operation.
           true - success
           false - failed.";
      }
      leaf reason {
        type   string;
        description
          "The specific reason that causes the failure.";
      }
    }
  }

  rpc region-delete {
    description
      "To delete a rule of region or a list of region.";
    input {
      leaf ip-address-region {
        type  uint8;
        description
          "This is mapped to ip address.
           We can acquire region through ip address stored the
           database.";
      }
    }
    output {
      leaf result {
        type  boolean;
        description
          "Return the result of the region-delete operation.
           true - success
           false - failed.";
      }
      leaf reason {
        type   string;
        description
          "The specific reason that causes the failure.";
      }
    }
  }
}

<CODE ENDS>
```

Figure 3: Data Model of I2NSF Capability Interface

6. Security Considerations

This document introduces no additional security threats and SHOULD follow the security requirements as stated in [i2nsf-framework].

7. Acknowledgements

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

This document has greatly benefited from inputs by Hyoungshick Kim and Se-Hui Lee.

8. References

8.1. Normative References

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

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

8.2. Informative References

[i2nsf-cap-interface-im] Xia, L., Strassner, J., Li, K., Zhang, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-05 (work in progress), March 2016.

[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-05 (work in progress), March 2016.

[supa-policy-info-model] Strassner, J. and J. Halpern, "Generic Policy Information Model for Simplified

Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-00 (work in progress), June 2016.

[i2nsf-framework]   Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-00 (work in progress), May 2016.

Authors' Addresses

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

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

Jin-Yong 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: wlsdyd0930@nate.com

Dae-Young Hyun
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 4776 5672
EMail: guseodud1@naver.com

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr

Tae-Jin Ahn
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8409
EMail: taejin.ahn@kt.com