Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

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

July 02, 2018

I2NSF Network Security Function-Facing Interface YANG Data Model
draft-ietf-i2nsf-nsf-facing-interface-dm-01

Abstract

This document defines a YANG data model corresponding to the information model for Network Security Functions (NSF) facing interface in Interface to Network Security Functions (I2NSF). It describes a data model for the features provided by generic security functions. This data model provides generic components whose vendors is well understood, so that the generic component can be used even if it has some vendor specific functions. These generic functions represent a point of interoperability, and can be provided by any product that offers the required Capabilities. Also, if vendors need additional features for its network security function, they can add the features by extending the YANG data model.

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of security services with the information model for Network Security Functions (NSF) facing interface in Interface to

Network Security Functions (I2NSF). It provides a specific information model and the corresponding data models for generic network security functions (i.e., network security functions), as defined in [i2nsf-nsf-cap-im]. With these data model, I2NSF controller can control the capabilities of NSFs.

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

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

- configuration of I2NSF security policy rule for generic network security function policy
- configuration of event clause for generic network security function policy
- configuration of condition clause for generic network security function policy
- configuration of action clause for generic network security function policy

2. Requirements Language

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

3. Terminology

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

- 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.
- 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:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

4. The Structure and Objective of I2NSF Security Policy

4.1. I2NSF Security Policy Rule

This shows a policy rule for generic network security functions. The object of a policy rule is defined as policy information and rule information. This includes ECA Policy Rule such as Event Clause Objects, Condition Clause Objects, Action Clause Objects, Resolution Strategy, and Default Action.

4.2. Event Clause

This shows an event clause for generic network security functions. An Event is any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of I2NSF Policy Rules, it is used to determine whether the Condition clause of the I2NSF Policy Rule can be evaluated or not. The object of an event clauses is defined as user security event, device security event, system security event, and time security event. The objects of event clauses can be extended according to specific vendor event features.

4.3. Condition Clause

This shows a condition clause for generic network security functions. A condition is defined as a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to determine whether or not the set

of Actions in that (imperative) I2NSF Policy Rule can be executed or not. These objects are defined as packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. The objects of action clauses can be extended according to specific vendor condition features.

4.4. Action Clause

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

5. Data Model Structure

This section shows a data model structure tree of generic network security functions that are defined in the [i2nsf-nsf-cap-im].

- Consideration of ECA Policy Model by Aggregating the Event, Condition, and Action Clauses Objects.
- Consideration of Capability Algebra.
- Consideration of NSFs Capability Categories (i.e., Network Security, Content Security, and Attack Mitigation Capabilities).
- Definitions for Network Security Event Class, Network Security Condition Class, and Network Security Action Class.

5.1. I2NSF Security Policy Rule

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

```
module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy
|  +--rw policy-name?           string
|  +--rw rules* [rule-name]
|  |  +--rw rule-name                        string
|  |  +--rw rule-description?                string
|  |  +--rw rule-priority?                   uint8
|  |  +--rw enable?                          boolean
|  |  +--rw session-aging-time?              uint16
|  |  +--rw long-connection
```

```
   |  |  |  +--rw enable?   boolean
   |  |  |  +--rw during?   uint16
   |  |  +--rw policy-event-clause-agg-ptr*       instance-identifier
   |  |  +--rw policy-condition-clause-agg-ptr*   instance-identifier
   |  |  +--rw policy-action-clause-agg-ptr*      instance-identifier
   |  |  +--rw time-zone
   |  |     +--rw absolute-time-zone
   |  |     |  +--rw time
   |  |     |  |  +--rw start-time?   yang:date-and-time
   |  |     |  |  +--rw end-time?     yang:date-and-time
   |  |     |  +--rw date
   |  |     |     +--rw absolute-date?   yang:date-and-time
   |  |     +--rw periodic-time-zone
   |  |        +--rw day
   |  |        |  +--rw sunday?      boolean
   |  |        |  +--rw monday?      boolean
   |  |        |  +--rw tuesday?     boolean
   |  |        |  +--rw wednesday?   boolean
   |  |        |  +--rw thursday?    boolean
   |  |        |  +--rw friday?      boolean
   |  |        |  +--rw saturday?    boolean
   |  |        +--rw month
   |  |           +--rw january?     boolean
   |  |           +--rw february?    boolean
   |  |           +--rw march?       boolean
   |  |           +--rw april?       boolean
   |  |           +--rw may?         boolean
   |  |           +--rw june?        boolean
   |  |           +--rw july?        boolean
   |  |           +--rw august?      boolean
   |  |           +--rw september?   boolean
   |  |           +--rw october?     boolean
   |  |           +--rw november?    boolean
   |  |           +--rw december?    boolean
   |  +--rw resolution-strategy
   |  |  +--rw (resolution-strategy-type)?
   |  |     +--:(fmr)
   |  |     |  +--rw first-matching-rule?   boolean
   |  |     +--:(lmr)
   |  |        +--rw last-matching-rule?    boolean
   |  +--rw default-action
   |  |  +--rw default-action-type?   boolean
   |  +--rw rule-group
   |     +--rw groups* [group-name]
   |        +--rw group-name     string
   |        +--rw rule-range
   |        |  +--rw start-rule?   string
   |        |  +--rw end-rule?     string
```

```
|        +--rw enable?        boolean
|        +--rw description?   string
+--rw event-clause-container
|  ...
+--rw condition-clause-container
|  ...
+--rw action-clause-container
   ...
```

Figure 1: Data Model Structure for Network Security Policy Identification

5.2. Event Clause

The data model for event rule has the following structure:

```
module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|     ...
|  +--rw eca-policy-rules* [rule-id]
|     ...
|  +--rw resolution-strategy
|     ...
|  +--rw default-action
|     ...
+--rw event-clause-container
|  +--rw event-clause-list* [eca-object-id]
|     +--rw entity-class?        identityref
|     +--rw eca-object-id        string
|     +--rw description?              string
|     +--rw sec-event-content    string
|     +--rw sec-event-format     sec-event-format
|     +--rw sec-event-type       string
+--rw condition-clause-container
|  ...
+--rw action-clause-container
   ...
```

Figure 2: Data Model Structure for Event Rule

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

5.3. Condition Clause

The data model for condition rule has the following structure:

```
module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|     ...
|  +--rw eca-policy-rules* [rule-id]
|     ...
|  +--rw resolution-strategy
|     ...
|  +--rw default-action
|     ...
+--rw event-clause-container
|  ...
+--rw condition-clause-container
|  +--rw condition-clause-list* [eca-object-id]
|     +--rw entity-class?                identityref
|     +--rw eca-object-id                string
|     +--rw packet-security-condition
|     |  +--rw packet-description?               string
|     |  +--rw packet-security-mac-condition
|     |  |  +--rw pkt-sec-cond-mac-dest*         yang:phys-address
|     |  |  +--rw pkt-sec-cond-mac-src*          yang:phys-address
|     |  |  +--rw pkt-sec-cond-mac-8021q*        string
|     |  |  +--rw pkt-sec-cond-mac-ether-type*   string
|     |  |  +--rw pkt-sec-cond-mac-tci*          string
|     |  +--rw packet-security-ipv4-condition
|     |  |  +--rw pkt-sec-cond-ipv4-header-length*     uint8
|     |  |  +--rw pkt-sec-cond-ipv4-tos*               uint8
|     |  |  +--rw pkt-sec-cond-ipv4-total-length*      uint16
|     |  |  +--rw pkt-sec-cond-ipv4-id*                uint8
|     |  |  +--rw pkt-sec-cond-ipv4-fragment*          uint8
|     |  |  +--rw pkt-sec-cond-ipv4-fragment-offset*   uint16
|     |  |  +--rw pkt-sec-cond-ipv4-ttl*               uint8
|     |  |  +--rw pkt-sec-cond-ipv4-protocol*          uint8
|     |  |  +--rw pkt-sec-cond-ipv4-src*               inet:ipv4-address
|     |  |  +--rw pkt-sec-cond-ipv4-dest*              inet:ipv4-address
|     |  |  +--rw pkt-sec-cond-ipv4-ipopts?            string
|     |  |  +--rw pkt-sec-cond-ipv4-sameip?            boolean
|     |  |  +--rw pkt-sec-cond-ipv4-geoip*             string
|     |  +--rw packet-security-ipv6-condition
|     |  |  +--rw pkt-sec-cond-ipv6-dscp*             string
|     |  |  +--rw pkt-sec-cond-ipv6-ecn*              string
|     |  |  +--rw pkt-sec-cond-ipv6-traffic-class*    uint8
|     |  |  +--rw pkt-sec-cond-ipv6-flow-label*       uint32
|     |  |  +--rw pkt-sec-cond-ipv6-payload-length*   uint16
|     |  |  +--rw pkt-sec-cond-ipv6-next-header*      uint8
```

```
|     |  |  +--rw pkt-sec-cond-ipv6-hop-limit*        uint8
|     |  |  +--rw pkt-sec-cond-ipv6-src*              inet:ipv6-address
|     |  |  +--rw pkt-sec-cond-ipv6-dest*             inet:ipv6-address
|     |  +--rw packet-security-tcp-condition
|     |  |  +--rw pkt-sec-cond-tcp-src-port*      inet:port-number
|     |  |  +--rw pkt-sec-cond-tcp-dest-port*     inet:port-number
|     |  |  +--rw pkt-sec-cond-tcp-seq-num*       uint32
|     |  |  +--rw pkt-sec-cond-tcp-ack-num*       uint32
|     |  |  +--rw pkt-sec-cond-tcp-window-size*   uint16
|     |  |  +--rw pkt-sec-cond-tcp-flags*         uint8
|     |  +--rw packet-security-udp-condition
|     |  |  +--rw pkt-sec-cond-udp-src-port*    inet:port-number
|     |  |  +--rw pkt-sec-cond-udp-dest-port*   inet:port-number
|     |  |  +--rw pkt-sec-cond-udp-length*      string
|     |  +--rw packet-security-icmp-condition
|     |     +--rw pkt-sec-cond-icmp-type*      uint8
|     |     +--rw pkt-sec-cond-icmp-code*      uint8
|     |     +--rw pkt-sec-cond-icmp-seg-num*   uint32
|     +--rw packet-payload-condition
|     |  +--rw packet-payload-description?   string
|     |  +--rw pkt-payload-content*          string
|     +--rw acl-number?                  uint32
|     +--rw application-condition
|     |  +--rw application-description?   string
|     |  +--rw application-object*        string
|     |  +--rw application-group*         string
|     |  +--rw application-label*         string
|     |  +--rw category
|     |     +--rw application-category* [name application-subcategory]
|     |        +--rw name                       string
|     |        +--rw application-subcategory    string
|     +--rw target-condition
|     |  +--rw target-description?        string
|     |  +--rw device-sec-context-cond
|     |     +--rw pc?                 boolean
|     |     +--rw mobile-phone?       boolean
|     |     +--rw voip-volte-phone?   boolean
|     |     +--rw tablet?             boolean
|     |     +--rw iot?                boolean
|     |     +--rw vehicle?            boolean
|     +--rw users-condition
|     |  +--rw users-description?   string
|     |  +--rw user
|     |  |  +--rw (user-name)?
|     |  |     +--:(tenant)
|     |  |     |  +--rw tenant    uint8
|     |  |     +--:(vn-id)
|     |  |        +--rw vn-id     uint8
```

```
|     |  +--rw group
|     |  |  +--rw (group-name)?
|     |  |     +--:(tenant)
|     |  |     |  +--rw tenant    uint8
|     |  |     +--:(vn-id)
|     |  |        +--rw vn-id     uint8
|     |  +--rw security-grup        string
|     +--rw url-category-condition
|     |  +--rw pre-defined-category*    string
|     |  +--rw user-defined-category*   string
|     +--rw context-condition
|     |  +--rw context-description?   string
|     +--rw gen-context-condition
|        +--rw gen-context-description?   string
|        +--rw geographic-location
|           +--rw src-geographic-location*    uint32
|           +--rw dest-geographic-location*   uint32
+--rw action-clause-container
   ...
```

Figure 3: Data Model Structure for Condition Rule

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

5.4. Action Clause

The data model for action rule has the following structure:

```
module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|     ...
|  +--rw eca-policy-rules* [rule-id]
|     ...
|  +--rw resolution-strategy
|     ...
|  +--rw default-action
|     ...
+--rw event-clause-container
|  ...
+--rw condition-clause-container
|  ...
+--rw action-clause-container
   +--rw action-clause-list* [eca-object-id]
```

```
        +--rw entity-class?     identityref
        +--rw eca-object-id     string
        +--rw rule-log?         boolean
        +--rw session-log?      boolean
        +--rw ingress-action
        |  +--rw ingress-description?   string
        |  +--rw ingress-action-type?   ingress-action
        +--rw egress-action
        |  +--rw egress-description?   string
        |  +--rw egress-action-type?   egress-action
        +--rw apply-profile
           +--rw profile-description?         string
           +--rw content-security-control
           |  +--rw content-security-control-types
           |     +--rw antivirus?             string
           |     +--rw ips?                   string
           |     +--rw ids?                   string
           |     +--rw url-filtering?         string
           |     +--rw data-filtering?        string
           |     +--rw mail-filtering?        string
           |     +--rw file-blocking?         string
           |     +--rw file-isolate?          string
           |     +--rw pkt-capture?           string
           |     +--rw application-control?   string
           |     +--rw voip-volte?            string
           +--rw attack-mitigation-control
              +--rw ddos-attack
              |  +--rw ddos-attack-type
              |     +--rw network-layer-ddos-attack
              |     |  +--rw network-layer-ddos-attack-type
              |     |     +--rw syn-flood?       string
              |     |     +--rw udp-flood?       string
              |     |     +--rw icmp-flood?      string
              |     |     +--rw ip-frag-flood?   string
              |     |     +--rw ipv6-related?    string
              |     +--rw app-layer-ddos-attack
              |        +--rw app-ddos-attack-types
              |           +--rw http-flood?      string
              |           +--rw https-flood?     string
              |           +--rw dns-flood?       string
              |           +--rw dns-amp-flood?   string
              |           +--rw ssl-ddos?        string
              +--rw single-packet-attack
                 +--rw single-packet-attack-type
                 +--rw scan-and-sniff-attack
                 |  +--rw scan-and-sniff-attack-types
                 |     +--rw ip-sweep?        string
                 |     +--rw port-scanning?   string
```

```
                    +--rw malformed-packet-attack
                    |  +--rw malformed-packet-attack-types
                    |     +--rw ping-of-death?   string
                    |     +--rw teardrop?        string
                    +--rw special-packet-attack
                       +--rw special-packet-attack-types
                          +--rw oversized-icmp?   string
                          +--rw tracert?          string
```

Figure 4: Data Model Structure for Action Rule

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

6. YANG Module

6.1. IETF NSF-Facing Interface YANG Data Module

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

<CODE BEGINS> file "ietf-i2nsf-policy-rule-for-nsf@2018-07-02.yang"

```
module ietf-i2nsf-policy-rule-for-nsf {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf";
  prefix
    policy-rule-for-nsf;

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

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

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

     WG Chair: Adrian Farrel
```

```
     <mailto:Adrain@olddog.co.uk>

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

     Editor: Jingyong Tim Kim
     <mailto:timkim@skku.edu>

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

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

  description
    "This module defines a YANG data module for network security
     functions.";
  revision "2018-07-02"{
    description "The fourth revision";
    reference
      "draft-ietf-i2nsf-capability-00";
  }

  typedef sec-event-format {
      type enumeration {
        enum unknown {
            description
              "If SecEventFormat is unknown";
        }
        enum guid {
            description
              "If SecEventFormat is GUID
              (Generic Unique IDentifier)";
        }
        enum uuid {
            description
              "If SecEventFormat is UUID
              (Universal Unique IDentifier)";
        }
        enum uri {
            description
              "If SecEventFormat is URI
              (Uniform Resource Identifier)";
        }
        enum fqdn {
            description
              "If SecEventFormat is FQDN
              (Fully Qualified Domain Name)";
```

```
        }
        enum fqpn {
            description
              "If SecEventFormat is FQPN
              (Fully Qualified Path Name)";
        }
      }
      description
        "This is used for SecEventFormat.";
  }

  typedef ingress-action {
      type enumeration {
        enum pass {
            description
              "If ingress action is pass";
        }
        enum drop {
            description
              "If ingress action is drop";
        }
        enum reject {
            description
              "If ingress action is reject";
        }
        enum alert {
            description
              "If ingress action is alert";
        }
        enum mirror {
            description
              "If ingress action is mirror";
        }
      }
      description
        "This is used for ingress action.";
  }

  typedef egress-action {
      type enumeration {
        enum invoke-signaling {
            description
              "If egress action is invoke signaling";
        }
        enum tunnel-encapsulation {
            description
              "If egress action is tunnel encapsulation";
        }
```

```
        enum forwarding {
            description
              "If egress action is forwarding";
        }
        enum redirection {
            description
              "If egress action is redirection";
        }
      }
      description
        "This is used for egress action.";
  }

  identity ECA-OBJECT-TYPE {
    description "TBD";
  }

  identity ECA-EVENT-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
  }

  identity ECA-CONDITION-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
  }

  identity ECA-ACTION-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
  }

   identity EVENT-USER-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }

   identity EVENT-DEV-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }

   identity EVENT-SYS-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
  }

   identity EVENT-TIME-TYPE {
```

```
    base ECA-EVENT-TYPE;
    description "TBD";
  }


  grouping i2nsf-eca-object-type {
    leaf entity-class {
      type identityref {
        base ECA-OBJECT-TYPE;
      }
      description "TBD";
    }
    leaf eca-object-id {
        type string;
        description "TBD";
    }
    description "TBD";
  }

  grouping i2nsf-event-type {
      description "TBD";
      leaf description {
        type string;
        description
          "This is description for event.
          Vendors can write instructions for event
          that vendor made";
      }

      leaf sec-event-content {
        type string;
        mandatory true;
        description
         "This is a mandatory string that contains the content
          of the SecurityEvent. The format of the content
          is specified in the SecEventFormat class
          attribute, and the type of event is defined in the
          SecEventType class attribute. An example of the
          SecEventContent attribute is a string hrAdmin,
          with the SecEventFormat set to 1 (GUID) and the
          SecEventType attribute set to 5 (new logon).";
      }

      leaf sec-event-format {
        type sec-event-format;
        mandatory true;
        description
         "This is a mandatory uint 8 enumerated integer, which
```

```
          is used to specify the data type of the
          SecEventContent attribute. The content is
          specified in the SecEventContent class attribute,
          and the type of event is defined in the
          SecEventType class attribute. An example of the
          SecEventContent attribute is string hrAdmin,
          with the SecEventFormat attribute set to 1 (GUID)
          and the SecEventType attribute set to 5
          (new logon).";
      }

      leaf sec-event-type {
        type string;
        mandatory true;
        description
         "This is a mandatory uint 8 enumerated integer, which
          is used to specify the type of event that involves
          this user. The content and format are specified in
          the SecEventContent and SecEventFormat class
          attributes, respectively. An example of the
          SecEventContent attribute is string hrAdmin,
          with the SecEventFormat attribute set to 1 (GUID)
          and the SecEventType attribute set to 5
         (new logon).";
      }

 }


  container i2nsf-security-policy {
    description
      "policy is a container
       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;
        description
          "The name of the policy.
           This must be unique.";
      }

      list rules {
        key "rule-name";
        description
          "This is a rule for network security functions.";
```

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

        leaf rule-description {
          type string;
          description
            "This description gives more information about
             rules.";
        }

        leaf rule-priority {
          type uint8;
          description
            "The priority keyword comes with a mandatory
             numeric value which can range from 1 till 255.";
        }

        leaf enable {
          type boolean;
          description
            "True is enable.
             False is not enbale.";
        }

        leaf session-aging-time {
          type uint16;
          description
            "This is session aging time.";
        }

        container long-connection {
          description
            "This is long-connection";

          leaf enable {
            type boolean;
            description
              "True is enable.
               False is not enbale.";
          }

          leaf during {
            type uint16;
```

```
            description
              "This is during time.";
          }
        }


        leaf-list policy-event-clause-agg-ptr {
            type instance-identifier;
            must 'derived-from-or-self (/event-clause-container/
            event-clause-list/entity-class, "ECA-EVENT-TYPE")';
            description
                "TBD";
        }
        leaf-list policy-condition-clause-agg-ptr {
            type instance-identifier;
            must 'derived-from-or-self (/condition-clause-container/
            condition-clause-list/entity-class, "ECA-CONDITION-TYPE")';
            description
                "TBD";
        }
        leaf-list policy-action-clause-agg-ptr {
            type instance-identifier;
            must 'derived-from-or-self (/action-clause-container/
            action-clause-list/entity-class, "ECA-ACTION-TYPE")';
            description
                "TBD";
        }

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

```
            }
            container date {
              description
                "This can be used to apply rules according to date";
              leaf absolute-date {
                type yang:date-and-time;
                description
                  "This is absolute date for time zone";
              }
            }
          }
          container periodic-time-zone {
            description
             "This can be used to apply rules according to
             periodic-time-zone";
            container day {
              description
               "This can be used to apply rules according
                to periodic day";
              leaf sunday {
                type boolean;
                description
                  "This is sunday for periodic day";
              }
              leaf monday {
                type boolean;
                description
                  "This is monday for periodic day";
              }
              leaf tuesday {
                type boolean;
                description
                  "This is tuesday for periodic day";
              }
              leaf wednesday {
                type boolean;
                description
                  "This is wednesday for periodic day";
              }
              leaf thursday {
                type boolean;
                description
                  "This is thursday for periodic day";
              }
              leaf friday {
                type boolean;
                description
                  "This is friday for periodic day";
```

```
            }
            leaf saturday {
              type boolean;
              description
                "This is saturday for periodic day";
            }
          }
          container month {
            description
             "This can be used to apply rules according
              to periodic month";
            leaf january {
              type boolean;
              description
                "This is january for periodic month";
            }
            leaf february {
              type boolean;
              description
                "This is february for periodic month";
            }
            leaf march {
              type boolean;
              description
                "This is march for periodic month";
            }
            leaf april {
              type boolean;
              description
                "This is april for periodic month";
            }
            leaf may {
              type boolean;
              description
                "This is may for periodic month";
            }
            leaf june {
              type boolean;
              description
                "This is june for periodic month";
            }
            leaf july {
              type boolean;
              description
                "This is july for periodic month";
            }
            leaf august {
              type boolean;
```

```
            description
              "This is august for periodic month";
          }
          leaf september {
            type boolean;
            description
              "This is september for periodic month";
          }
          leaf october {
            type boolean;
            description
              "This is october for periodic month";
          }
          leaf november {
            type boolean;
            description
              "This is november for periodic month";
          }
          leaf december {
            type boolean;
            description
              "This is december for periodic month";
          }
        }
      }
    }
  }

    container resolution-strategy {
      description
        "The resolution strategies can be used to
        specify how to resolve conflicts that occur between
        the actions of the same or different policy rules that
        are matched and contained in this particular NSF";


      choice resolution-strategy-type {
        description
          "Vendors can use YANG data model to configure rules";

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

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

        }
      }

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

        leaf default-action-type {
          type boolean;
          description
            "True is permit
             False is deny.";
        }
      }


      container rule-group {
        description
          "This is rule group";

        list groups {
          key "group-name";
          description
            "This is a group for rules";

          leaf group-name {
            type string;
            description
              "This is a group for rules";
          }

          container rule-range {
            description
              "This is a rule range.";

            leaf start-rule {
              type string;
              description
```

```
                "This is a start rule";
            }
            leaf end-rule {
              type string;
              description
                "This is a end rule";
            }
          }
          leaf enable {
            type boolean;
            description
              "This is enable
               False is not enable.";
          }
          leaf description {
            type string;
            description
              "This is a desription for rule-group";
          }
        }
      }
    }

  container event-clause-container {
    description "TBD";
    list event-clause-list {
    key eca-object-id;
    uses i2nsf-eca-object-type {
      refine entity-class {
          default ECA-EVENT-TYPE;
      }
    }

    description
      " This is abstract. An event is defined as any important
        occurrence in time of a change in the system being
        managed, and/or in the environment of the system being
        managed. When used in the context of policy rules for
        a flow-based NSF, it is used to determine whether the
        Condition clause of the Policy Rule can be evaluated
        or not. Examples of an I2NSF event include time and
        user actions (e.g., logon, logoff, and actions that
        violate any ACL.).";

     uses i2nsf-event-type;
     }
  }
    container condition-clause-container {
```

```
      description "TBD";
      list condition-clause-list {
        key eca-object-id;
        uses i2nsf-eca-object-type {
            refine entity-class {
                default ECA-CONDITION-TYPE;
            }
        }
        description
          " This is abstract.  A condition is defined as a set
          of attributes, features, and/or values that are to be
          compared with a set of known attributes, features,
          and/or values in order to determine whether or not the
          set of Actions in that (imperative) I2NSF Policy Rule
          can be executed or not. Examples of I2NSF Conditions
          include matching attributes of a packet or flow, and
          comparing the internal state of an NSF to a desired
          state.";

          container packet-security-condition {
            description
              "TBD";
            leaf packet-description {
              type string;
              description
                "This is description for packet condition.
                Vendors can write instructions for packet condition
                that vendor made";
            }

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

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

              leaf-list pkt-sec-cond-mac-src {
                type yang:phys-address;
                description
                  "The MAC source address (6 octets long).";
              }
```

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

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

            leaf-list pkt-sec-cond-mac-tci {
              type string;
              description
               "This is an optional string attribute, and defines
                the Tag Control Information. This consists of a 3
                bit user priority field, a drop eligible indicator
                (1 bit), and a VLAN identifier (12 bits).";
            }
          }

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

            leaf-list pkt-sec-cond-ipv4-header-length {
              type uint8;
              description
                "The IPv4 packet header consists of 14 fields,
                 of which 13 are required.";
            }

            leaf-list pkt-sec-cond-ipv4-tos {
              type uint8;
              description
                "The ToS field could specify a datagram's priority
                 and request a route for low-delay,
                 high-throughput, or highly-reliable service..";
```

```
            }

            leaf-list pkt-sec-cond-ipv4-total-length {
              type uint16;
              description
                "This 16-bit field defines the entire packet size,
                 including header and data, in bytes.";
            }

            leaf-list pkt-sec-cond-ipv4-id {
              type uint8;
              description
                "This field is an identification field and is
                 primarily used for uniquely identifying
                 the group of fragments of a single IP datagram.";
            }

            leaf-list pkt-sec-cond-ipv4-fragment {
              type uint8;
              description
                "IP fragmentation is an Internet Protocol (IP)
                 process that breaks datagrams into smaller pieces
                 (fragments), so that packets may be formed that
                 can pass through a link with a smaller maximum
                 transmission unit (MTU) than the original
                 datagram size.";
            }

            leaf-list pkt-sec-cond-ipv4-fragment-offset {
              type uint16;
              description
                "Fragment offset field along with Don't Fragment
                 and More Fragment flags in the IP protocol
                 header are used for fragmentation and reassembly
                 of IP datagrams.";
            }

            leaf-list pkt-sec-cond-ipv4-ttl {
              type uint8;
              description
                "The ttl keyword is used to check for a specific
                 IP time-to-live value in the header of
                 a packet.";
            }

            leaf-list pkt-sec-cond-ipv4-protocol {
              type uint8;
              description
```

```
            "Internet Protocol version 4(IPv4) is the fourth
             version of the Internet Protocol (IP).";
          }

          leaf-list pkt-sec-cond-ipv4-src {
            type inet:ipv4-address;
            description
              "Defines the IPv4 Source Address.";
          }

          leaf-list pkt-sec-cond-ipv4-dest {
            type inet:ipv4-address;
            description
              "Defines the IPv4 Destination Address.";
          }

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

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

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

        container packet-security-ipv6-condition {
          description
             "The purpose of this Class is to represent packet
             IPv6 packet header information that can be used as
```

```
            part of a test to determine if the set of Policy
            Actions in this ECA Policy Rule should be executed
            or not.";

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

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

          leaf-list pkt-sec-cond-ipv6-traffic-class {
            type uint8;
            description
              "The bits of this field hold two values. The 6
               most-significant bits are used for
               differentiated services, which is used to
               classify packets.";
          }

          leaf-list pkt-sec-cond-ipv6-flow-label {
            type uint32;
            description
              "The flow label when set to a non-zero value
               serves as a hint to routers and switches
               with multiple outbound paths that these
               packets should stay on the same path so that
               they will not be reordered.";
          }

          leaf-list pkt-sec-cond-ipv6-payload-length {
            type uint16;
            description
              "The size of the payload in octets,
               including any extension headers.";
          }

          leaf-list pkt-sec-cond-ipv6-next-header {
            type uint8;
            description
              "Specifies the type of the next header.
```

```
            This field usually specifies the transport
            layer protocol used by a packet's payload.";
        }

        leaf-list pkt-sec-cond-ipv6-hop-limit {
          type uint8;
          description
            "Replaces the time to live field of IPv4.";
        }

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

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

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

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

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

        leaf-list pkt-sec-cond-tcp-seq-num {
          type uint32;
          description
```

```
            "If the SYN flag is set (1), then this is the
             initial sequence number.";
          }

          leaf-list pkt-sec-cond-tcp-ack-num {
            type uint32;
            description
              "If the ACK flag is set then the value of this
               field is the next sequence number that the sender
               is expecting.";
          }

          leaf-list pkt-sec-cond-tcp-window-size {
            type uint16;
            description
              "The size of the receive window, which specifies
               the number of windows size units
               (by default,bytes) (beyond the segment
               identified by the sequence number in the
               acknowledgment field) that the sender of this
               segment is currently willing to recive.";
          }

          leaf-list pkt-sec-cond-tcp-flags {
            type uint8;
            description
              "This is a mandatory string attribute, and defines
               the nine Control bit flags (9 bits).";
          }
        }

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

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

          leaf-list pkt-sec-cond-udp-dest-port {
            type inet:port-number;
            description
```

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

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

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

              leaf-list pkt-sec-cond-icmp-type {
                type uint8;
                description
                  "ICMP type, see Control messages.";
              }

              leaf-list pkt-sec-cond-icmp-code {
                type uint8;
                description
                  "ICMP subtype, see Control messages.";
              }

              leaf-list pkt-sec-cond-icmp-seg-num {
                type uint32;
                description
                  "The icmp Sequence Number.";
              }
            }
          }

          container packet-payload-condition {
            description
              "TBD";
            leaf packet-payload-description {
              type string;
              description
               "This is description for payload condition.
               Vendors can write instructions for payload condition
               that vendor made";
            }
            leaf-list pkt-payload-content {
```

```
            type string;
            description
              "The content keyword is very important in
               signatures. Between the quotation marks you
               can write on what you would like the
               signature to match.";
          }
        }

        leaf acl-number {
          type uint32;
          description
           "This is acl-number.";
        }

        container application-condition {
          description
            "TBD";
          leaf application-description {
            type string;
            description
             "This is description for application condition.";
          }
          leaf-list application-object {
            type string;
            description
              "This is application object.";
          }
          leaf-list application-group {
            type string;
            description
              "This is application group.";
          }
          leaf-list application-label {
            type string;
            description
              "This is application label.";
          }
          container category {
            description
              "TBD";
            list application-category {
              key "name application-subcategory";
              description
                "TBD";
              leaf name {
                type string;
                description
```

```
                "This is name for application category.";
              }
              leaf application-subcategory {
                type string;
                description
                 "This is application subcategory.";
              }
            }
          }
        }

        container target-condition {
          description
            "TBD";
          leaf target-description {
            type string;
            description
              "This is description for target condition.
              Vendors can write instructions for target condition
              that vendor made";
          }

          container device-sec-context-cond {
            description
              "The device attribute that can identify a device,
               including the device type (i.e., router, switch,
               pc, ios, or android) and the device's owner as
               well.";

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

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

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

            leaf tablet {
```

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


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

          leaf vehicle {
            type boolean;
            description
              "If type of a device is vehicle.";
          }
        }
      }
      container users-condition {
        description
          "TBD";
        leaf users-description {
          type string;
          description
            "This is description for user condition.
            Vendors can write instructions for user condition
            that vendor made";
        }

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

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

```
            case tenant {
              description
                "Tenant information.";

              leaf tenant {
                type uint8;
                mandatory true;
                description
                  "User's tenant information.";
              }
            }

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

              leaf vn-id {
                type uint8;
                mandatory true;
                description
                  "User's VN-ID information.";
              }
            }
          }
        }
        container group {
          description
            "The user (or user group) information with which
             network flow is associated: The user has many
             attributes such as name, id, password, type,
             authentication mode and so on. Name/id is often
             used in the security policy to identify the user.
             Besides, NSF is aware of the IP address of the
             user provided by a unified user management system
             via network. Based on name-address association,
             NSF is able to enforce the security functions
             over the given user (or user group)";

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

            case tenant {
              description
                "Tenant information.";

              leaf tenant {
```

```
                type uint8;
                mandatory true;
                description
                  "User's tenant information.";
              }
            }

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

              leaf vn-id {
                type uint8;
                mandatory true;
                description
                  "User's VN-ID information.";
              }
            }
          }
        }
        leaf security-grup {
          type string;
          mandatory true;
          description
            "security-grup.";
        }
      }

      container url-category-condition {
        description
          "TBD";
        leaf url-category-description {
          type string;
          description
            "This is description for url category condition.
            Vendors can write instructions for context condition
            that vendor made";
        }

        leaf-list pre-defined-category {
          type string;
          description
            "This is pre-defined-category.";
        }
        leaf-list user-defined-category {
          type string;
          description
            "This user-defined-category.";
```

```
      }
    }

    container context-condition {
      description
        "TBD";
      leaf context-description {
        type string;
        description
          "This is description for context condition.
          Vendors can write instructions for context condition
          that vendor made";
      }
    }

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

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

        leaf-list src-geographic-location {
          type uint32;
          description
            "This is mapped to ip address. We can acquire
             source region through ip address stored the
             database.";
        }
        leaf-list dest-geographic-location {
          type uint32;
          description
            "This is mapped to ip address. We can acquire
             destination region through ip address stored
             the database.";
        }
```

```
          }
        }
      }
    }
    container action-clause-container {
      description "TBD";
      list action-clause-list {
      key eca-object-id;
      uses i2nsf-eca-object-type {
        refine entity-class {
          default ECA-ACTION-TYPE;
        }
      }
      description
        "An action is used to control and monitor aspects of
         flow-based NSFs when the event and condition clauses
         are satisfied. NSFs provide security functions by
         executing various Actions. Examples of I2NSF Actions
         include providing intrusion detection and/or protection,
         web and flow filtering, and deep packet inspection
         for packets and flows.";

        leaf rule-log {
          type boolean;
          description
            "True is enable
             False is not enable.";
        }
        leaf session-log {
          type boolean;
          description
            "True is enable
             False is not enable.";
        }
        container ingress-action {
          description
            "TBD";
          leaf ingress-description {
            type string;
            description
              "This is description for ingress action.
               Vendors can write instructions for ingress action
               that vendor made";
          }
          leaf ingress-action-type {
            type ingress-action;
            description
              "Ingress action type: permit, deny, and mirror.";
```

```
        }
      }
      container egress-action {
        description
          "TBD";
        leaf egress-description {
          type string;
          description
            "This is description for egress action.
            Vendors can write instructions for egress action
            that vendor made";
        }
        leaf egress-action-type {
          type egress-action;
          description
            "Egress-action-type: invoke-signaling,
             tunnel-encapsulation, and forwarding.";
        }
      }

      container apply-profile {
        description
          "TBD";
        leaf profile-description {
          type string;
          description
            "This is description for apply profile action.
            Vendors can write instructions for apply
            profile action that vendor made";
        }

          container content-security-control {
            description
             "Content security control is another category of
             security capabilities applied to application layer.
             Through detecting the contents carried over the
             traffic in application layer, these capabilities
             can realize various security purposes, such as
             defending against intrusion, inspecting virus,
             filtering malicious URL or junk email, and blocking
             illegal web access or data retrieval.";

            container content-security-control-types {
              description
               "Content Security types: Antivirus, IPS, IDS,
                url-filtering, data-filtering, mail-filtering,
                file-blocking, file-isolate, pkt-capture,
                application-control, and voip-volte.";
```

```
                leaf antivirus {
                    type string;
                    description
                      "Additional inspection of antivirus.";
                }

                leaf ips {
                    type string;
                    description
                      "Additional inspection of IPS.";
                }

                leaf ids {
                    type string;
                    description
                      "Additional inspection of IDS.";
                }

                leaf url-filtering {
                    type string;
                    description
                      "Additional inspection of URL filtering.";
                }

                leaf data-filtering {
                   type string;
                   description
                     "Additional inspection of data filtering.";
                }

                leaf mail-filtering {
                  type string;
                  description
                    "Additional inspection of mail filtering.";
                }

                leaf file-blocking {
                  type string;
                  description
                    "Additional inspection of file blocking.";
                }

                leaf file-isolate {
                  type string;
                  description
                    "Additional inspection of file isolate.";
                }
```

```
            leaf pkt-capture {
              type string;
              description
                "Additional inspection of packet capture.";
            }

            leaf application-control {
              type string;
              description
                "Additional inspection of app control.";
            }

            leaf voip-volte {
              type string;
              description
                "Additional inspection of VoIP/VoLTE.";
            }
          }
        }

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

            container ddos-attack {
              description
                "A distributed-denial-of-service (DDoS) is
                 where the attack source is more than one,
                 often thousands of unique IP addresses.";

              container ddos-attack-type {
                description
                  "DDoS-attack types: Network Layer
                  DDoS Attacks and Application Layer
                  DDoS Attacks.";

                container network-layer-ddos-attack {
                  description
                    "Network layer DDoS-attack.";
                  container network-layer-ddos-attack-type {
                    description
                      "Network layer DDoS attack types:
                       Syn Flood Attack, UDP Flood Attack,
                       ICMP Flood Attack, IP Fragment Flood,
                       IPv6 Related Attacks, and etc";
```

```
            leaf syn-flood {
              type string;
              description
                "Additional Inspection of
                 Syn Flood Attack.";
            }

            leaf udp-flood {
              type string;
              description
                "Additional Inspection of
                 UDP Flood Attack.";
            }

            leaf icmp-flood {
              type string;
              description
                "Additional Inspection of
                 ICMP Flood Attack.";
            }

            leaf ip-frag-flood {
              type string;
              description
                "Additional Inspection of
                 IP Fragment Flood.";
            }

            leaf ipv6-related {
              type string;
              description
                "Additional Inspection of
                 IPv6 Related Attacks.";
            }
          }
        }

        container app-layer-ddos-attack {
          description
            "Application layer DDoS-attack.";

          container app-ddos-attack-types {
            description
              "Application layer DDoS-attack types:
               Http Flood Attack, Https Flood Attack,
               DNS Flood Attack, and
               DNS Amplification Flood Attack,
               SSL DDoS Attack, and etc.";
```

```
                leaf http-flood {
                  type string;
                  description
                    "Additional Inspection of
                     Http Flood Attack.";
                }

                leaf https-flood {
                  type string;
                  description
                    "Additional Inspection of
                     Https Flood Attack.";
                }

                leaf dns-flood {
                  type string;
                  description
                    "Additional Inspection of
                     DNS Flood Attack.";
                }

                leaf dns-amp-flood {
                  type string;
                  description
                    "Additional Inspection of
                     DNS Amplification Flood Attack.";
                }

                leaf ssl-ddos {
                  type string;
                  description
                    "Additional Inspection of
                     SSL Flood Attack.";
                }
              }
            }
          }
        }

        container single-packet-attack {
          description
            "Single Packet Attacks.";
          container single-packet-attack-type {
            description
              "DDoS-attack types: Scanning Attack,
               Sniffing Attack, Malformed Packet Attack,
               Special Packet Attack, and etc.";
```

```
                container scan-and-sniff-attack {
                  description
                    "Scanning and Sniffing Attack.";
                  container scan-and-sniff-attack-types {
                    description
                      "Scanning and sniffing attack types:
                       IP Sweep attack, Port Scanning,
                       and etc.";

                    leaf ip-sweep {
                      type string;
                      description
                        "Additional Inspection of
                         IP Sweep Attack.";
                    }

                    leaf port-scanning {
                      type string;
                      description
                        "Additional Inspection of
                         Port Scanning Attack.";
                    }
                  }
                }

                container malformed-packet-attack {
                  description
                    "Malformed Packet Attack.";
                  container malformed-packet-attack-types {
                    description
                      "Malformed packet attack types:
                       Ping of Death Attack, Teardrop Attack,
                       and etc.";

                    leaf ping-of-death {
                      type string;
                      description
                        "Additional Inspection of
                         Ping of Death Attack.";
                    }

                    leaf teardrop {
                      type string;
                      description
                        "Additional Inspection of
                         Teardrop Attack.";
                    }
                  }
```

```
                        }

                        container special-packet-attack {
                          description
                            "special Packet Attack.";
                          container special-packet-attack-types {
                            description
                              "Special packet attack types:
                               Oversized ICMP Attack, Tracert Attack,
                               and etc.";

                            leaf oversized-icmp {
                              type string;
                              description
                                "Additional Inspection of
                                 Oversize ICMP Attack.";
                            }

                            leaf tracert {
                              type string;
                              description
                                "Additional Inspection of
                                 Tracrt Attack.";
                            }
                          }
                        }
                      }
                    }
                  }
              }
          }
      }
    }
  }
```

<CODE ENDS>

Figure 5: YANG Data Module of I2NSF NSF-Facing-Interface

7. Security Considerations

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

8. Acknowledgments

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

Technology Development for the Customized Security Service Provisioning).

9. Contributors

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

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

10. References

10.1. Normative References

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

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

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

10.2. Informative References

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

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

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

Appendix A. Changes from draft-ietf-i2nsf-nsf-facing-interface-dm-01

The following changes are made from draft-ietf-i2nsf-nsf-facing-interface-dm-00:

1. We added rule enable, session aging time, and long connection attributes.

2. We added a rule group attribute.

3. We added additional conditions such as application and url.

4. We replaced manual to description to clarify the meaning.

Authors' Addresses

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

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

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

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

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 34129
Republic of Korea

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

EMail: linqiushi@huawei.com