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

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2018.
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 models, 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. Objectives

4.1. I2NSF Security Policy Rule

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

4.2. Event 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. These objects are defined as user security event, device security event, system security event, and time security event. These objects can be extended according to specific vendor event features.

4.3. Condition 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 user security event, device
security event, system security event, and time security event.
These objects are defined as packet security condition, packet
payload security condition, target security condition, user security
condition, context condition, and generic context condition. These
objects can be extended according to specific vendor condition
features.

4.4. Action Caluse

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

5. Data Model Structure

This section shows an following mapped features of a data model
structure tree of generic network security functions, as defined in
the [i2nsf-nsf-cap-im].

- 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 identification of network security policy has the
following structure:
The data model for event rule has the following structure:
module: ietf-i2nsf-nsf-facing-interface
+-rw generic-nsf
  +-rw i2nsf-security-policy* [policy-name]
  ...  
  +-rw eca-policy-rules* [rule-id]
  ...  
  +-rw resolution-strategy
  ...  
  +-rw default-action
  ...
+-rw event-clause-container
  +-rw event-clause-list* [eca-object-id]
    +-rw entity-class? identityref
    +-rw eca-object-id string
    +-rw manual? string
    +-rw sec-event-content string
    +-rw sec-event-format sec-event-format
    +-rw sec-event-type string
  +-rw condition-clause-container
    ...  
  +-rw action-clause-container
  ...

Figure 2: Data Model Structure for Event Rule

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

5.3. Condition Clause

The data model for condition rule has the following structure:

module: ietf-i2nsf-nsf-facing-interface
+-rw generic-nsf
  +-rw i2nsf-security-policy* [policy-name]
  ...  
  +-rw eca-policy-rules* [rule-id]
  ...  
  +-rw resolution-strategy
  ...  
  +-rw default-action
  ...
  +-rw event-clause-container
  ...
  +-rw condition-clause-container
++-rw condition-clause-list* [eca-object-id]
  +++-rw entity-class? identityref
  +++-rw eca-object-id string
  +++-rw (condition-type)?
  +++: (packet-security-condition)
      +=-rw packet-manual? string
      +=-rw packet-security-mac-condition
         +=-rw pkt-sec-cond-mac-dest* yang:phys-address
         +=-rw pkt-sec-cond-mac-src* yang:phys-address
         +=-rw pkt-sec-cond-mac-8021q* string
         +=-rw pkt-sec-cond-mac-ether-type* string
         +=-rw pkt-sec-cond-mac-tci* string
      +=-rw packet-security-ipv4-condition
         +=-rw pkt-sec-cond-ipv4-header-length* uint8
         +=-rw pkt-sec-cond-ipv4-tos* uint8
         +=-rw pkt-sec-cond-ipv4-total-length* uint16
         +=-rw pkt-sec-cond-ipv4-id* uint8
         +=-rw pkt-sec-cond-ipv4-fragment* uint8
         +=-rw pkt-sec-cond-ipv4-fragment-offset* uint16
         +=-rw pkt-sec-cond-ipv4-ttl* uint8
         +=-rw pkt-sec-cond-ipv4-protocol* uint8
         +=-rw pkt-sec-cond-ipv4-src* inet:ipv4-address
         +=-rw pkt-sec-cond-ipv4-dest* inet:ipv4-address
         +=-rw pkt-sec-cond-ipv4-iptopt? string
         +=-rw pkt-sec-cond-ipv4-sameip? boolean
         +=-rw pkt-sec-cond-ipv4-geoipt? string
      +=-rw packet-security-ipv6-condition
         +=-rw pkt-sec-cond-ipv6-dscp* string
         +=-rw pkt-sec-cond-ipv6-ecn* string
         +=-rw pkt-sec-cond-ipv6-traffic-class* uint8
         +=-rw pkt-sec-cond-ipv6-flow-label* uint32
         +=-rw pkt-sec-cond-ipv6-payload-length* uint16
         +=-rw pkt-sec-cond-ipv6-next-header* uint8
         +=-rw pkt-sec-cond-ipv6-hop-limit* uint8
         +=-rw pkt-sec-cond-ipv6-src* inet:ipv6-address
         +=-rw pkt-sec-cond-ipv6-dest* inet:ipv6-address
      +=-rw packet-security-tcp-condition
         +=-rw pkt-sec-cond-tcp-seq-num* uint32
         +=-rw pkt-sec-cond-tcp-ack-num* uint32
         +=-rw pkt-sec-cond-tcp-window-size* uint16
         +=-rw pkt-sec-cond-tcp-flags* uint8
      +=-rw packet-security-udp-condition
         +=-rw pkt-sec-cond-udp-length* string
      +=-rw packet-security-icmp-condition
         +=-rw pkt-sec-cond-icmp-type* uint8
         +=-rw pkt-sec-cond-icmp-code* uint8
         +=-rw pkt-sec-cond-icmp-seg-num* uint32
      +++: (packet-payload-condition)
These objects are defined as packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. These objects can be extended according to specific vendor condition features. We will add additional condition objects for more generic network security functions.
5.4. Action Clause

The data model for action rule has the following structure:

```
module: ietf-i2nsf-nsf-facing-interface
  +--rw generic-nsf
    |  +--rw i2nsf-security-policy* [policy-name]
    |     ...
    |     +--rw eca-policy-rules* [rule-id]
    |     ...
    |     +--rw resolution-strategy
    |     ...
    |     +--rw default-action
    |     ...
    +--rw event-clause-container
    |     ...
    +--rw condition-clause-container
    |     ...
    +--rw action-clause-container
    +--rw action-clause-list* [eca-object-id]
      +--rw entity-class? identityref
      +--rw eca-object-id string
      +--rw (action-type)?
        +--:(ingress-action)
          |  +--rw ingress-manual? string
          |  +--rw ingress-action-type? ingress-action
        +--:(egress-action)
          |  +--rw egress-manual? string
          |  +--rw egress-action-type? egress-action
        +--:(apply-profile)
          +--rw profile-manual? string
          +--rw (apply-profile-action-type)?
            +--:(content-security-control)
              |  +--rw content-security-control-types
              |     +--rw antivirus? boolean
              |     +--rw ips? boolean
              |     +--rw ids? boolean
              |     +--rw url-filtering? boolean
              |     +--rw data-filtering? boolean
              |     +--rw mail-filtering? boolean
              |     +--rw file-blocking? boolean
              |     +--rw file-isolate? boolean
              |     +--rw pkt-capture? boolean
              |     +--rw application-control? boolean
              |     +--rw voip-volte? boolean
            +--:(attack-mitigation-control)
              +--rw (attack-mitigation-control-type)?
```
These objects are defined as ingress action, egress action, and apply profile action. These objects can be extended according to specific vendor action feature. We will add additional action objects for more generic network security functions.

6. YANG Module

6.1. IETF NSF-Facing Interface YANG Data Module

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

<CODE BEGINS> file "ietf-i2nsf-nsf-facing-interface@2017-10-30.yang"
module ietf-i2nsf-nsf-facing-interface {
  yang-version 1.1;
}
namespace
prefix
  nsf-facing-interface;

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

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

contact
  "WG Web: <http://tools.ietf.org/wg/i2nsf>
  WG List: <mailto:i2nsf@ietf.org>
  WG Chair: Adrian Farrel
  <mailto:Adrain@olddog.co.uk>
  WG Chair: Linda Dunbar
  <mailto:Linda.duhbar@huawei.com>
  Editor: Jingyong Tim Kim
  <mailto:timkim@skku.edu>
  Editor: Jaehoon Paul Jeong
  <mailto:pauljeong@skku.edu>
  Editor: Susan Hares
  <mailto:shares@ndzh.com>";

description
  "This module defines a YANG data module for network security functions.";
revision "2017-10-30"{
  description "The third revision";
  reference
    "draft-ietf-i2nsf-capability-00";
}

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

description
  "This is used for SecEventFormat.";
}
typedef ingress-action {
  type enumeration {
    enum pass {
      description
        "If ingress action is pass";
    }
    enum drop {
      description
        "If ingress action is drop";
    }
    enum reject {
      description
        "If ingress action is reject";
    }
    enum alert {
      description
"If ingress action is alert";
}
enum mirror {
    description
    "If ingress action is mirror";
}
}
description
"This is used for ingress action."
}
typedef egress-action {
    type enumeration {
        enum invoke-signaling {
            description
            "If egress action is invoke signaling";
        }
        enum tunnel-encapsulation {
            description
            "If egress action is tunnel encapsulation";
        }
        enum forwarding {
            description
            "If egress action is forwarding";
        }
        enum redirection {
            description
            "If egress action is redirection";
        }
    }
    description
    "This is used for egress action."
}

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

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

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

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

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

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

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

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

grouping i2nsf-event-type {
  description "TBD";
  leaf manual {
    type string;
    description
      "This is manual for event. Vendors can write instructions for event that vendor made";
  }
}
leaf sec-event-content {
  type string;
  mandatory true;
  description
    "This is a mandatory string that contains the content
    of the SecurityEvent. The format of the content
    is specified in the SecEventFormat class
    attribute, and the type of event is defined in the
    SecEventType class attribute. An example of the
    SecEventContent attribute is a string hrAdmin,
    with the SecEventFormat set to 1 (GUID) and the
    SecEventType attribute set to 5 (new logon).");
}

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

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

list i2nsf-security-policy {
    key "policy-name";
    description
        "policy is a list
            including a set of security rules according to certain logic,
            i.e., their similarity or mutual relations, etc. The network
            security policy is able to apply over both the unidirectional
            and bidirectional traffic across the NSF.";

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

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

list eca-policy-rules {
    key "rule-id";
    description
        "This is a rule for network security functions.";

    leaf rule-id {
        type uint8;
        mandatory true;
        description
            "The id of the rule.
                This must be unique.";
    }
}
leaf rule-description {
  type string;
  description
    "This description gives more information about rules.";
}

leaf rule-rev {
  type uint8;
  description
    "This shows rule version.";
}

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

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

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

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

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

choice resolution-strategy-type {
  description
  "Vendors can use YANG data model to configure rules";
  case fmr {
    leaf first-matching-rule {
      type boolean;
      description
      "If the resolution strategy is first matching rule";
    }
  }
  case lmr {
    leaf last-matching-rule {
      type boolean;
      description
      "If the resolution strategy is last matching rule";
    }
  }
}

container default-action {
  description
  "This default action can be used to specify a predefined
  action when no other alternative action was matched
  by the currently executing I2NSF Policy Rule. An analogy
  is the use of a default statement in a C switch statement."
  leaf default-action-type {
    type ingress-action;
    description
    "Ingress action type: permit, deny, and mirror.";
  }
}

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

default ECA-EVENT-TYPE;
}
}

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

uses i2nsf-event-type;
}
}

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

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

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

  case packet-security-condition {
    leaf packet-manual {
      type string;
      description "This is manual for packet condition.";
    }
  }

Vendors can write instructions for packet condition that vendor made;
}

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

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

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

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

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

  leaf-list pkt-sec-cond-mac-tci {
    type string;
    description "This is an optional string attribute, and defines the Tag Control Information. This consists of a 3-bit user priority field, a drop eligible indicator (1 bit), and a VLAN identifier (12 bits).";
  }
}
container packet-security-ipv4-condition {
  description
  "The purpose of this Class is to represent IPv4 packet header information that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be executed or not."

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

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

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

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

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

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

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

leaf-list pkt-sec-cond-ipv4-protocol {
  type uint8;
  description
    "Internet Protocol version 4(IPv4) is the fourth
version of the Internet Protocol (IP)."
}

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

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

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

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

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

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

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

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

leaf-list pkt-sec-cond-ipv6-traffic-class {
type uint8;
description
"The bits of this field hold two values. The 6
most-significant bits are used for
differentiated services, which is used to
classify packets."
leaf-list pkt-sec-cond-ipv6-flow-label {
  type uint32;
  description
    "The flow label when set to a non-zero value
    serves as a hint to routers and switches
    with multiple outbound paths that these
    packets should stay on the same path so that
    they will not be reordered.";
}

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

leaf-list pkt-sec-cond-ipv6-next-header {
  type uint8;
  description
    "Specifies the type of the next header.
    This field usually specifies the transport
    layer protocol used by a packet’s payload.";
}

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

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

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

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

leaf-list pkt-sec-cond-tcp-seq-num {
  type uint32;
  description
    "If the SYN flag is set (1), then this is the initial sequence number.";
}

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

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

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

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

  leaf-list pkt-sec-cond-udp-length {

Kim, et al. Expires May 3, 2018
type string;
description
"This is a mandatory string attribute, and defines
the length in bytes of the UDP header and data
(16 bits).";
}
}

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

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

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

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

case packet-payload-condition {
    leaf packet-payload-manual {
        type string;
description
        "This is manual for payload condition.
Vendors can write instructions for payload condition
that vendor made";
    }
}

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

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

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

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

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

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

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

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

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

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

case tenant {
    description
    "Tenant information.";
}

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

case vn-id {
    description
    "Vendor's network identifier.";
}

leaf vn-id {
  type uint8;
  mandatory true;
  description
    "User’s VN-ID information.";
}

type uint8;
mandatory true;
description
  "User’s VN-ID information.";

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

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

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

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

  leaf-list dest-geographic-location {
    type uint32;
    description
    "This is mapped to ip address. We can acquire destination region through ip address stored the database.";
  }
}
container action-clause-container {
  description "TBD";
  list action-clause-list {
    key eca-object-id;
    uses i2nsf-eca-object-type {
      refine entity-class {
        default ECA-ACTION-TYPE;
      }
    }
  }
}

description
"An action is used to control and monitor aspects of flow-based NSFs when the event and condition clauses are satisfied. NSFs provide security functions by executing various Actions. Examples of I2NSF Actions include providing intrusion detection and/or protection, web and flow filtering, and deep packet inspection for packets and flows.";

choice action-type {
  description
    "Vendors can use YANG data model to configure rules by concreting this action type";
  case ingress-action {
    leaf ingress-manual {
      type string;
      description
        "This is manual for ingress action. Vendors can write instructions for ingress action that vendor made";
    }
    leaf ingress-action-type {
      type ingress-action;
      description
        "Ingress action type: permit, deny, and mirror.";
    }
  }
  case egress-action {
    leaf egress-manual {
      type string;
      description
        "This is manual for egress action. Vendors can write instructions for egress action";
leaf egress-action-type {
    type egress-action;
    description
        "Egress-action-type: invoke-signaling, tunnel-encapsulation, and forwarding.";
}

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

case content-security-control {
    container content-security-control-types {
        description
            "Content Security types: Antivirus, IPS, IDS, url-filtering, data-filtering, mail-filtering, file-blocking, file-isolate, pkt-capture, application-control, and voip-volte.";
        leaf antivirus {
            type boolean;
            description
                "Additional inspection of antivirus.";
        }
        leaf ips {
            type string;
            description
                "This is manual for apply profile action. Vendors can write instructions for apply profile action that vendor made";
        }
    }
    description
        "Advanced action types: Content Security Control and Attack Mitigation Control.";
}

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

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

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

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

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

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

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

leaf application-control {

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

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

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

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

container network-layer-ddos-attack {
   description
   "Network layer DDoS-attack.";
}

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

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

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

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

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

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

container app-ddos-attack-types {
  description
}

leaf http-flood {

type boolean;
description
"Additional Inspection of
Http Flood Attack."
;
}

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

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

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

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

}

}

}

}

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

container scan-and-sniff-attack {


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

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

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

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

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

    leaf teardrop {
        type boolean;
        description
            "Additional Inspection of
            Teardrop Attack.";
    }
}
container special-packet-attack {
    description
    "special Packet Attack.";
    container special-packet-attack-types {
        description
        "Special packet attack types:
        Oversized ICMP Attack, Tracert Attack,
        and etc.";

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

        leaf tracert {
            type boolean;
            description
            "Additional Inspection of
            Tracert Attack.";
        }
    }
}

<CODE ENDS>

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

7. Security Considerations

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

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

9. Contributors

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

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

10. References

10.1. Normative References


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]

[supa-policy-info-model]
Appendix A. draft-kim-i2nsf-nsf-facing-interface-data-model-03

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

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

2. Resolution Strategy mechanism is added to specify how to resolve conflicts that occur between the actions of the same or different policy rules that are matched and contained in this particular NSF.

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

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

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

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

Authors' Addresses

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

Phone: +82 10 8273 0930
EMail: timkim@skku.edu
Jaehoon Paul Jeong  
Department of Software  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 16419  
Republic of Korea  
Phone: +82 31 299 4957  
Fax: +82 31 290 7996  
EMail: pauljeong@skku.edu  
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Jung-Soo Park  
Electronics and Telecommunications Research Institute  
218 Gajeong-Ro, Yuseong-Gu  
Daejeon 34129  
Republic of Korea  
Phone: +82 42 860 6514  
EMail: pjs@etri.re.kr

Susan Hares  
Huawei  
7453 Hickory Hill  
Saline, MI 48176  
USA  
Phone: +1-734-604-0332  
EMail: shares@ndzh.com

Qiushi Lin  
Huawei  
Huawei Industrial Base  
Shenzhen, Guangdong 518129  
China  
EMail: linqiushi@huawei.com