

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

D. Hong
J. Jeong
J. Kim
Sungkyunkwan University
S. Hares
L. Xia
Huawei
H. Birkholz
Fraunhofer SIT
July 2, 2018

YANG Data Model for Monitoring I2NSF Network Security Functions
draft-hong-i2nsf-nsf-monitoring-data-model-04

Abstract

This document proposes a YANG data model for monitoring Network Security Functions (NSFs) in the Interface to Network Security Functions (I2NSF) system. If the monitoring of NSFs is performed in a comprehensive way, it is possible to detect the indication of malicious activity, anomalous behavior or the potential sign of denial of service attacks in a timely manner. This monitoring functionality is based on the monitoring information that is generated by NSFs. Thus, this document describes not only a data tree to specify an information model for monitoring NSFs, but also the corresponding YANG data model for monitoring NSFs.

Status of This Memo

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	2
3. Terminology	3
3.1. Tree Diagrams	3
4. Information Model Structure	3
5. YANG Data Model	11
6. Acknowledgments	45
7. References	46
7.1. Normative References	46
7.2. Informative References	46
Appendix A. Changes from draft-hong-i2nsf-nsf-monitoring-data-model-03	47
Authors' Addresses	47

1. Introduction

This document defines a YANG [RFC6020] data model for monitoring Network Security Functions (NSFs). This monitoring means the acquisition of vital information about NSFs via notifications, events, records or counters. The data model for the monitoring presented in this document is derived from the information model for monitoring NSFs through the NSF-Facing Interface specified in [i2nsf-monitoring-im].

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-terminology][i2nsf-framework]. Especially, the following terms are from [i2nsf-monitoring-im].

- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.
- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] is as follows:

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

4. Information Model Structure

Figure 1 shows the overview of a structure tree of monitoring information based on the [i2nsf-monitoring-im].

```

module: ietf-i2nsf-nsf-monitoring-dm
  +--rw counters
    +--rw system-interface
      | +--rw acquisition-method?      identityref
      | +--rw emission-type?          identityref
      | +--rw dampening-type?         identityref
      | +--rw interface-name?         string

```

```

    +--rw in-total-traffic-pkts?      uint32
    +--rw out-total-traffic-pkts?     uint32
    +--rw in-total-traffic-bytes?     uint32
    +--rw out-total-traffic-bytes?    uint32
    +--rw in-drop-traffic-pkts?       uint32
    +--rw out-drop-traffic-pkts?      uint32
    +--rw in-drop-traffic-bytes?      uint32
    +--rw out-drop-traffic-bytes?     uint32
    +--rw total-traffic?               uint32
    +--rw in-traffic-ave-rate?         uint32
    +--rw in-traffic-peak-rate?       uint32
    +--rw in-traffic-ave-speed?       uint32
    +--rw in-traffic-peak-speed?      uint32
    +--rw out-traffic-ave-rate?       uint32
    +--rw out-traffic-peak-rate?      uint32
    +--rw out-traffic-ave-speed?      uint32
    +--rw out-traffic-peak-speed?     uint32
    +--rw message?                    string
    +--rw time-stamp?                 yang:date-and-time
    +--rw vendor-name?                string
    +--rw nsf-name?                   string
    +--rw module-name?                string
    +--rw severity?                   severity
+--rw nsf-firewall
    +--rw acquisition-method?          identityref
    +--rw emission-type?               identityref
    +--rw dampening-type?              identityref
    +--rw src-ip?                      inet:ipv4-address
    +--rw dst-ip?                      inet:ipv4-address
    +--rw src-port?                    inet:port-number
    +--rw dst-port?                    inet:port-number
    +--rw src-zone?                    string
    +--rw dst-zone?                    string
    +--rw src-region?                  string
    +--rw dst-region?                  string
    +--rw policy-id?                   uint8
    +--rw policy-name?                 string
    +--rw src-user?                    string
    +--rw protocol?                    identityref
    +--rw app?                         string
    +--rw total-traffic?               uint32
    +--rw in-traffic-ave-rate?         uint32
    +--rw in-traffic-peak-rate?       uint32
    +--rw in-traffic-ave-speed?       uint32
    +--rw in-traffic-peak-speed?      uint32
    +--rw out-traffic-ave-rate?       uint32
    +--rw out-traffic-peak-rate?      uint32
    +--rw out-traffic-ave-speed?      uint32

```

```

|   +---rw out-traffic-peak-speed?   uint32
+---rw nsf-policy-hits
|   +---rw acquisition-method?       identityref
|   +---rw emission-type?            identityref
|   +---rw dampening-type?           identityref
|   +---rw src-ip?                   inet:ipv4-address
|   +---rw dst-ip?                   inet:ipv4-address
|   +---rw src-port?                 inet:port-number
|   +---rw dst-port?                 inet:port-number
|   +---rw src-zone?                 string
|   +---rw dst-zone?                 string
|   +---rw src-region?               string
|   +---rw dst-region?               string
|   +---rw policy-id?                uint8
|   +---rw policy-name?              string
|   +---rw src-user?                 string
|   +---rw protocol?                 identityref
|   +---rw app?                      string
|   +---rw message?                  string
|   +---rw time-stamp?               yang:date-and-time
|   +---rw vendor-name?              string
|   +---rw nsf-name?                 string
|   +---rw module-name?              string
|   +---rw severity?                 severity
|   +---rw hit-times?                uint32
notifications:
+---n system-detection-alarm
|   +---ro alarm-catagory?            identityref
|   +---ro acquisition-method?        identityref
|   +---ro emission-type?             identityref
|   +---ro dampening-type?            identityref
|   +---ro usage?                     uint8
|   +---ro threshold?                 uint8
|   +---ro message?                   string
|   +---ro time-stamp?                yang:date-and-time
|   +---ro vendor-name?               string
|   +---ro nsf-name?                  string
|   +---ro module-name?               string
|   +---ro severity?                  severity
+---n system-detection-event
|   +---ro event-catagory?            identityref
|   +---ro acquisition-method?        identityref
|   +---ro emission-type?             identityref
|   +---ro dampening-type?            identityref
|   +---ro user                       string
|   +---ro group                      string
|   +---ro login-ip-addr              inet:ipv4-address

```

```

|   +--ro authentication?      identityref
|   +--ro message?            string
|   +--ro time-stamp?         yang:date-and-time
|   +--ro vendor-name?        string
|   +--ro nsf-name?           string
|   +--ro module-name?        string
|   +--ro severity?           severity
+---n nsf-detection-flood
|   +--ro event-name?          identityref
|   +--ro dst-ip?             inet:ipv4-address
|   +--ro dst-port?           inet:port-number
|   +--ro rule-id             uint8
|   +--ro rule-name           string
|   +--ro profile?            string
|   +--ro raw-info?           string
|   +--ro sub-attack-type?     identityref
|   +--ro start-time          yang:date-and-time
|   +--ro end-time            yang:date-and-time
|   +--ro attack-rate?        uint32
|   +--ro attack-speed?       uint32
|   +--ro message?            string
|   +--ro time-stamp?         yang:date-and-time
|   +--ro vendor-name?        string
|   +--ro nsf-name?           string
|   +--ro module-name?        string
|   +--ro severity?           severity
+---n nsf-detection-session-table
|   +--ro current-session?     uint8
|   +--ro maximum-session?     uint8
|   +--ro threshold?          uint8
|   +--ro message?            string
|   +--ro time-stamp?         yang:date-and-time
|   +--ro vendor-name?        string
|   +--ro nsf-name?           string
|   +--ro module-name?        string
|   +--ro severity?           severity
+---n nsf-detection-virus
|   +--ro src-ip?             inet:ipv4-address
|   +--ro dst-ip?             inet:ipv4-address
|   +--ro src-port?           inet:port-number
|   +--ro dst-port?           inet:port-number
|   +--ro src-zone?           string
|   +--ro dst-zone?           string
|   +--ro rule-id             uint8
|   +--ro rule-name           string
|   +--ro profile?            string
|   +--ro raw-info?           string
|   +--ro virus?              identityref

```

```

    +--ro virus-name?      string
    +--ro file-type?       string
    +--ro file-name?       string
    +--ro message?         string
    +--ro time-stamp?      yang:date-and-time
    +--ro vendor-name?     string
    +--ro nsf-name?        string
    +--ro module-name?     string
    +--ro severity?        severity
+---n nsf-detection-intrusion
    +--ro src-ip?          inet:ipv4-address
    +--ro dst-ip?          inet:ipv4-address
    +--ro src-port?        inet:port-number
    +--ro dst-port?        inet:port-number
    +--ro src-zone?        string
    +--ro dst-zone?        string
    +--ro rule-id          uint8
    +--ro rule-name        string
    +--ro profile?         string
    +--ro raw-info?        string
    +--ro protocol?        identityref
    +--ro app?             string
    +--ro sub-attack-type? identityref
    +--ro message?         string
    +--ro time-stamp?      yang:date-and-time
    +--ro vendor-name?     string
    +--ro nsf-name?        string
    +--ro module-name?     string
    +--ro severity?        severity
+---n nsf-detection-botnet
    +--ro src-ip?          inet:ipv4-address
    +--ro dst-ip?          inet:ipv4-address
    +--ro src-port?        inet:port-number
    +--ro dst-port?        inet:port-number
    +--ro src-zone?        string
    +--ro dst-zone?        string
    +--ro rule-id          uint8
    +--ro rule-name        string
    +--ro profile?         string
    +--ro raw-info?        string
    +--ro attack-type?     identityref
    +--ro protocol?        identityref
    +--ro botnet-name?     string
    +--ro role?            string
    +--ro message?         string
    +--ro time-stamp?      yang:date-and-time
    +--ro vendor-name?     string
    +--ro nsf-name?        string

```

```

|   +--ro module-name?   string
|   +--ro severity?      severity
+---n nsf-detection-web-attack
|   +--ro src-ip?        inet:ipv4-address
|   +--ro dst-ip?        inet:ipv4-address
|   +--ro src-port?      inet:port-number
|   +--ro dst-port?      inet:port-number
|   +--ro src-zone?      string
|   +--ro dst-zone?      string
|   +--ro rule-id        uint8
|   +--ro rule-name      string
|   +--ro profile?       string
|   +--ro raw-info?      string
|   +--ro sub-attack-type? identityref
|   +--ro request-method? identityref
|   +--ro req-uri?       string
|   +--ro uri-category?  string
|   +--ro filtering-type* identityref
|   +--ro message?       string
|   +--ro time-stamp?    yang:date-and-time
|   +--ro vendor-name?   string
|   +--ro nsf-name?      string
|   +--ro module-name?   string
|   +--ro severity?      severity
+---n system-access-log
|   +--ro login-ip        inet:ipv4-address
|   +--ro administrator?  string
|   +--ro login-mode?     login-mode
|   +--ro operation-type? operation-type
|   +--ro result?         string
|   +--ro content?        string
|   +--ro acquisition-method? identityref
|   +--ro emission-type?  identityref
|   +--ro dampening-type? identityref
+---n system-res-util-log
|   +--ro system-status?  string
|   +--ro cpu-usage?      uint8
|   +--ro memory-usage?   uint8
|   +--ro disk-usage?     uint8
|   +--ro disk-left?      uint8
|   +--ro session-num?    uint8
|   +--ro process-num?    uint8
|   +--ro in-traffic-rate? uint32
|   +--ro out-traffic-rate? uint32
|   +--ro in-traffic-speed? uint32
|   +--ro out-traffic-speed? uint32
|   +--ro acquisition-method? identityref
|   +--ro emission-type?  identityref

```



```

|   +---ro dampening-type?          identityref
+---n system-user-activity-log
|   +---ro acquisition-method?      identityref
|   +---ro emission-type?           identityref
|   +---ro dampening-type?          identityref
|   +---ro user                     string
|   +---ro group                    string
|   +---ro login-ip-addr            inet:ipv4-address
|   +---ro authentication?          identityref
|   +---ro access?                  identityref
|   +---ro online-duration?         string
|   +---ro logout-duration?         string
|   +---ro additional-info?         string
+---n nsf-log-ddos
|   +---ro attack-type?             identityref
|   +---ro attack-ave-rate?         uint32
|   +---ro attack-ave-speed?        uint32
|   +---ro attack-pkt-num?          uint32
|   +---ro attack-src-ip?           inet:ipv4-address
|   +---ro action?                  log-action
|   +---ro acquisition-method?      identityref
|   +---ro emission-type?           identityref
|   +---ro dampening-type?          identityref
|   +---ro message?                 string
|   +---ro time-stamp?              yang:date-and-time
|   +---ro vendor-name?             string
|   +---ro nsf-name?                string
|   +---ro module-name?             string
|   +---ro severity?                severity
+---n nsf-log-virus
|   +---ro attack-type?             identityref
|   +---ro action?                  log-action
|   +---ro os?                      string
|   +---ro time                     yang:date-and-time
|   +---ro acquisition-method?      identityref
|   +---ro emission-type?           identityref
|   +---ro dampening-type?          identityref
|   +---ro message?                 string
|   +---ro time-stamp?              yang:date-and-time
|   +---ro vendor-name?             string
|   +---ro nsf-name?                string
|   +---ro module-name?             string
|   +---ro severity?                severity
+---n nsf-log-intrusion
|   +---ro attack-type?             identityref
|   +---ro action?                  log-action
|   +---ro time                     yang:date-and-time
|   +---ro attack-rate?             uint32

```

```

    +---ro attack-speed?          uint32
    +---ro acquisition-method?    identityref
    +---ro emission-type?         identityref
    +---ro dampening-type?        identityref
    +---ro message?               string
    +---ro time-stamp?            yang:date-and-time
    +---ro vendor-name?           string
    +---ro nsf-name?              string
    +---ro module-name?           string
    +---ro severity?              severity
+---n nsf-log-botnet
    +---ro attack-type?           identityref
    +---ro action?                log-action
    +---ro botnet-pkt-num?        uint8
    +---ro os?                    string
    +---ro acquisition-method?    identityref
    +---ro emission-type?         identityref
    +---ro dampening-type?        identityref
    +---ro message?               string
    +---ro time-stamp?            yang:date-and-time
    +---ro vendor-name?           string
    +---ro nsf-name?              string
    +---ro module-name?           string
    +---ro severity?              severity
+---n nsf-log-dpi
    +---ro attack-type?           dpi-type
    +---ro acquisition-method?    identityref
    +---ro emission-type?         identityref
    +---ro dampening-type?        identityref
    +---ro src-ip?                inet:ipv4-address
    +---ro dst-ip?                inet:ipv4-address
    +---ro src-port?              inet:port-number
    +---ro dst-port?              inet:port-number
    +---ro src-zone?              string
    +---ro dst-zone?              string
    +---ro src-region?            string
    +---ro dst-region?            string
    +---ro policy-id?             uint8
    +---ro policy-name?           string
    +---ro src-user?              string
    +---ro protocol?              identityref
    +---ro app?                   string
    +---ro message?               string
    +---ro time-stamp?            yang:date-and-time
    +---ro vendor-name?           string
    +---ro nsf-name?              string
    +---ro module-name?           string
    +---ro severity?              severity

```

```

+---n nsf-log-vuln-scan
|   +--ro vulnerability-id?      uint8
|   +--ro victim-ip?            inet:ipv4-address
|   +--ro protocol?             identityref
|   +--ro port-num?            inet:port-number
|   +--ro level?               severity
|   +--ro os?                  string
|   +--ro vulnerability-info?   string
|   +--ro fix-suggestion?       string
|   +--ro service?             string
|   +--ro acquisition-method?   identityref
|   +--ro emission-type?       identityref
|   +--ro dampening-type?      identityref
|   +--ro message?             string
|   +--ro time-stamp?          yang:date-and-time
|   +--ro vendor-name?         string
|   +--ro nsf-name?            string
|   +--ro module-name?         string
|   +--ro severity?            severity
+---n nsf-log-web-attack
|   +--ro attack-type?          identityref
|   +--ro rsp-code?            string
|   +--ro req-clientapp?        string
|   +--ro req-cookies?         string
|   +--ro req-host?            string
|   +--ro raw-info?            string
|   +--ro acquisition-method?   identityref
|   +--ro emission-type?       identityref
|   +--ro dampening-type?      identityref
|   +--ro message?            string
|   +--ro time-stamp?          yang:date-and-time
|   +--ro vendor-name?         string
|   +--ro nsf-name?            string
|   +--ro module-name?         string
|   +--ro severity?            severity

```

Figure 1: Information Model for NSF Monitoring

5. YANG Data Model

This section introduces a YANG data model for the information model of monitoring information based on [i2nsf-monitoring-im].

```

<CODE BEGINS> file "ietf-i2nsf-nsf-monitoring-dm@2018-07-02.yang"
module ietf-i2nsf-nsf-monitoring-dm {
  yang-version 1.1;

```

```
namespace
  "urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring-dm";
prefix
  monitoring-information;
import ietf-inet-types {
  prefix inet;
}
import ietf-yang-types {
  prefix yang;
}
organization
  "IETF I2NSF (Interface to Network Security Functions)
  Working Group";
contact
  "WG Web: <http://tools.ietf.org/wg/i2nsf>
  WG List: <mailto:i2nsf@ietf.org>

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

  Editor: Dongjin Hong
  <mailto:dong.jin@skku.edu>

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

description
  "This module defines a YANG data module for monitoring NSFs.";

revision "2018-07-02" {
  description "Fifth revision";
  reference
    "draft-zhang-i2nsf-info-model-monitoring-06";
}

typedef severity {
  type enumeration {
    enum high {
      description
        "high-level";
    }
    enum middle {
      description
        "middle-level";
    }
    enum low {
      description
        "low-level";
    }
  }
}
```

```
    }  
  }  
  description  
    "An indicator representing severity";  
}  
typedef log-action {  
  type enumeration {  
    enum allow {  
      description  
        "If action is allow";  
    }  
    enum alert {  
      description  
        "If action is alert";  
    }  
    enum block {  
      description  
        "If action is block";  
    }  
    enum discard {  
      description  
        "If action is discard";  
    }  
    enum declare {  
      description  
        "If action is declare";  
    }  
    enum block-ip {  
      description  
        "If action is block-ip";  
    }  
    enum block-service {  
      description  
        "If action is block-service";  
    }  
  }  
  description  
    "This is used for protocol";  
}  
typedef dpi-type {  
  type enumeration {  
    enum file-blocking {  
      description  
        "DPI for blocking file";  
    }  
    enum data-filtering {  
      description  
        "DPI for filtering data";  
    }  
  }  
}
```

```
    }
    enum application-behavior-control{
        description
            "DPI for controlling application behavior";
    }
}
description
    "This is used for dpi type";
}
typedef operation-type{
    type enumeration {
        enum login{
            description
                "Login operation";
        }
        enum logout{
            description
                "Logout operation";
        }
        enum configuration{
            description
                "Configuration operation";
        }
    }
}
description
    "An indicator representing operation-type";
}
typedef login-mode{
    type enumeration {
        enum root{
            description
                "Root login-mode";
        }
        enum user{
            description
                "User login-mode";
        }
        enum guest{
            description
                "Guest login-mode";
        }
    }
}
description
    "An indicator representing login-mode";
}

identity characteristics {
    description
```

```
    "Base identity for monitoring information
    characteristics";
}
identity acquisition-method {
    base characteristics;
    description
        "The type of acquisition-method. Can be multiple types at once.";
}
identity subscription {
    base acquisition-method;
    description
        "The acquisition-method type is subscription";
}
identity query {
    base acquisition-method;
    description
        "The acquisition-method type is query";
}
identity emission-type {
    base characteristics;
    description
        "The type of emission-type.";
}
identity periodical {
    base emission-type;
    description
        "The emission-type type is periodical.";
}
identity on-change {
    base emission-type;
    description
        "The emission-type type is on-change.";
}
identity dampening-type {
    base characteristics;
    description
        "The type of dampening-type.";
}
identity no-dampening {
    base dampening-type;
    description
        "The dampening-type is no-dampening.";
}
identity on-repetition {
    base dampening-type;
    description
        "The dampening-type is on-repetition.";
}
```

```
identity none {
  base dampening-type;
  description
    "The dampening-type is none.";
}

identity authentication-mode {
  description
    "User authentication mode types: e.g., Local Authentication,
    Third-Party Server Authentication,
    Authentication Exemption, or SSO Authentication.";
}
identity local-authentication {
  base authentication-mode;
  description
    "Authentication-mode : local authentication.";
}
identity third-party-server-authentication {
  base authentication-mode;
  description
    "If authentication-mode is
    third-part-server-authentication";
}
identity exemption-authentication {
  base authentication-mode;
  description
    "If authentication-mode is
    exemption-authentication";
}
identity sso-authentication {
  base authentication-mode;
  description
    "If authentication-mode is
    sso-authentication";
}

identity alarm-type {
  description
    "Base identity for detectable alarm types";
}
identity MEM-USAGE-ALARM {
  base alarm-type;
  description
    "A memory alarm is alerted";
}
identity CPU-USAGE-ALARM {
  base alarm-type;
  description
```



```
    "A cpu alarm is alerted";
}
identity DISK-USAGE-ALARM {
    base alarm-type;
    description
        "A disk alarm is alerted";
}
identity HW-FAILURE-ALARM {
    base alarm-type;
    description
        "A hardware alarm is alerted";
}
identity IFNET-STATE-ALARM {
    base alarm-type;
    description
        "An interface alarm is alerted";
}
identity event-type {
    description
        "Base identity for detectable event types";
}
identity ACCESS-DENIED {
    base event-type;
    description
        "The system event is access-denied.";
}
identity CONFIG-CHANGE {
    base event-type;
    description
        "The system event is config-change.";
}

identity flood-type {
    description
        "Base identity for detectable flood types";
}
identity syn-flood {
    base flood-type;
    description
        "A SYN flood is detected";
}
identity ack-flood {
    base flood-type;
    description
        "An ACK flood is detected";
}
identity syn-ack-flood {
    base flood-type;
```

```
    description
      "An SYN-ACK flood is detected";
  }
  identity fin-rst-flood {
    base flood-type;
    description
      "A FIN-RST flood is detected";
  }
  identity tcp-con-flood {
    base flood-type;
    description
      "A TCP connection flood is detected";
  }
  identity udp-flood {
    base flood-type;
    description
      "A UDP flood is detected";
  }
  identity icmp-flood {
    base flood-type;
    description
      "An ICMP flood is detected";
  }
  identity https-flood {
    base flood-type;
    description
      "A HTTPS flood is detected";
  }
  identity http-flood {
    base flood-type;
    description
      "A HTTP flood is detected";
  }
  identity dns-reply-flood {
    base flood-type;
    description
      "A DNS reply flood is detected";
  }
  identity dns-query-flood {
    base flood-type;
    description
      "A DNS query flood is detected";
  }
  identity sip-flood {
    base flood-type;
    description
      "A SIP flood is detected";
  }
}
```

```
identity nsf-event-name {
  description
    "Base identity for detectable nsf event types";
}
identity SEC-EVENT-DDOS {
  base nsf-event-name;
  description
    "The nsf event is sec-event-ddos.";
}
identity SESSION-USAGE-HIGH {
  base nsf-event-name;
  description
    "The nsf event is session-usage-high";
}
identity SEC-EVENT-VIRUS {
  base nsf-event-name;
  description
    "The nsf event is sec-event-virus";
}
identity SEC-EVENT-INTRUSION {
  base nsf-event-name;
  description
    "The nsf event is sec-event-intrusion";
}
identity SEC-EVENT-BOTNET {
  base nsf-event-name;
  description
    "The nsf event is sec-event-botnet";
}
identity SEC-EVENT-WEBATTACK {
  base nsf-event-name;
  description
    "The nsf event is sec-event-webattack";
}
identity attack-type {
  description
    "The root ID of attack based notification
    in the notification taxonomy";
}
identity system-attack-type {
  base attack-type;
  description
    "This ID is intended to be used
    in the context of system events";
}
identity nsf-attack-type {
  base attack-type;
  description
```

```
    "This ID is intended to be used in the context of nsf event";
}
identity botnet-attack-type {
    base nsf-attack-type;
    description
        "This is a ID stub limited to indicating
        that this attack type is botnet.
        The usual semantic and taxonomy is missing
        and name is used.";
}
identity virus-type {
    base nsf-attack-type;
    description
        "The type of virus. Can be multiple types at once. This attack
        type is associated with a detected system-log virus-attack";
}
identity trojan {
    base virus-type;
    description
        "The detected virus type is trojan";
}
identity worm {
    base virus-type;
    description
        "The detected virus type is worm";
}
identity macro {
    base virus-type;
    description
        "The detected virus type is macro";
}
identity intrusion-attack-type {
    base nsf-attack-type;
    description
        "The attack type is associatied with
        a detectedsystem-log intrusion";
}
identity brute-force {
    base intrusion-attack-type;
    description
        "The intrusion type is brute-force";
}
identity buffer-overflow {
    base intrusion-attack-type;
    description
        "The intrusion type is buffer-overflow";
}
identity web-attack-type {
```

```
    base nsf-attack-type;
    description
        "The attack type associated with
        a detected system-log web-attack";
}
identity command-injection {
    base web-attack-type;
    description
        "The detected web attack type is command injection";
}
identity xss {
    base web-attack-type;
    description
        "The detected web attack type is XSS";
}
identity csrf {
    base web-attack-type;
    description
        "The detected web attack type is CSRF";
}
identity ddos-attack-type {
    base nsf-attack-type;
    description
        "The attack type is associated with a detected nsf-log event";
}

identity req-method {
    description
        "A set of request types (if applicable).
        For instance, PUT or GET in HTTP";
}
identity put-req {
    base req-method;
    description
        "The detected request type is PUT";
}
identity get-req {
    base req-method;
    description
        "The detected request type is GET";
}

identity filter-type {
    description
        "The type of filter used to detect, for example,
        a web-attack. Can be applicable to more than
        web-attacks. Can be more than one type.";
}
```

```
identity whitelist {
  base filter-type;
  description
    "The applied filter type is whitelist";
}
identity blacklist {
  base filter-type;
  description
    "The applied filter type is blacklist";
}
identity user-defined {
  base filter-type;
  description
    "The applied filter type is user-defined";
}
identity balicious-category {
  base filter-type;
  description
    "The applied filter is balicious category";
}
identity unknown-filter {
  base filter-type;
  description
    "The applied filter is unknown";
}

identity access-mode {
  description
    "Base identity for detectable access mode.";
}
identity ppp {
  base access-mode;
  description
    "Access-mode : ppp";
}
identity svn {
  base access-mode;
  description
    "Access-mode : svn";
}
identity local {
  base access-mode;
  description
    "Access-mode : local";
}

identity protocol-type {
  description
```

```
        "An identity used to enable type choices in leafs
        and leaflists wrt protocol metadata.";
    }
    identity tcp {
        base ipv4;
        base ipv6;
        description
            "TCP protocol type.";
    }
    identity udp {
        base ipv4;
        base ipv6;
        description
            "UDP protocol type.";
    }
    identity icmp {
        base ipv4;
        base ipv6;
        description
            "General ICMP protocol type.";
    }
    identity icmpv4 {
        base ipv4;
        description
            "ICMPv4 protocol type.";
    }
    identity icmpv6 {
        base ipv6;
        description
            "ICMPv6 protocol type.";
    }
    identity ip {
        base protocol-type;
        description
            "General IP protocol type.";
    }
    identity ipv4 {
        base ip;
        description
            "IPv4 protocol type.";
    }
    identity ipv6 {
        base ip;
        description
            "IPv6 protocol type.";
    }
    identity http {
        base tcp;
```

```
    description
      "HTTP protocol type.";
  }
  identity ftp {
    base tcp;
    description
      "FTP protocol type.";
  }
  grouping common-monitoring-data {
    description
      "The data set of common monitoring";
    leaf message {
      type string;
      description
        "This is a freetext annotation of
        monitoring notification content";
    }
    leaf time-stamp {
      type yang:date-and-time;
      description
        "Indicates the time of message generation";
    }
    leaf vendor-name {
      type string;
      description
        "The name of the NSF vendor";
    }
    leaf nsf-name {
      type string;
      description
        "The name (or IP) of the NSF
        generating the message";
    }
    leaf module-name {
      type string;
      description
        "The module name outputting the message";
    }
    leaf severity {
      type severity;
      description
        "The severity of the alarm such
        asvcritical, high, middle, low.";
    }
  }
  grouping characteristics{
    description
      "A set of monitoring information characteristics";
```



```
leaf acquisition-method {
  type identityref {
    base acquisition-method;
  }
  description
    "The acquisition-method for characteristics";
}
leaf emission-type {
  type identityref {
    base emission-type;
  }
  description
    "The emission-type for characteristics";
}
leaf dampening-type {
  type identityref {
    base dampening-type;
  }
  description
    "The dampening-type for characteristics";
}
}
grouping i2nsf-system-alarm-type-content {
  description
    "A set of system alarm type contents";
  leaf usage {
    type uint8;
    description
      "specifies the amount of usage";
  }
  leaf threshold {
    type uint8;
    description
      "The threshold triggering the alarm or the event";
  }
}
grouping i2nsf-system-event-type-content {
  description
    "System event metadata associated with system events caused
    by user activity.";
  leaf user {
    type string;
    mandatory true;
    description
      "Name of a user";
  }
  leaf group {
    type string;
  }
}
```

```
        mandatory true;
        description
            "Group to which a user belongs.";
    }
    leaf login-ip-addr {
        type inet:ipv4-address;
        mandatory true;
        description
            "Login IP address of a user.";
    }
    leaf authentication {
        type identityref {
            base authentication-mode;
        }
        description
            "The authentication-mode for authentication";
    }
}
grouping i2nsf-nsf-event-type-content-extend {
    description
        "A set of common IPv4-related NSF event
        content elements";
    leaf src-ip {
        type inet:ipv4-address;
        description
            "The source IP address of the packet";
    }
    leaf dst-ip {
        type inet:ipv4-address;
        description
            "The destination IP address of the packet";
    }
    leaf src-port {
        type inet:port-number;
        description
            "The source port of the packet";
    }
    leaf dst-port {
        type inet:port-number;
        description
            "The destination port of the packet";
    }
    leaf src-zone {
        type string;
        description
            "The source security zone of the packet";
    }
    leaf dst-zone {
```

```
        type string;
        description
            "The destination security zone of the packet";
    }
    leaf rule-id {
        type uint8;
        mandatory true;
        description
            "The ID of the rule being triggered";
    }
    leaf rule-name {
        type string;
        mandatory true;
        description
            "The name of the rule being triggered";
    }
    leaf profile {
        type string;
        description
            "Security profile that traffic matches.";
    }
    leaf raw-info {
        type string;
        description
            "The information describing the packet
            triggering the event.";
    }
}
grouping i2nsf-nsf-event-type-content {
    description
        "A set of common IPv4-related NSF event
        content elements";
    leaf dst-ip {
        type inet:ipv4-address;
        description
            "The destination IP address of the packet";
    }
    leaf dst-port {
        type inet:port-number;
        description
            "The destination port of the packet";
    }
    leaf rule-id {
        type uint8;
        mandatory true;
        description
            "The ID of the rule being triggered";
    }
}
```

```
    leaf rule-name {
      type string;
      mandatory true;
      description
        "The name of the rule being triggered";
    }
    leaf profile {
      type string;
      description
        "Security profile that traffic matches.";
    }
    leaf raw-info {
      type string;
      description
        "The information describing the packet
        triggering the event.";
    }
  }
  grouping traffic-rates {
    description
      "A set of traffic rates
      for statistics data";
    leaf total-traffic {
      type uint32;
      description
        "Total traffic";
    }
    leaf in-traffic-ave-rate {
      type uint32;
      description
        "Inbound traffic average rate in pps";
    }
    leaf in-traffic-peak-rate {
      type uint32;
      description
        "Inbound traffic peak rate in pps";
    }
    leaf in-traffic-ave-speed {
      type uint32;
      description
        "Inbound traffic average speed in bps";
    }
    leaf in-traffic-peak-speed {
      type uint32;
      description
        "Inbound traffic peak speed in bps";
    }
    leaf out-traffic-ave-rate {
```

```
        type uint32;
        description
            "Outbound traffic average rate in pps";
    }
    leaf out-traffic-peak-rate {
        type uint32;
        description
            "Outbound traffic peak rate in pps";
    }
    leaf out-traffic-ave-speed {
        type uint32;
        description
            "Outbound traffic average speed in bps";
    }
    leaf out-traffic-peak-speed {
        type uint32;
        description
            "Outbound traffic peak speed in bps";
    }
}
grouping i2nsf-system-counter-type-content{
    description
        "A set of system counter type contents";
    leaf interface-name {
        type string;
        description
            "Network interface name configured in NSF";
    }
    leaf in-total-traffic-pkts {
        type uint32;
        description
            "Total inbound packets";
    }
    leaf out-total-traffic-pkts {
        type uint32;
        description
            "Total outbound packets";
    }
    leaf in-total-traffic-bytes {
        type uint32;
        description
            "Total inbound bytes";
    }
    leaf out-total-traffic-bytes {
        type uint32;
        description
            "Total outbound bytes";
    }
}
```

```
    leaf in-drop-traffic-pkts {
      type uint32;
      description
        "Total inbound drop packets";
    }
    leaf out-drop-traffic-pkts {
      type uint32;
      description
        "Total outbound drop packets";
    }
    leaf in-drop-traffic-bytes {
      type uint32;
      description
        "Total inbound drop bytes";
    }
    leaf out-drop-traffic-bytes {
      type uint32;
      description
        "Total outbound drop bytes";
    }
    uses traffic-rates;
  }
  grouping i2nsf-nsf-counters-type-content {
    description
      "A set of nsf counters type contents";
    leaf src-ip {
      type inet:ipv4-address;
      description
        "The source IP address of the packet";
    }
    leaf dst-ip {
      type inet:ipv4-address;
      description
        "The destination IP address of the packet";
    }
    leaf src-port {
      type inet:port-number;
      description
        "The source port of the packet";
    }
    leaf dst-port {
      type inet:port-number;
      description
        "The destination port of the packet";
    }
    leaf src-zone {
      type string;
      description
```

```
        "The source security zone of the packet";
    }
    leaf dst-zone {
        type string;
        description
            "The destination security zone of the packet";
    }
    leaf src-region {
        type string;
        description
            "Source region of the traffic";
    }
    leaf dst-region{
        type string;
        description
            "Destination region of the traffic";
    }
    leaf policy-id {
        type uint8;
        description
            "The ID of the policy being triggered";
    }
    leaf policy-name {
        type string;
        description
            "The name of the policy being triggered";
    }
    leaf src-user{
        type string;
        description
            "User who generates traffic";
    }
    leaf protocol {
        type identityref {
            base protocol-type;
        }
        description
            "Protocol type of traffic";
    }
    leaf app {
        type string;
        description
            "Application type of traffic";
    }
}

notification system-detection-alarm {
    description
```

```
    "This notification is sent, when a system alarm
    is detected.";
  leaf alarm-catagory {
    type identityref {
      base alarm-type;
    }
    description
      "The alarm catagory for
      system-detection-alarm notification";
  }
  uses characteristics;
  uses i2nsf-system-alarm-type-content;
  uses common-monitoring-data;
}
notification system-detection-event {
  description
    "This notification is sent, when a security-sensitive
    authentication action fails.";
  leaf event-catagory {
    type identityref {
      base event-type;
    }
    description
      "The event catagory for system-detection-event";
  }
  uses characteristics;
  uses i2nsf-system-event-type-content;
  uses common-monitoring-data;
}
notification nsf-detection-flood {
  description
    "This notification is sent,
    when a specific flood type is detected";
  leaf event-name {
    type identityref {
      base SEC-EVENT-DDOS;
    }
    description
      "The event name for nsf-detection-flood";
  }
  uses i2nsf-nsf-event-type-content;
  leaf sub-attack-type {
    type identityref {
      base flood-type;
    }
    description
      "Any one of Syn flood, ACK flood, SYN-ACK flood,
      FIN/RST flood, TCP Connection flood, UDP flood,
```



```
        Icmp flood, HTTPS flood, HTTP flood, DNS query flood,
        DNS reply flood, SIP flood, and etc.";
    }
    leaf start-time {
        type yang:date-and-time;
        mandatory true;
        description
            "The time stamp indicating when the attack started";
    }
    leaf end-time {
        type yang:date-and-time;
        mandatory true;
        description
            "The time stamp indicating when the attack ended";
    }
    leaf attack-rate {
        type uint32;
        description
            "The PPS rate of attack traffic";
    }
    leaf attack-speed {
        type uint32;
        description
            "The BPS speed of attack traffic";
    }
    uses common-monitoring-data;
}
notification nsf-detection-session-table {
    description
        "This notification is sent, when an a session table event
        is deteced";
    leaf current-session {
        type uint8;
        description
            "The number of concurrent sessions";
    }
    leaf maximum-session {
        type uint8;
        description
            "The maximum number of sessions that the session
            table can support";
    }
    leaf threshold {
        type uint8;
        description
            "The threshold triggering the event";
    }
    uses common-monitoring-data;
```

```
}
notification nsf-detection-virus {
  description
    "This notification is sent, when a virus is detected";
  uses i2nsf-nsf-event-type-content-extend;
  leaf virus {
    type identityref {
      base virus-type;
    }
    description
      "The virus type for nsf-detection-virus notification";
  }
  leaf virus-name {
    type string;
    description
      "The name of the detected virus";
  }

  leaf file-type {
    type string;
    description
      "The type of file virus code is found in (if applicable).";
  }
  leaf file-name {
    type string;
    description
      "The name of file virus code is found in (if applicable).";
  }
  uses common-monitoring-data;
}
notification nsf-detection-intrusion {
  description
    "This notification is send, when an intrusion event
    is detected.";
  uses i2nsf-nsf-event-type-content-extend;
  leaf protocol {
    type identityref {
      base protocol-type;
    }
    description
      "The protocol type for nsf-detection-intrusion notification";
  }
  leaf app {
    type string;
    description
      "The employed application layer protocol";
  }
  leaf sub-attack-type {
```

```
    type identityref {
      base intrusion-attack-type;
    }
    description
      "The sub attack type for intrusion attack";
  }
  uses common-monitoring-data;
}

notification nsf-detection-botnet {
  description
    "This notification is send, when a botnet event is
    detected";
  uses i2nsf-nsf-event-type-content-extend;
  leaf attack-type {
    type identityref {
      base botnet-attack-type;
    }
    description
      "The attack type for botnet attack";
  }
  leaf protocol {
    type identityref {
      base protocol-type;
    }
    description
      "The protocol type for nsf-detection-botnet notification";
  }
  leaf botnet-name {
    type string;
    description
      "The name of the detected botnet";
  }
  leaf role {
    type string;
    description
      "The role of the communicating
      parties within the botnet";
  }
  uses common-monitoring-data;
}

notification nsf-detection-web-attack {
  description
    "This notification is send, when an attack event is
    detected";
  uses i2nsf-nsf-event-type-content-extend;
  leaf sub-attack-type {
    type identityref {
      base web-attack-type;
    }
  }
}
```

```
    }
    description
      "Concret web attack type, e.g., sql injection,
       command injection, XSS, CSRF";
  }
  leaf request-method {
    type identityref {
      base req-method;
    }
    description
      "The method of requirement. For instance, PUT or
       GET in HTTP";
  }
  leaf req-uri {
    type string;
    description
      "Requested URI";
  }
  leaf uri-category {
    type string;
    description
      "Matched URI category";
  }
  leaf-list filtering-type {
    type identityref {
      base filter-type;
    }
    description
      "URL filtering type, e.g., Blacklist, Whitelist,
       User-Defined, Predefined, Malicious Category,
       Unknown";
  }
  uses common-monitoring-data;
}
notification system-access-log {
  description
    "The notification is send, if there is
     a new system log entry about
     a system access event";
  leaf login-ip {
    type inet:ipv4-address;
    mandatory true;
    description
      "Login IP address of a user";
  }
  leaf administrator {
    type string;
    description
```

```
        "Administrator that maintains the device";
    }
    leaf login-mode {
        type login-mode;
        description
            "Specifies the administrator log-in mode";
    }
    leaf operation-type {
        type operation-type;
        description
            "The operation type that the administrator execute";
    }
    leaf result {
        type string;
        description
            "Command execution result";
    }
    leaf content {
        type string;
        description
            "The Operation performed by an administrator after login";
    }
    uses characteristics;
}
notification system-res-util-log {
    description
        "This notification is send, if there is
        a new log entry representing ressource
        utilization updates.";
    leaf system-status {
        type string;
        description
            "The current systems
            running status";
    }
    leaf cpu-usage {
        type uint8;
        description
            "Specifies the relative amount of
            cpu usage wrt plattform ressource";
    }
    leaf memory-usage {
        type uint8;
        description
            "Specifies the amount of memory usage";
    }
    leaf disk-usage {
        type uint8;
    }
}
```

```
        description
            "Specifies the amount of disk usage";
    }
    leaf disk-left {
        type uint8;
        description
            "Specifies the amount of disk left";
    }
    leaf session-num {
        type uint8;
        description
            "The total number of sessions";
    }
    leaf process-num {
        type uint8;
        description
            "The total number of process";
    }
    leaf in-traffic-rate {
        type uint32;
        description
            "The total inbound traffic rate in pps";
    }
    leaf out-traffic-rate {
        type uint32;
        description
            "The total outbound traffic rate in pps";
    }
    leaf in-traffic-speed {
        type uint32;
        description
            "The total inbound traffic speed in bps";
    }
    leaf out-traffic-speed {
        type uint32;
        description
            "The total outbound traffic speed in bps";
    }
    uses characteristics;
}
notification system-user-activity-log {
    description
        "This notification is send, if there is
        a new user activity log entry";
    uses characteristics;
    uses i2nsf-system-event-type-content;
    leaf access {
        type identityref {
```

```
        base access-mode;
    }
    description
        "The access type for system-user-activity-log notification";
}
leaf online-duration {
    type string;
    description
        "Online duration";
}
leaf logout-duration {
    type string;
    description
        "Lockout duration";
}
leaf additional-info {
    type string;
    description
        "User activities. e.g., Successful
        User Login, Failed Login attempts,
        User Logout, Successful User
        Password Change, Failed User
        Password Change, User Lockout,
        User Unlocking, Unknown";
}
}
notification nsf-log-ddos {
    description
        "This notification is send, if there is
        a new DDoS event log entry in the nsf log";
    leaf attack-type {
        type identityref {
            base ddos-attack-type;
        }
        description
            "The ddos attack type for
            nsf-log-ddos notification";
    }
    leaf attack-ave-rate {
        type uint32;
        description
            "The ave PPS of attack traffic";
    }
    leaf attack-ave-speed {
        type uint32;
        description
            "the ave bps of attack traffic";
    }
}
```

```
    leaf attack-pkt-num {
        type uint32;
        description
            "the number of attack packets";
    }
    leaf attack-src-ip {
        type inet:ipv4-address;
        description
            "The source IP addresses of attack
            traffics. If there are a large
            amount of IP addresses, then
            pick a certain number of resources
            according to different rules.";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
    }
    uses characteristics;
    uses common-monitoring-data;
}
notification nsf-log-virus {
    description
        "This notification is send, If there is
        a new virus event log enry in the nsf log";
    leaf attack-type {
        type identityref {
            base virus-type;
        }
        description
            "The virus type for nsf-log-virus notification";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
    }
    leaf os{
        type string;
        description
            "simple os information";
    }
    leaf time {
```



```
    type yang:date-and-time;
    mandatory true;
    description
        "Indicate the time when the message is generated";
}
uses characteristics;
uses common-monitoring-data;
}
notification nsf-log-intrusion {
    description
        "This notification is send, if there is
        a new intrusion event log entry in the nsf log";
    leaf attack-type {
        type identityref {
            base intrusion-attack-type;
        }
        description
            "The intrusion attack type for
            nsf-log-intrusion notification";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
    }
    leaf time {
        type yang:date-and-time;
        mandatory true;
        description
            "Indicate the time when the message is generated";
    }
    leaf attack-rate {
        type uint32;
        description
            "The PPS of attack traffic";
    }
    leaf attack-speed {
        type uint32;
        description
            "The bps of attack traffic";
    }
    uses characteristics;
    uses common-monitoring-data;
}
notification nsf-log-botnet {
    description
```

```
    "This notification is send, if there is
    a new botnet event log in the nsf log";
  leaf attack-type {
    type identityref {
      base botnet-attack-type;
    }
    description
      "The botnet attack type for
      nsf-log-botnet notification";
  }
  leaf action {
    type log-action;
    description
      "Action type: allow, alert,
      block, discard, declare,
      block-ip, block-service";
  }
  leaf botnet-pkt-num{
    type uint8;
    description
      "The number of the packets sent to
      or from the detected botnet";
  }
  leaf os{
    type string;
    description
      "simple os information";
  }
  uses characteristics;
  uses common-monitoring-data;
}

notification nsf-log-dpi {
  description
    "This notification is send, if there is
    a new dpi event in the nsf log";
  leaf attack-type {
    type dpi-type;
    description
      "The type of the dpi";
  }
  uses characteristics;
  uses i2nsf-nsf-counters-type-content;
  uses common-monitoring-data;
}

notification nsf-log-vuln-scan {
  description
    "This notification is send, if there is
    a new vulnerability-scan report in the nsf log";
```

```
leaf vulnerability-id {
    type uint8;
    description
        "The vulnerability id";
}
leaf victim-ip {
    type inet:ipv4-address;
    description
        "IP address of the victim host which has vulnerabilities";
}
leaf protocol {
    type identityref {
        base protocol-type;
    }
    description
        "The protocol type for
        nsf-log-vuln-scan notification";
}
leaf port-num {
    type inet:port-number;
    description
        "The port number";
}
leaf level {
    type severity;
    description
        "The vulnerability severity";
}
leaf os {
    type string;
    description
        "simple os information";
}
leaf vulnerability-info {
    type string;
    description
        "The information about the vulnerability";
}
leaf fix-suggestion {
    type string;
    description
        "The fix suggestion to the vulnerability";
}
leaf service {
    type string;
    description
        "The service which has vulnerabillity in the victim host";
}
```

```
    uses characteristics;
    uses common-monitoring-data;
}
notification nsf-log-web-attack {
  description
    "This notificatio is send, if there is
    a new web-attack event in the nsf log";
  leaf attack-type {
    type identityref {
      base web-attack-type;
    }
    description
      "The web attack type for
      nsf-log-web-attack notification";
  }
  leaf rsp-code {
    type string;
    description
      "Response code";
  }
  leaf req-clientapp {
    type string;
    description
      "The client application";
  }
  leaf req-cookies {
    type string;
    description
      "Cookies";
  }
  leaf req-host {
    type string;
    description
      "The domain name of the requested host";
  }
  leaf raw-info {
    type string;
    description
      "The information describing
      the packet triggering the event.";
  }
  uses characteristics;
  uses common-monitoring-data;
}
container counters {
  description
    "This is probably better covered by an import
    as this will not be notifications.
```

```
Counter are not very suitable as telemetry, maybe
via periodic subscriptions, which would still
violate principle of least surprise.";
container system-interface {
  description
    "The system counter type is interface counter";
  uses characteristics;
  uses i2nsf-system-counter-type-content;
  uses common-monitoring-data;
}
container nsf-firewall {
  description
    "The nsf counter type is firewall counter";
  uses characteristics;
  uses i2nsf-nsf-counters-type-content;
  uses traffic-rates;
}
container nsf-policy-hits {
  description
    "The counters of policy hit";
  uses characteristics;
  uses i2nsf-nsf-counters-type-content;
  uses common-monitoring-data;
  leaf hit-times {
    type uint32;
    description
      "The hit times for policy";
  }
}
}
}
}
}
<CODE ENDS>
```

Figure 2: Data Model of Monitoring

6. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This document has greatly benefited from inputs by Daeyoung Hyun.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

7.2. Informative References

- [i2nsf-framework]
Lopez,, D., Lopez,, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.
- [i2nsf-monitoring-im]
Xia,, L., Zhang,, D., Wu, Y., Kumar, R., Lohiya, A., and H. Birkholz, "An Information Model for the Monitoring of Network Security Functions (NSF)", draft-zhang-i2nsf-info-model-monitoring-06 (work in progress), May 2018.
- [i2nsf-terminology]
Hares,, S., Strassner,, J., Lopez,, D., Xia,, L., and H. Birkholz,, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), July 2018.
- [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.

Appendix A. Changes from draft-hong-i2nsf-nsf-monitoring-data-model-03

The following changes are made from draft-hong-i2nsf-nsf-monitoring-data-model-03:

1. The YANG data model has been reorganized in detail by synchronizing with the latest info model.
2. The YANG data model has been reorganized by a partial implementation based on ConfD.

Authors' Addresses

Dongjin Hong
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 7630 5473
EMail: dong.jin@skku.edu

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

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

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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu
China

EMail: Frank.xialiang@huawei.com

Henk Birkholz
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

EMail: henk.birkholz@sit.fraunhofer.de

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

S. Hyun
Chosun University
J. Jeong
T. Roh
S. Wi
Sungkyunkwan University
J. Park
ETRI
July 2, 2018

I2NSF Registration Interface YANG Data Model
draft-hyun-i2nsf-registration-interface-dm-04

Abstract

This document describes an YANG data model for I2NSF registration interface between Security Controller and Developer's Management System. The data model is required for NSF instance registration and dynamic life cycle management of NSF instances.

Status of This Memo

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	2
3. Terminology	3
3.1. Tree Diagrams	3
4. High-Level YANG	4
4.1. Registration Interface	4
4.2. Registration Request	4
4.3. Instance Management Request	5
4.4. NSF Capability Information	5
4.5. NSF Access Information	6
4.6. NSF Performance Capability	6
4.7. Role-Based ACL(Access Control List)	6
5. YANG Modules	7
5.1. XML Example of Registration Interface Data Model	12
6. Security Considerations	13
7. Acknowledgments	13
8. References	13
8.1. Normative References	13
8.2. Informative References	14
Appendix A. Changes from draft-hyun-i2nsf-registration- interface-dm-03	16
Authors' Addresses	16

1. Introduction

This document provides a YANG [RFC6020] data model that defines the required data for the registration interface between Security Controller and Developer's Management System to dynamically manage a pool of NSF instances. This document defines a YANG data model based on the [i2nsf-reg-inf-im]. The terms used in this document are defined in [i2nsf-terminology].

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-terminology], [capability-im], [RFC8329], [nsf-triggered-steering], [supa-policy-data-model], and [supa-policy-info-model].

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy. [nsf-triggered-steering]
- o Advanced Inspection/Action: As like the I2NSF information model for NSF facing interface [capability-im], Advanced Inspection/Action means that a security function calls another security function for further inspection based on its own inspection result. [nsf-triggered-steering]
- o Network Security Function Profile (NSF Capability Information): NSF Capability Information specifies the inspection capabilities of the associated NSF instance. Each NSF instance has its own NSF Capability Information to specify the type of security service it provides and its resource capacity etc. [nsf-triggered-steering]
- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol. [supa-policy-info-model]
- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol. [supa-policy-info-model]

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] is as follows:

Brackets "[" and "]" enclose list keys.

Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

Ellipsis ("...") stands for contents of subtrees that are not shown.

4. High-Level YANG

This section provides an overview of the high level YANG.

4.1. Registration Interface

```
module : ietf-i2nsf-regs-interface-model
  +--rw regs-req
  |   uses i2nsf-regs-req
  +--rw instance-mgmt-req
  |   uses i2nsf-instance-mgmt-req
```

Figure 1: High-Level YANG of I2NSF Registration Interface

Each of these sections mirror sections of [i2nsf-reg-inf-im].

4.2. Registration Request

This section expands the i2nsf-regs-req in Figure 1.

```
Registration Request
  +--rw i2nsf-regs-req
  |   +--rw nsf-capability-information
  |   |   uses i2nsf-nsf-capability-information
  |   +--rw nsf-access-info
  |   |   uses i2nsf-nsf-access-info
```

Figure 2: High-Level YANG of I2NSF Registration Request

Registration Request contains the capability information of newly created NSF to notify its capability to Security Controller. The request also contains Network Access Information so that the Security Controller can access the NSF.

4.3. Instance Management Request

This section expands the `i2nsf-instance-mgmt-req` in Figure 1.

Instance Management Request

```

+--rw i2nsf-instance-mgmt-req
  +--rw req-level uint16
  +--rw req-id uint64
  +--rw (req-type)?
    +--rw (instanciation-request)
      +--rw in-nsf-capability-information
      |   uses i2nsf-nsf-capability-information
    +--rw (deinstanciation-request)
      +--rw de-nsf-access-info
      |   uses i2nsf-nsf-access-info
    +--rw (reinstanciation-request)
      +--rw re-nsf-capability-information
      |   uses i2nsf-nsf-capability-information
```

Figure 3: High-Level YANG of I2NSF Instance Mgmt Request

Instance management request consists of two types: `instanciation-request`, `deinstanciation-request`, and `reinstanciation-request`. The `instanciation-request` is used to request generation of a new NSF instance with NSF Capability Information which specifies required NSF capability information. The `deinstanciation-request` is used to remove an existing NSF with NSF Access Information. The `reinstanciation nsf request` is used to updating a existing NSF information with NSF capabilities.

4.4. NSF Capability Information

This section expands the `i2nsf-nsf-capability-information` in Figure 2 and Figure 3.

NSF Capability Information

```

+--rw i2nsf-nsf-capability-information
  +--rw i2nsf-capability
  |   uses ietf-i2nsf-capability
  +--rw performance-capability
  |   uses i2nsf-nsf-performance-caps
```

Figure 4: High-Level YANG of I2NSF NSF Capability Information

In Figure 4, `ietf-i2nsf-capability` refers module `ietf-i2nsf-capability` in `[i2nsf-capability-dm]`. We add the performance

capability because it is absent in [i2nsf-capability-dm] and [netmod-acl-model]

4.5. NSF Access Information

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

NSF Access Information

```
+--rw i2nsf-nsf-access-info
  +--rw nsf-address   inet:ipv4-address
  +--rw nsf-port-address inet:port-number
```

Figure 5: High-Level YANG of I2NSF NSF Access Information

This information is used by other components to access an NSF.

4.6. NSF Performance Capability

This section expands the i2nsf-nsf-performance-caps in Figure 4.

NSF Performance Capability

```
+--rw i2nsf-nsf-performance-caps
  +--rw processing
  |   +--rw processing-average uint16
  |   +--rw processing-peak uint16
  +--rw bandwidth
  |   +--rw outbound
  |   |   +--rw outbound-average uint16
  |   |   +--rw outbound-peak uint16
  |   +--rw inbound
  |   |   +--rw inbound-average uint16
  |   |   +--rw inbound-peak uint16
```

Figure 6: High-Level YANG of I2NSF NSF Performance Capability

When the Security Controller requests the Developer Management System to create a new NSF instance, the performance capability is used to specify the performance requirements of the new instance.

4.7. Role-Based ACL(Access Control List)

This section expands the ietf-netmod-acl-model in [netmod-acl-model].

```
Role-Based ACL
+--rw role-based-acl
    uses ietf-netmod-acl-model
```

Figure 7: Role-Based ACL

In [netmod-acl-model], ietf-netmod-acl-model refers module ietf-netmod-acl-model in [netmod-acl-model]. We add the role-based ACL because it is absent in [i2nsf-capability-dm].

5. YANG Modules

This section introduces a YANG module for the information model of the required data for the registration interface between Security Controller and Developer's Management System, as defined in the [i2nsf-reg-inf-im].

```
<CODE BEGINS> file "ietf-i2nsf-regs-interface@2018-07-02.yang"
    module ietf-i2nsf-regs-interface {
        namespace
            "urn:ietf:params:xml:ns:yang:ietf-i2nsf-regs-interface";
        prefix
            regs-interface;
        import ietf-inet-types{
            prefix inet;
        }

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

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

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

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

            Editor: Sangwon Hyun
            <mailto:swhyun77@skku.edu>

            Editor: Taekyun Roh
            <mailto:tkroh0198@skku.edu>

            Editor: Sarang Wi
```

<mailto:dnl9795@skku.edu>

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

Editor: Jung-Soo Park
<mailto:pjs@etri.re.kr>;

```
description
  "It defines a YANG data module for Registration Interface.";
revision "2018-07-02" {
  description "The second revision";
  reference
    "draft-hares-i2nsf-capability-data-model-07
    draft-hyun-i2nsf-registration-interface-im-05";
}
list interface-container {
  key "interface-name";
  description
    "i2nsf-reg-interface-container";
  leaf interface-name {
    type string;
    description
      "interface name";
  }
  container i2nsf-regs-req {
    description
      "The capability information of newly
      created NSF to notify its
      capability to Security Controller";
    container nsf-capability-information {
      description
        "nsf-capability-information";
      uses i2nsf-nsf-capability-information;
    }
    container nsf-access-info {
      description
        "nsf-access-info";
      uses i2nsf-nsf-access-info;
    }
    container ietf-netmod-acl-model {
      description
        "netmod-acl-model";
      uses ietf-netmod-acl-model;
    }
  }
  container i2nsf-instance-mgmt-req {
    description
```



```
    "Required information for instantiation-request,
    deinstanciacion-request and reinstanciacion-request";
    leaf req-level {
        type uint16;
        description
            "req-level";
    }
    leaf req-id {
        type uint64;
        mandatory true;
        description
            "req-id";
    }
    choice req-type {
        description
            "req-type";
        case instantiation-request {
            description
                "instantiation-request";
            container in-nsf-capability-information {
                description
                    "nsf-capability-information";
                uses i2nsf-nsf-capability-information;
            }
        }
        case deinstanciacion-request {
            description
                "deinstanciacion-request";
            container de-nsf-access-info {
                description
                    "nsf-access-info";
                uses i2nsf-nsf-access-info;
            }
        }
        case reinstanciacion-request {
            description
                "reinstanciacion nsf's information";
            container re-nsf-capability-information {
                description
                    "nsf-capability-information";
                uses i2nsf-nsf-capability-information;
            }
        }
    }
}
}
}
}
grouping i2nsf-nsf-performance-caps {
    description
```

```
    "NSF performance capabilities";
    container processing{
        description
            "processing info";
        leaf processing-average{
            type uint16;
            description
                "processing-average";
        }
        leaf processing-peak{
            type uint16;
            description
                "processing peak";
        }
    }
    container bandwidth{
        description
            "bandwidth info";
        container inbound{
            description
                "inbound";
            leaf inbound-average{
                type uint16;
                description
                    "inbound-average";
            }
            leaf inbound-peak{
                type uint16;
                description
                    "inbound-peak";
            }
        }
        container outbound{
            description
                "outbound";
            leaf outbound-average{
                type uint16;
                description
                    "outbound-average";
            }
            leaf outbound-peak{
                type uint16;
                description
                    "outbound-peak";
            }
        }
    }
}
```

```
    grouping i2nsf-nsf-capability-information {
      description
        "Detail information of an NSF";
      container performance-capability {
        uses i2nsf-nsf-performance-caps;
        description
          "performance-capability";
      }
      container i2nsf-capability {
        description
          "It refers draft-hares-i2nsf-capability-data-model-07.txt
          later";
      }
    }
    grouping ietf-netmod-acl-model {
      description
        "Detail information";
      container role-based-acl {
        description
          "It refers draft-ietf-netmod-acl-model-15.txt
          later";
      }
    }
    grouping i2nsf-nsf-access-info {
      description
        "NSF access information";
      leaf nsf-address {
        type inet:ipv4-address;
        mandatory true;
        description
          "nsf-address";
      }
      leaf nsf-port-address {
        type inet:port-number;
        description
          "nsf-port-address";
      }
    }
  }
}
```

<CODE ENDS>

Figure 8: Data Model of I2NSF Registration Interface

5.1. XML Example of Registration Interface Data Model

Requirement: Registering the IDS NSF with VoIP/VoLTE security capability using Registration interface.

Here is the configuration xml for this Registration Interface:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <i2nsf-regs-req>
        <i2nsf-nsf-capability-information>
          <ietf-i2nsf-capability>
            <nsf-capabilities>
              <nsf-capabilities-id>1</nsf-capabilities-id>
              <con-sec-control-capabilities>
                <content-security-control>
                  <ids>
                    <ids-support>true</ids-support>
                    <ids-fcn nc:operation="create">
                      <ids-fcn-name>ids-service</ids-fcn-name>
                    </ids-fcn>
                  </ids>
                  <voip-volte>
                    <voip-volte-support>true</voip-volte-support>
                    <voip-volte-fcn nc:operation="create">
                      <voip-volte-fcn-name>
                        ips-service
                      </voip-volte-fcn-name>
                    </voip-volte-fcn>
                  </voip-volte>
                </content-security-control>
              </con-sec-control-capabilities>
            </nsf-capabilities>
          </ietf-i2nsf-capability>
          <i2nsf-nsf-performance-caps>
            <processing>
              <processing-average>1000</processing-average>
              <processing-peak>5000</processing-peak>
            </processing>
            <bandwidth>
              <outbound>
                <outbound-average>1000</outbound-average>
                <outbound-peak>5000</outbound-peak>
              </outbound>
            </bandwidth>
          </i2nsf-nsf-performance-caps>
        </i2nsf-nsf-capability-information>
      </i2nsf-regs-req>
    </config>
  </edit-config>
</rpc>
```

```
        </outbound>
        <inbound>
          <inbound-average>1000</inbound-average>
          <inbound-peak>5000</inbound-peak>
        </inbound>
      </bandwidth>
    </i2nsf-nsf-performance-caps>
  </i2nsf-nsf-capability-information>
  <nsf-access-info>
    <nsf-address>10.0.0.1</nsf-address>
    <nsf-port-address>145</nsf-port-address>
  </nsf-access-info>
</i2nsf-regs-req>
</config>
</edit-config>
</rpc>
```

Figure 9: Registration Interface example

6. Security Considerations

The information model of the registration interface is based on the I2NSF framework without any architectural changes. Thus, this document shares the security considerations of the I2NSF framework architecture that are specified in [RFC8329] for the purpose of achieving secure communication among components in the proposed architecture.

7. 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).

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs toIndicate Requirement Levels", RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

8.2. Informative References

- [capability-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez,
"Information Model of NSFs Capabilities", draft-i2nsf-
capability-00 (work in progress), September 2017.
- [i2nsf-capability-dm]
Hares, S., Jeong, J., Kim, J., Moskowitz, R., and Q. Lin,
"I2NSF Capability YANG Data Model", draft-hares-i2nsf-
capability-data-model-07 (work in progress), March 2018.
- [i2nsf-reg-inf-im]
Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF
Registration Interface Information Model", draft-hyun-
i2nsf-registration-interface-im-04 (work in progress),
October 2017.
- [i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H.
Birkholz, "Interface to Network Security Functions (I2NSF)
Terminology", draft-ietf-i2nsf-terminology-05 (work in
progress), January 2018.
- [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.
- [netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair,
"Network Access Control List (ACL) YANG Data Model",
draft-ietf-netmod-acl-model-16 (work in progress),
February 2018.
- [nsf-triggered-steering]
Hyun, S., Jeong, J., Park, J., and S. Hares, "Service
Function Chaining-Enabled I2NSF Architecture", draft-hyun-
i2nsf-nsf-triggered-steering-05 (work in progress), March
2018.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R.
Kumar, "Framework for Interface to Network Security
Functions", RFC 8329, February 2018.

`[supa-policy-data-model]`

Halpern, J., Strassner, J., and S. van der Meer, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-data-model-04 (work in progress), June 2017.

`[supa-policy-info-model]`

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

Appendix A. Changes from draft-hyun-i2nsf-registration-interface-dm-03

The following changes are made from draft-hyun-i2nsf-registration-interface-dm-03:

- o We added Re-instantiation item.
- o The references were updated to reflect the latest documents.

Authors' Addresses

Sangwon Hyun
Department of Computer Engineering
Chosun University
309, Pilmun-daero, Dong-gu
Gwangju, Jeollanam-do 61452
Republic of Korea

EMail: shyun@chosun.ac.kr

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

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

Taekyun Roh
Electrical Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: tkroh0198@skku.edu
URI: http://imtl.skku.ac.kr/xs/index.php?mid=board_YoKq57

Sarang Wi
Electrical Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: dnl9795@skku.edu
URI: http://imtl.skku.ac.kr/xs/index.php?mid=board_YoKq57

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

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

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

S. Hyun
Chosun University
J. Jeong
T. Roh
S. Wi
Sungkyunkwan University
J. Park
ETRI
July 2, 2018

Registration Interface Information Model
draft-hyun-i2nsf-registration-interface-im-05

Abstract

This document describes an information model for Interface to Network Security Functions (I2NSF) Registration Interface between Security Controller and Developer's Management System (DMS). The information model is required to support NSF instance via the registration interface. This document explains the procedures over I2NSF registration interface for these functionalities. It also describes the detailed information which should be exchanged via I2NSF registration interface.

Status of This Memo

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
4. Objectives	3
5. Information Model	4
5.1. NSF Instance Managment Mechanism	5
5.2. NSF Registration Mechanism	6
5.3. NSF Access Information	6
5.4. NSF Capability Information (Capabilities of an NSF instance)	7
5.4.1. Performance Capabilities	7
5.5. Role-based Access Control List	8
6. Security Considerations	9
7. Acknowledgments	9
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Appendix A. Lifecycle Managment Mechanism	12
Appendix B. Changes from draft-hyun-i2nsf-registration-interface-im-04	12
Authors' Addresses	12

1. Introduction

A number of virtual network security function instances typically exist in Interface to Network Security Functions (I2NSF) framework [RFC8329]. Since these NSF instances may have different security capabilities, it is important to register the security capabilities of each NSF instance into the security controller after they have been created. In addition, it is required to instantiate NSFs of some required security capabilities on demand. As an example, if additional security capabilities are required to meet the new security requirements that an I2NSF user requests, the security controller should be able to request the DMS to instantiate NSFs that have the required security capabilities.

This document describes the information model which is required for the registration interface between security controller and developer's management system to support registration and instantiation of NSFs. It further describes the procedure based on the information model which should be performed by the security controller and the developer's management system via the registration interface.

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-terminology][capability-im][RFC8329] [nsf-triggered-steering].

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy [nsf-triggered-steering].
- o Advanced Inspection/Action: As like the I2NSF information model for NSF-facing interface [capability-im], Advanced Inspection/Action means that a security function calls another security function for further inspection based on its own inspection result [nsf-triggered-steering].
- o Network Security Function Profile (NSF Profile): NSF Profile specifies the security and performance capability of an NSF instance. Each NSF instance has its own NSF Profile which describes the type of security service it can provide and its performance capability. [nsf-triggered-steering].

4. Objectives

- o Registering NSF instances from Developer's Management System: Depending on system's security requirements, it may require some NSFs by default. In this case, DMS creates these default NSF instances without the need of receiving requests from Security

Controller. After creating them, DMS notifies Security Controller of those NSF instances via registration interface.

- o Creating an NSF instance to serve another NSF's inspection request: In I2NSF framework, an NSF can trigger another type of NSF(s) for more advanced security inspection of the traffic. In this case, the next NSF is determined by the current NSF's inspection result and client's policy. However, if there is no available NSF instance to serve the former NSF's request, we should create an NSF instance by requesting Developer's Management System (DMS) through registration interface.
- o Creating NSF instances required to enforce security policy rules from Client: In I2NSF framework, users decide which security service is necessary in the system. If there is no NSF instances to enforce the client's security policy, then we should also create the required instances by requesting DMS via registration interface.
- o Deleting unnecessary NSF instances: In I2NSF framework, users decide which security service is unnecessary in the system. If there is unused NSF instances to enforce the client's security policy, then we should also delete the existing instances by requesting DMS via registration interface.
- o Updating an NSF instances: After NSF instance is registered in I2NSF framework, capability of NSF instance can occur added and changed. This situation should be recognized by the security controller of the I2NSF framework. Therefore, if there is updated NSF instances, DMS notifies Security Controller of those NSF instances via registration interface.

5. Information Model

The I2NSF registration interface was only used for registering new NSF instances to Security Controller. In this document, however, we extend its utilization to support on demand NSF instantiation/de-instantiation and describe the information that should be exchanged via the registration interface for the functionality. Moreover, we also define the information model of NSF Profile because, for registration interface, NSF Profile (i.e., capabilities of an NSF) needs to be clarified so that the components of I2NSF framework can exchange the set of capabilities in a standardized manner. This is typically done through the following process:

- 1) Security Controller first recognizes the set of capabilities (i.e., NSF Profile) or the signature of a specific NSF required or wasted in the current system.

- 2) Developer's Management System (DMS) matches the recognized information to an NSF based on the information model definition.
- 3) Developer's Management System creates or eliminates NSFs matching with the above information.
- 4) Security Controller can then add/remove the corresponding NSF instance to/from its list of available NSF instances in the system.

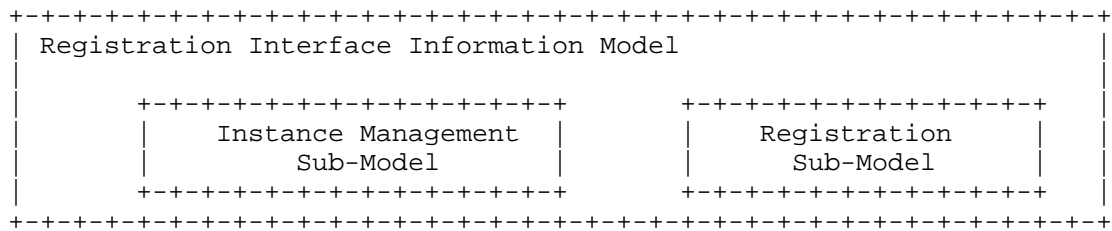


Figure 1: Registration Interface Information Model

As illustrated in Figure 1, the information model for Registration Interface consists of two sub-models: instance management, registration sub-models. The instance management functionality and the registration functionality use NSF Profile to achieve their goals. In this context, NSF Profile is the capability objects that describe and/or prescribe inspection capability an NSF instance can provide.

5.1. NSF Instance Management Mechanism

For the instance management of NSFs, Security Controller in I2NSF framework requires two types of requests: Instantiation Request and Deinstantiation Request. Security Controller sends the request messages to DMS when required. Once receiving the request, DMS conducts creating/eliminating the corresponding NSF instance and responds Security Controller with the results.

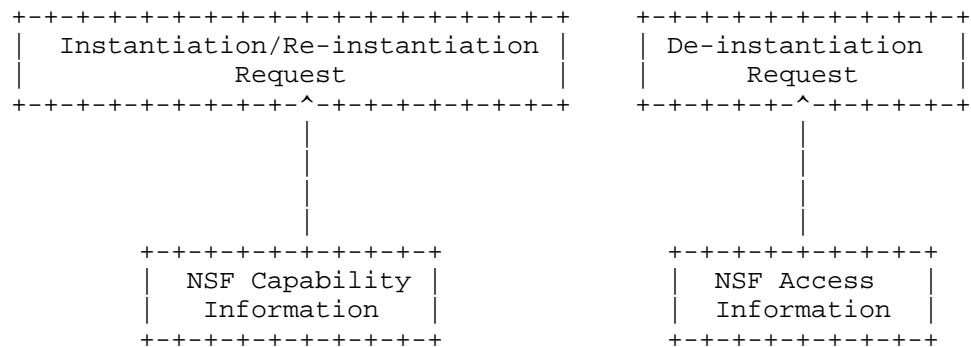


Figure 2: Overview of Instance Management Sub-Model

5.2. NSF Registration Mechanism

In order to register a new NSF instance, DMS should generate a Registration Message to Security Controller. A Registration Message consists of an NSF Profile and an NSF Access Information. The former describes the inspection capability of the new NSF instance and the latter is for enabling network access to the new instance from other components. After this registration process, as explained in [capability-im], the I2NSF capability interface can conduct controlling and monitoring the new registered NSF instance.

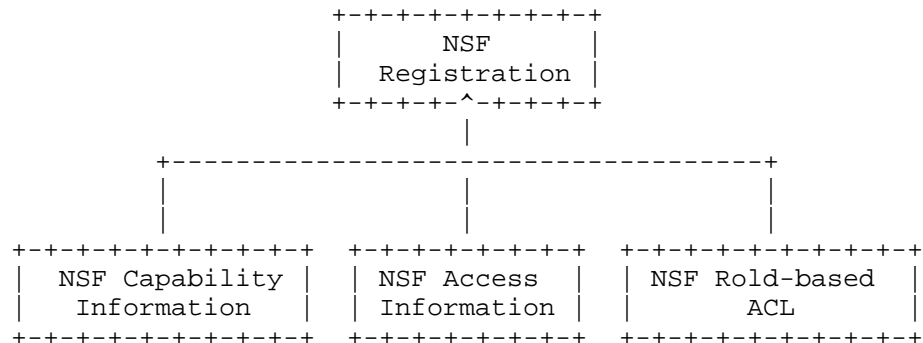


Figure 3: Registration Mechanism Sub-Model Overview

5.3. NSF Access Information

NSF Access Information contains the followings that are required to communicate with an NSF: IPv4 address, IPv6 address, port number, and supported transport protocol(s) (e.g., Virtual Extensible LAN (VXLAN) [RFC 7348], Generic Protocol Extension for VXLAN (VXLAN-GPE)

[draft-ietf-nvo3-vxlan-gpe], Generic Route Encapsulation (GRE), Ethernet etc.). In this document, NSF Access Information is used to identify a specific NSF instance (i.e. NSF Access Information is the signature(unique identifier) of an NSF instance in the overall system).

5.4. NSF Capability Information (Capabilities of an NSF instance)

NSF Profile basically describes the inspection capabilities of an NSF instance. In Figure 4, we show capability objects of an NSF instance. Following the information model of NSF capabilities defined in [capability-im], we share the same security capabilities: Network-Security Capabilities, Content-Security Capabilities, and Attack Mitigation Capabilities. Also, NSF Profile additionally contains the performance capabilities and role-Based access control list (ACL) as shown in Figure 4.

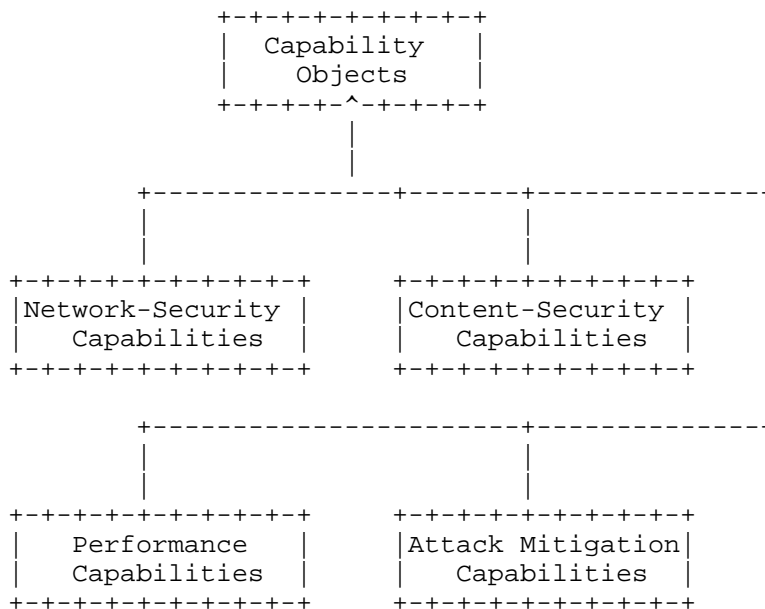


Figure 4: NSF Profile Overview

5.4.1. Performance Capabilities

This information represents the processing capability of an NSF. This information can be used to determine whether the NSF is in congestion by comparing this with the workload that the NSF currently undergoes. Moreover, this information can specify an available amount of each type of resources such as processing power which are

available on the NSF. (The registration interface can control the usages and limitations of the created instance and make the appropriate request according to the status.) As illustrated in Figure 5, this information consists of two items: Processing and Bandwidth. Processing information describes the NSF's available processing power. Bandwidth describes the information about available network amount in two cases, outbound, inbound. This two information can be used for the NSF's instance request.

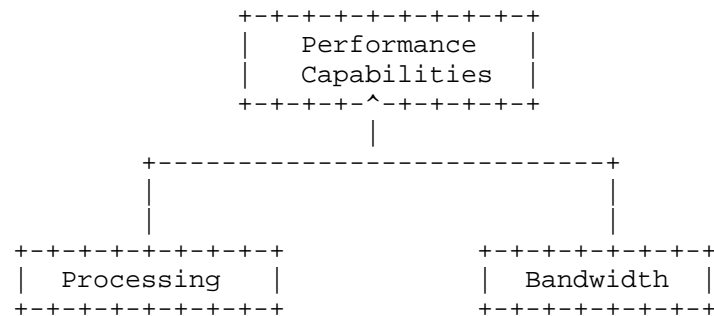


Figure 5: Performance Capability Overview

5.5. Role-based Access Control List

This information specifies access policies of an NSF to determine whether to permit or deny the access of an entity to the NSF based on the role given to the entity. Each NSF is associated with a role-based access control list (ACL) so that it can determine whether to permit or deny the access request from an entity. Figure 6 and Figure 7 show the structure of the role-based ACL, which is composed of role-id, access-type, and permit/deny. The role-id identifies roles of entities (e.g., administrator, developer etc.). The access-type identifies the specific type of access requests such as NSF rule configuration/update and NSF monitoring. Consequently, the role-based ACL in Figure 6 and Figure 7 specifies a set of access-types to be permitted and to be denied by each role-id.

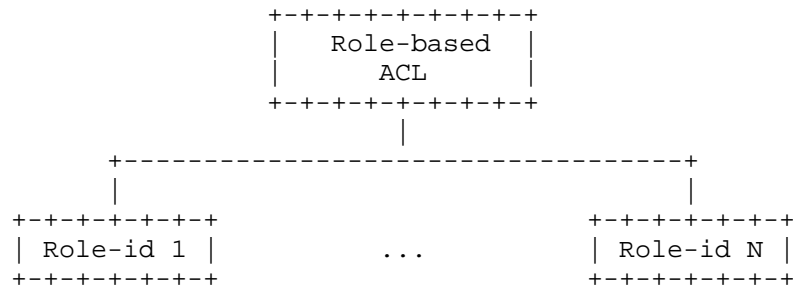


Figure 6: Role-based Access Control List

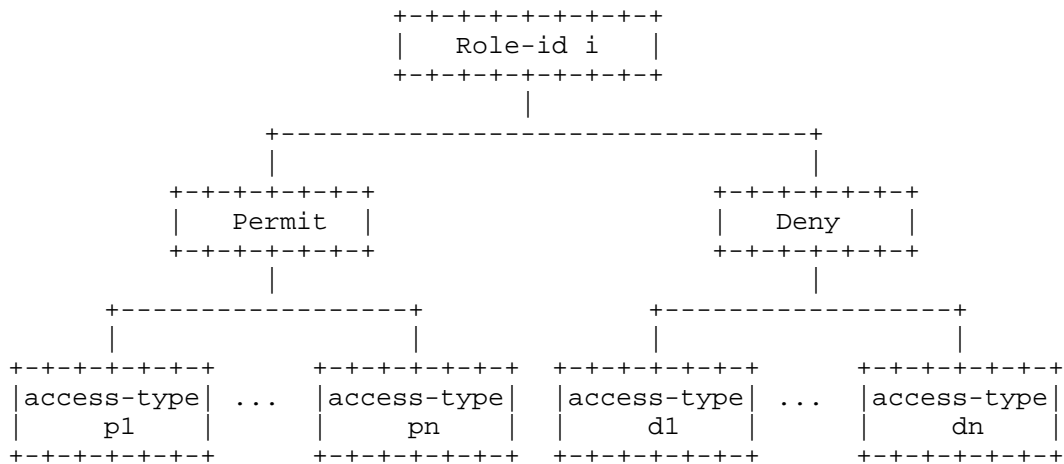


Figure 7: Role-id Subtree

6. Security Considerations

The information model of the registration interface is based on the I2NSF framework without any architectural changes. Thus, this document shares the security considerations of the I2NSF framework that are specified in [RFC8329] for the purpose of achieving secure communication between components in the proposed architecture.

7. 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).

This document has greatly benefited from inputs by SangUk Woo and YunSuk Yeo.

8. References

8.1. Normative References

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

8.2. Informative References

[capability-im]

Xia, L., Strassner, J., Basile, C., and D. Lopez,
"Information Model of NSFs Capabilities", draft-ietf-
i2nsf-capability-01 (work in progress), April 2018.

[draft-ietf-nvo3-vxlan-gpe]

Maino, Ed., F., Kreeger, Ed., L., and U. Elzur, Ed.,
"Generic Protocol Extension for VXLAN", draft-ietf-nvo3-
vxlan-gpe-06 (work in progress), April 2018.

[i2nsf-nsf-monitoring]

Hong, D., Jeong, J., Kim, J., Hares, S., Xia, L., and H.
Birkholz, "YANG Data Model for Monitoring I2NSF Network
Security Functions", draft-hong-i2nsf-nsf-monitoring-data-
model-03 (work in progress), March 2018.

[i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H.
Birkholz, "Interface to Network Security Functions (I2NSF)
Terminology", draft-ietf-i2nsf-terminology-05 (work in
progress), January 2018.

[nfv-architecture]

Yang, Hyunsik. and Younghan. Kim, "I2NSF on the NFV
Reference Architecture", draft-yang-i2nsf-nfv-
architecture-01 (work in progress), March 2018.

[nfv-framework]

"Network Functions Virtualisation (NFV); Architectural
Framework", ETSI GS NFV 002 ETSI GS NFV 002 V1.1.1,
October 2013.

[nsf-triggered-steering]

Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-05 (work in progress), March 2018.

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

Appendix A. Lifecycle Managemenet Mechanism

NFV can be used to virtualize NSFs in I2NSF systems, and DMSs likely perform life-cycle management of NSF instances via Ve-Vnfm interface [nfv-framework] in this environment. The security controller periodically gathers information of NSF instances via the monitoring interface [i2nsf-nsf-monitoring]. This information can be additionally used for life-cycle management of NSF instances, and so the security controller can deliver the information to the DMSs via the registration interface.

Appendix B. Changes from draft-hyun-i2nsf-registration-interface-im-04

The following changes are made from draft-hyun-i2nsf-registration-interface-im-04:

- o Section 4 has been revised to discuss about updating an modified NSF instance via registration interface.
- o Figures 2, 3 and 4 have been updated.
- o Appendix A has been added to discuss about the use of the registration interface related to lifecycle management.
- o The references have been updated to reflect the latest documents.

Authors' Addresses

Sangwon Hyun
Department of Computer Engineering
Chosun University
309, Pilmun-daero, Dong-gu
Gwangju, Jeollanam-do 61452
Republic of Korea

EMail: shyun@chosun.ac.kr

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

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

TaeKyun Roh
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: tkroh0198@skku.edu
URI: http://imtl.skku.ac.kr/xs/index.php?mid=board_YoKq57

SaRang Wi
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: dnl9795@skku.edu
URI: http://imtl.skku.ac.kr/xs/index.php?mid=board_YoKq57

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

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

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: March 18, 2020

J. Jeong
Sungkyunkwan University
S. Hyun
Myongji University
T. Ahn
Korea Telecom
S. Hares
Huawei
D. Lopez
Telefonica I+D
September 15, 2019

Applicability of Interfaces to Network Security Functions to Network-
Based Security Services
draft-ietf-i2nsf-applicability-18

Abstract

This document describes the applicability of Interface to Network Security Functions (I2NSF) to network-based security services in Network Functions Virtualization (NFV) environments, such as firewall, deep packet inspection, or attack mitigation engines.

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 March 18, 2020.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. I2NSF Framework	5
4. Time-dependent Web Access Control Service	8
5. Intent-based Security Services	13
6. I2NSF Framework with SFC	15
7. I2NSF Framework with SDN	17
7.1. Firewall: Centralized Firewall System	19
7.2. Deep Packet Inspection: Centralized VoIP/VoLTE Security System	20
7.3. Attack Mitigation: Centralized DDoS-attack Mitigation System	20
8. I2NSF Framework with NFV	21
9. Security Considerations	23
10. Acknowledgments	24
11. Contributors	24
12. References	24
12.1. Normative References	24
12.2. Informative References	26
Appendix A. Changes from draft-ietf-i2nsf-applicability-17	28
Authors' Addresses	28

1. Introduction

Interface to Network Security Functions (I2NSF) defines a framework and interfaces for interacting with Network Security Functions (NSFs). Note that an NSF is defined as software that provides a set of security-related services, such as (i) detecting unwanted activity, (ii) blocking or mitigating the effect of such unwanted activity in order to fulfill service requirements, and (iii) supporting communication stream integrity and confidentiality [i2nsf-terminology].

The I2NSF framework allows heterogeneous NSFs developed by different security solution vendors to be used in the Network Functions Virtualization (NFV) environment [ETSI-NFV] by utilizing the capabilities of such NSFs through I2NSF interfaces such as Customer-Facing Interface [consumer-facing-inf-dm] and NSF-Facing Interface

[nsf-facing-inf-dm]. In the I2NSF framework, each NSF initially registers the profile of its own capabilities with the Security Controller (i.e., network operator management system [RFC8329]) of the I2NSF system via the Registration Interface [registration-inf-dm]. This registration enables an I2NSF User (i.e., network security administrator) to select and use the NSF to enforce a given security policy. Note that Developer's Management System (DMS) is management software that provides a vendor's security service software as a Virtual Network Function (VNF) in an NFV environment (or middlebox in the legacy network) as an NSF, and registers the capabilities of an NSF into Security Controller via Registration Interface for a security service [RFC8329].

Security Controller maintains the mapping between a capability and an NSF, so it can perform to translate a high-level security policy received from I2NSF User to a low-level security policy configured and enforced in an NSF [policy-translation]. Security Controller can monitor the states and security attacks in NSFs through NSF monitoring [nsf-monitoring-dm].

This document illustrates the applicability of the I2NSF framework with five different scenarios:

1. The enforcement of time-dependent web access control.
2. The support of intent-based security services through I2NSF and Security Policy Translator [policy-translation].
3. The application of I2NSF to a Service Function Chaining (SFC) environment [RFC7665].
4. The integration of the I2NSF framework with Software-Defined Networking (SDN) [RFC7149] to provide different security functionality such as firewalls [opsawg-firewalls], Deep Packet Inspection (DPI), and Distributed Denial of Service (DDoS) attack mitigation.
5. The use of Network Functions Virtualization (NFV) [ETSI-NFV] as a supporting technology.

The implementation of I2NSF in these scenarios has allowed us to verify the applicability and effectiveness of the I2NSF framework for a variety of use cases.

2. Terminology

This document uses the terminology described in [RFC7665], [RFC7149], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ITU-T.X.800], [NFV-Terminology], [RFC8329], and [i2nsf-terminology]. In addition, the following terms are defined below:

- o Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation.
- o Centralized Firewall System: A centralized firewall that can establish and distribute policy rules into network resources for efficient firewall management.
- o Centralized VoIP Security System: A centralized security system that handles the security functions required for VoIP and VoLTE services.
- o Firewall: A service function at the junction of two network segments that inspects some suspicious packets that attempt to cross the boundary. It also rejects any packet that does not satisfy certain criteria for, for example, disallowed port numbers or IP addresses.
- o Network Function: A functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behavior [NFV-Terminology].
- o Network Functions Virtualization (NFV): A principle of separating network functions (or network security functions) from the hardware they run on by using virtual hardware abstraction [NFV-Terminology].
- o Network Security Function (NSF): Software that provides a set of security-related services. Examples include detecting unwanted activity and blocking or mitigating the effect of such unwanted activity in order to fulfill service requirements. The NSF can also help in supporting communication stream integrity and confidentiality [i2nsf-terminology].
- o Security Policy Translator (SPT): Software that translates a high-level security policy for the Consumer-Facing Interface into a low-level security policy for the NSF-Facing Interface [policy-translation]. The SPT is a core part of the Security Controller in the I2NSF system.

- o Service Function Chaining (SFC): The execution of an ordered set of abstract service functions (i.e., network functions) according to ordering constraints that must be applied to packets, frames, and flows selected as a result of classification. The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied [RFC7665].
- o Software-Defined Networking (SDN): A set of techniques that enables to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

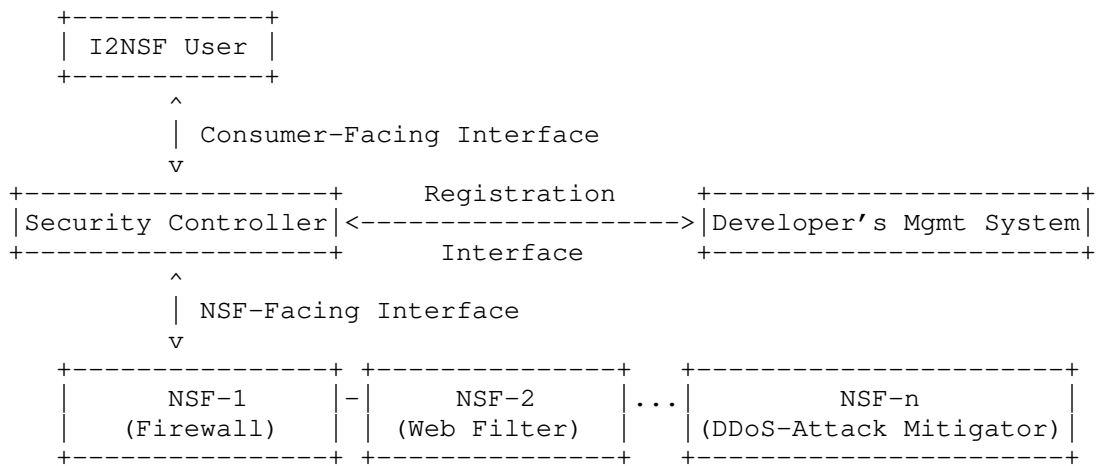


Figure 1: I2NSF Framework

3. I2NSF Framework

This section summarizes the I2NSF framework as defined in [RFC8329]. As shown in Figure 1, an I2NSF User can use security functions by delivering high-level security policies, which specify security requirements that the I2NSF user wants to enforce, to the Security Controller via the Consumer-Facing Interface (CFI) [consumer-facing-inf-dm].

The Security Controller receives and analyzes the high-level security policies from an I2NSF User, and identifies what types of security capabilities are required to meet these high-level security policies. The Security Controller then identifies NSFs that have the required security capabilities, and generates low-level security policies for

each of the NSFs so that the high-level security policies are eventually enforced by those NSFs [policy-translation]. Finally, the Security Controller sends the generated low-level security policies to the NSFs via the NSF-Facing Interface (NFI) [nsf-facing-inf-dm].

As shown in Figure 1, with a Developer's Management System (called DMS), developers (or vendors) inform the Security Controller of the capabilities of the NSFs through the Registration Interface (RI) [registration-inf-dm] for registering (or deregistering) the corresponding NSFs. Note that the lifecycle management of NSF code from DMS (e.g., downloading of NSF modules and testing of NSF code) is out of scope for I2NSF.

The Consumer-Facing Interface can be implemented with the Consumer-Facing Interface YANG data model [consumer-facing-inf-dm] using RESTCONF [RFC8040] which befits a web-based user interface for an I2NSF User to send a Security Controller a high-level security policy. Data models specified by YANG [RFC6020] describe high-level security policies to be specified by an I2NSF User. The data model defined in [consumer-facing-inf-dm] can be used for the I2NSF Consumer-Facing Interface. Note that an inside attacker at the I2NSF User can misuse the I2NSF system so that the network system under the I2NSF system is vulnerable to security attacks. To handle this type of threat, the Security Controller needs to monitor the activities of all the I2NSF Users as well as the NSFs through the I2NSF NSF monitoring functionality [nsf-monitoring-dm]. Note that the monitoring of the I2NSF Users is out of scope for I2NSF.

The NSF-Facing Interface can be implemented with the NSF-Facing Interface YANG data model [nsf-facing-inf-dm] using NETCONF [RFC6241] which befits a command-line-based remote-procedure call for a Security Controller to configure an NSF with a low-level security policy. Data models specified by YANG [RFC6020] describe low-level security policies for the sake of NSFs, which are translated from the high-level security policies by the Security Controller. The data model defined in [nsf-facing-inf-dm] can be used for the I2NSF NSF-Facing Interface.

The Registration Interface can be implemented with the Registration Interface YANG data model [registration-inf-dm] using NETCONF [RFC6241] which befits a command-line-based remote-procedure call for a DMS to send a Security Controller an NSF's capability information. Data models specified by YANG [RFC6020] describe the registration of an NSF's capabilities to enforce security services at the NSF. The data model defined in [registration-inf-dm] can be used for the I2NSF Registration Interface.

The I2NSF framework can chain multiple NSFs to implement low-level security policies with the SFC architecture [RFC7665].

The following sections describe different security service scenarios illustrating the applicability of the I2NSF framework.

```
<?xml version="1.0" encoding="UTF-8" ?>
<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy">
  <policy-name>block_website</policy-name>
  <rule>
    <rule-name>block_website_during_working_hours</rule-name>
    <event>
      <time-information>
        <begin-time>09:00</begin-time>
        <end-time>18:00</end-time>
      </time-information>
    </event>
    <condition>
      <firewall-condition>
        <source-target>
          <src-target>Staff_Members'_PCs</src-target>
        </source-target>
      </firewall-condition>
      <custom-condition>
        <destination-target>
          <dest-target>SNS_Websites</dest-target>
        </destination-target>
      </custom-condition>
    </condition>
    <action>
      <primary-action>drop</primary-action>
    </action>
  </rule>
</policy>
```

Figure 2: A High-level Security Policy XML File for Time-based Web Filter

```

<?xml version="1.0" encoding="UTF-8" ?>
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
  <system-policy>
    <system-policy-name>block_website</system-policy-name>
    <rules>
      <rule-name>block_website_during_working_hours</rule-name>
      <time-intervals>
        <absolute-time-interval>
          <begin-time>09:00</begin-time>
          <end-time>18:00</end-time>
        </absolute-time-interval>
      </time-intervals>
      <condition-clause-container>
        <packet-security-ipv6-condition>
          <pkt-sec-ipv6-src>
            <ipv6-address>
              <ipv6>2001:DB8:10:1::10</ipv6>
              <ipv6>2001:DB8:10:1::20</ipv6>
              <ipv6>2001:DB8:10:1::30</ipv6>
            </ipv6-address>
          </pkt-sec-ipv6-src>
        </packet-security-ipv6-condition>
        <packet-security-url-category-condition>
          <user-defined-category>example1.com</user-defined-category>
          <user-defined-category>example2.com</user-defined-category>
          <user-defined-category>example3.com</user-defined-category>
          <user-defined-category>example4.com</user-defined-category>
        </packet-security-url-category-condition>
      </condition-clause-container>
      <action-clause-container>
        <packet-action>
          <egress-action>drop</egress-action>
        </packet-action>
      </action-clause-container>
    </rules>
  </system-policy>
</i2nsf-security-policy>

```

Figure 3: A Low-level Security Policy XML File for Time-based Web Filter

4. Time-dependent Web Access Control Service

This service scenario assumes that an enterprise network administrator wants to control the staff members' access to a particular Internet service (e.g., social networking service (SNS)) during business hours. The following is an example high-level

security policy rule for a web filter that the administrator requests: Block the staff members' access to SNS websites from 9 AM (i.e., 09:00) to 6 PM (i.e., 18:00) by dropping their packets. Figure 2 is a high-level security policy XML code for the web filter that is sent from the I2NSF User to the Security Controller via the Consumer-Facing Interface [consumer-facing-inf-dm].

The security policy name is "block_website" with the tag "policy-name", and the security policy rule name is "block_website_during_working_hours" with the tag "rule-name". The filtering event has the time span where the filtering begin time is the time "09:00" (i.e., 9:00AM) with the tag "begin-time", and the filtering end time is the time "18:00" (i.e., 6:00PM) with the tag "end-time". The filtering condition has the source target of "Staff_Members'_PCs" with the tag "src-target", and the destination target of "SNS_Websites" with the tag "dest-target".

Assume that "Staff_Members'_PCs" are 2001:DB8:10:1::10, 2001:DB8:10:1::20, and 2001:DB8:10:1::30, and that "SNS_Websites" are example1.com, example2.com, example3.com, and example4.com, as shown in Figure 3. Note that Figure 3 is a low-level security policy XML code for the web filter that is sent from the Security Controller to an NSF via the NSF-Facing Interface [nsf-facing-inf-dm].

The source target can be translated by the Security Policy Translator (SPT) in the Security Controller to the IP addresses of computers (or mobile devices) used by the staff members. Refer to Section 5 for the detailed description of the SPT. The destination target can also be translated by the SPT to the actual websites corresponding to the symbolic website name "SNS_Websites", and then either each website's URL or the corresponding IP address(es) can be used by both firewall and web filter. The action is to "drop" the packets satisfying the above event and condition with the tag "primary-action".

After receiving the high-level security policy, the Security Controller identifies required security capabilities, e.g., IP address and port number inspection capabilities and URL inspection capability. In this scenario, it is assumed that the IP address and port number inspection capabilities are required to check whether a received packet is an HTTP-session packet from a staff member, which is part of an HTTP session generated by the staff member. The URL inspection capability is required to check whether the target URL of a received packet is one of the target websites (i.e., example1.com, example2.com, example3.com, and example4.com) or not.

The Security Controller maintains the security capabilities of each active NSF in the I2NSF system, which have been reported by the Developer's Management System via the Registration interface. Based

on this information, the Security Controller identifies NSFs that can perform the IP address and port number inspection and URL inspection through the security policy translation in Section 5. In this scenario, it is assumed that a firewall NSF has the IP address and port number inspection capabilities and a web filter NSF has URL inspection capability.

The Security Controller generates a low-level security policy for the NSFs to perform IP address and port number inspection, URL inspection, and time checking, which is shown in Figure 3. Specifically, the Security Controller may interoperate with an access control server in the enterprise network in order to retrieve the information (e.g., IP address in use, company identifier (ID), and role) of each employee that is currently using the network. Based on the retrieved information, the Security Controller generates a low-level security policy to check whether the source IP address of a received packet matches any one being used by a staff member.

In addition, the low-level security policy's rule (shortly, low-level security rule) should be able to determine that a received packet uses either the HTTP protocol without Transport Layer Security (TLS) [RFC8446] or the HTTP protocol with TLS as HTTPS. The low-level security rule for web filter checks that the target URL field of a received packet is equal to one of the target SNS websites (i.e., example1.com, example2.com, example3.com, and example4.com), or that the destination IP address of a received packet is an IP address corresponding to one of the SNS websites. Note that if HTTPS is used for an HTTP-session packet, the HTTP protocol header is encrypted, so the URL information may not be seen from the packet for the web filtering. Thus, the IP address(es) corresponding to the target URL needs to be obtained from the certificate in TLS versions prior to 1.3 [RFC8446] or the Server Name Indication (SNI) in a TCP-session packet in TLS versions without the encrypted SNI [tls-esni]. Also, to obtain IP address(es) corresponding to a target URL, the DNS name resolution process can be observed through a packet capturing tool because the DNS name resolution will translate the target URL into IP address(es). The IP addresses obtained through either TLS or DNS can be used by both firewall and web filter for whitelisting or blacklisting the TCP five-tuples of HTTP sessions.

Finally, the Security Controller sends the low-level security policy of the IP address and port number inspection to the firewall NSF and the low-level security policy for URL inspection to the web filter NSF.

The following describes how the time-dependent web access control service is enforced by the NSFs of firewall and web filter.

1. A staff member tries to access one of the target SNS websites (i.e., example1.com, example2.com, example3.com, and example4.com) during business hours, e.g., 10 AM.
2. The packet is forwarded from the staff member's device to the firewall, and the firewall checks the source IP address and port number. Now the firewall identifies the received packet is an HTTP-session packet from the staff member.
3. The firewall triggers the web filter to further inspect the packet, and the packet is forwarded from the firewall to the web filter. The SFC architecture [RFC7665] can be utilized to support such packet forwarding in the I2NSF framework.
4. The web filter checks the received packet's target URL field or its destination IP address corresponding to the target URL, and detects that the packet is being sent to the server for example1.com. The web filter then checks that the current time is within business hours. If so, the web filter drops the packet, and consequently the staff member's access to one of the SNS websites (i.e., example1.com, example2.com, example3.com, and example4.com) during business hours is blocked.

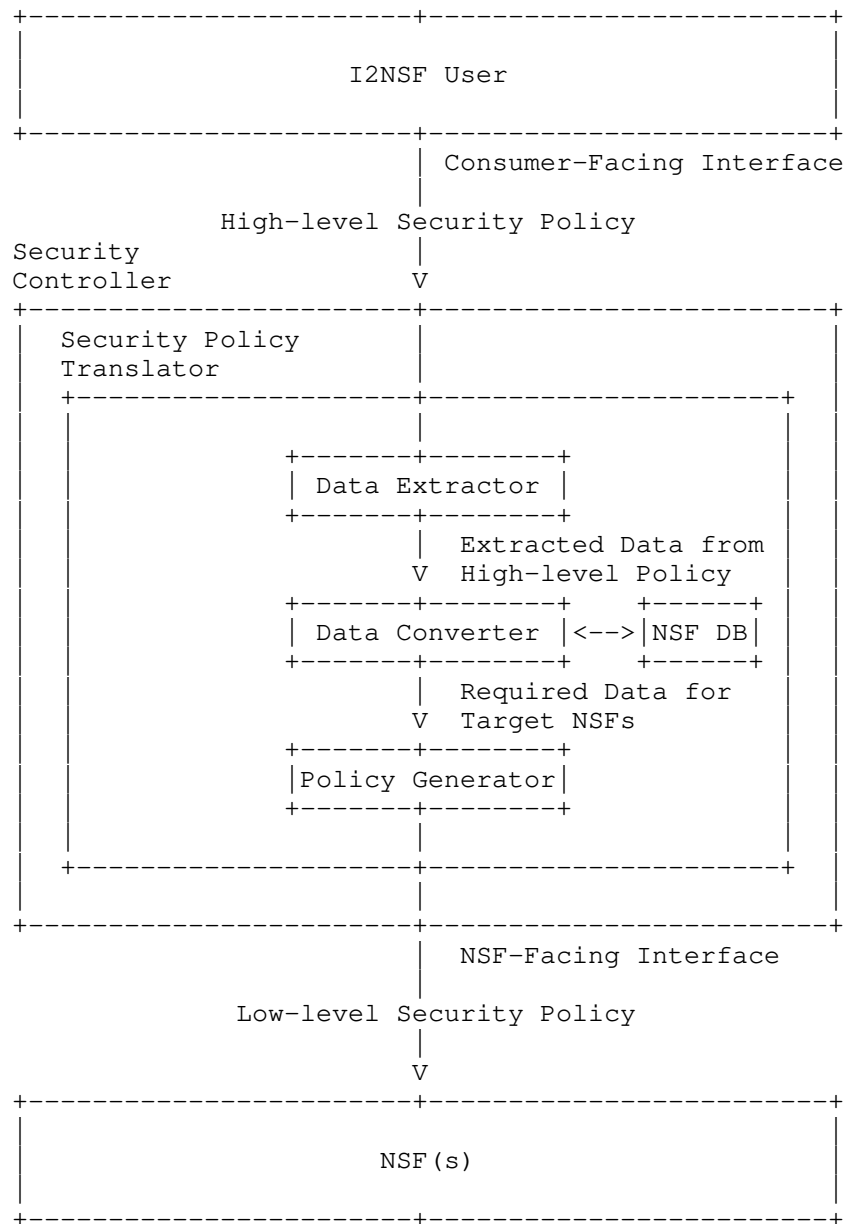


Figure 4: Security Policy Translation and Enforcement in I2NSF System

5. Intent-based Security Services

I2NSF aims at providing intent-based security services to configure specific security policies into NSFs with customer-friendly security policies at a high level. For example, when an I2NSF User submits a high-level security policy (e.g., web filtering as shown in Figure 2) to the Security Controller, the Security Policy Translator (SPT) in the Security Controller will translate it into the corresponding low-level security policy as shown in Figure 3 [policy-translation]. A security administrator using the I2NSF User can describe a security policy without the knowledge of the detailed information about subjects (e.g., source and destination) and objects (e.g., web traffic) of the security policy's rule(s).

Figure 4 shows the security policy translation and enforcement in the I2NSF system [policy-translation]. As shown in Figure 4, an I2NSF User delivers a high-level security policy to the Security Controller using the Consumer-Facing Interface (denoted as CFI). The high-level security policy is translated by the SPT in the Security Controller into the corresponding low-level security policy which is understandable by target NSF(s). The Security Controller delivers the low-level security policy to the appropriate NSF(s) to enforce the policy's rules.

The SPT consists of three modules for security policy translations such as Data Extractor, Data Converter, and Policy Generator, as shown in Figure 4. The Data Extractor extracts data from a high-level security policy delivered by the I2NSF User. The data correspond to the leaf nodes in the YANG data model for the Consumer-Facing Interface. In the high-level policy in Figure 2, the data are the tag values of policy-name, rule-name, begin-time, end-time, src-target, dest-target, and primary-action. That is, the tag values are "block_website", "block_website_during_working_hours", "09:00", "18:00", "Staff_Members'_PCs", "SNS_Websites", and "drop."

The Data Converter converts the extracted high-level policy data received from the Data Extractor into the corresponding low-level policy data. The low-level policy data have the capability information of NSFs to be selected as target NSFs for the required security service enforcement specified by the high-level security policy. The tag values in the extracted high-level policy data are replaced with the tag values in the low-level policy data, which are the leaf nodes of the YANG data model for the NSF-Facing Interface (denoted as NFI). The value of each leaf node in CFI is translated into the value of the corresponding leaf node in NFI. For example, "block_website" of policy-name in CFI (in Figure 2) is translated into "block_website" of system-policy-name in NFI (in Figure 3). The tag values of rule-name, begin-time, end-time, and primary-action in

CFI are mapped into the same values of rule-name, begin-time, end-time, and egress-action in NFI. However, the tag values of src-target and dest-target in CFI are translated into IP addresses and URLs, respectively, for the sake of NFI. That is, "Staff_Members'_PCs" of CFI is translated into three IPv6 addresses such as "2001:DB8:10:1::10", "2001:DB8:10:1::20", and "2001:DB8:10:1::30" for the sake of NFI. Also, "SNS_Websites" of CFI is translated into four URLs such as "example1.com", "example2.com", "example3.com", and "example4.com" for the sake of NFI. In addition to the data conversion, the Data Converter searches for appropriate NSFs having capabilities corresponding to the leaf nodes of the YANG data model for NFI. For the data conversion and NSF search, an NSF database (denoted as NSF DB) can be consulted, as shown in Figure 4, because the NSF DB has the capability information of NSFs that the DMS(s) registered with the Security Controller using the Registration Interface.

The Policy Generator generates a low-level security policy corresponding to the low-level policy data made by the Data Converter per a target NSF. That is, the Policy Generator can build such a low-level security policy XML file like Figure 3 with the NSF DB because the NSF DB has the mapping information between the CFI YANG data model and the NFI YANG data model.

Therefore, by allowing the I2NSF User to express its security policy without knowing the detailed information of entities for security policies, the I2NSF can efficiently support the intent-based security services with the help of the security policy translator along with the NSF DB.

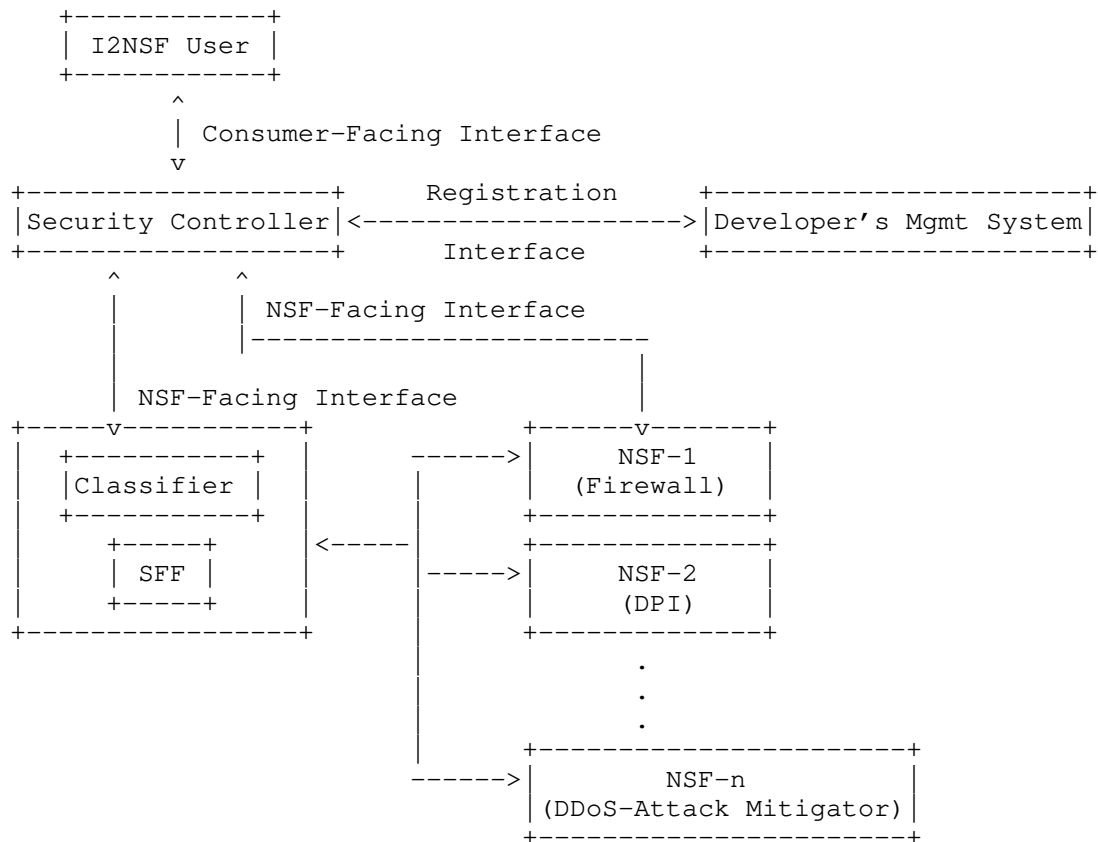


Figure 5: An I2NSF Framework with SFC

6. I2NSF Framework with SFC

In the I2NSF architecture, an NSF can trigger an advanced security action (e.g., DPI or DDoS attack mitigation) on a packet based on the result of its own security inspection of the packet. For example, a firewall triggers further inspection of a suspicious packet with DPI. For this advanced security action to be fulfilled, the suspicious packet should be forwarded from the current NSF to the successor NSF. SFC [RFC7665] is a technology that enables this advanced security action by steering a packet with multiple service functions (e.g., NSFs), and this technology can be utilized by the I2NSF architecture to support the advanced security action.

Figure 5 shows an I2NSF framework with the support of SFC. As shown in the figure, SFC generally requires classifiers and service function forwarders (SFFs); classifiers are responsible for

determining which service function path (SFP) (i.e., an ordered sequence of service functions) a given packet should pass through, according to pre-configured classification rules, and SFFs perform forwarding the given packet to the next service function (e.g., NSF) on the SFP of the packet by referring to their forwarding tables. In the I2NSF architecture with SFC, the Security Controller can take responsibilities of generating classification rules for classifiers and forwarding tables for SFFs. By analyzing high-level security policies from I2NSF users, the Security Controller can construct SFPs that are required to meet the high-level security policies, generates classification rules of the SFPs, and then configures classifiers with the classification rules over NSF-Facing Interface so that relevant traffic packets can follow the SFPs. Also, based on the global view of NSF instances available in the system, the Security Controller constructs forwarding tables, which are required for SFFs to forward a given packet to the next NSF over the SFP, and configures SFFs with those forwarding tables over NSF-Facing Interface.

To trigger an advanced security action in the I2NSF architecture, the current NSF appends metadata describing the security capability required to the suspicious packet via a network service header (NSH) [RFC8300]. It then sends the packet to the classifier. Based on the metadata information, the classifier searches an SFP which includes an NSF with the required security capability, changes the SFP-related information (e.g., service path identifier and service index [RFC8300]) of the packet with the new SFP that has been found, and then forwards the packet to the SFF. When receiving the packet, the SFF checks the SFP-related information such as the service path identifier and service index contained in the packet and forwards the packet to the next NSF on the SFP of the packet, according to its forwarding table.

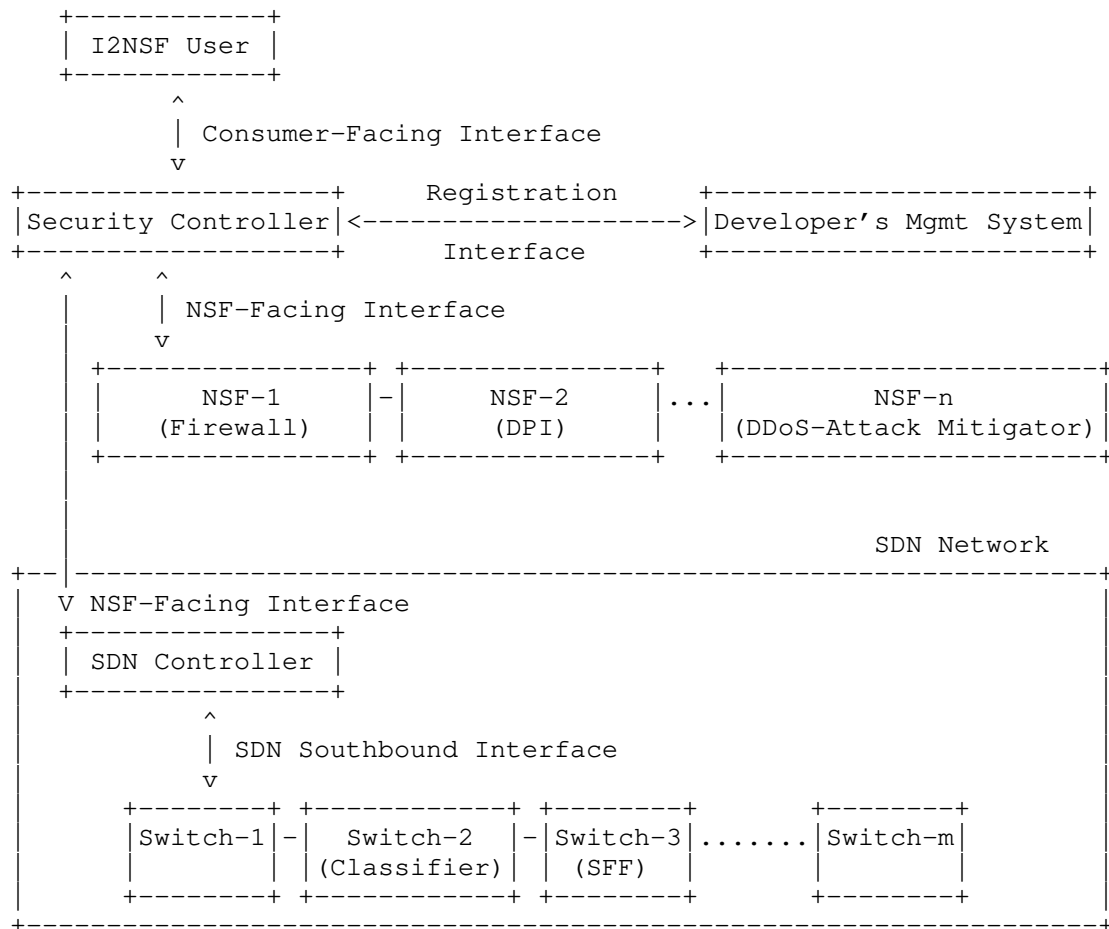


Figure 6: An I2NSF Framework with SDN Network

7. I2NSF Framework with SDN

This section describes an I2NSF framework with SDN for I2NSF applicability and use cases, such as firewall, deep packet inspection, and DDoS-attack mitigation functions. SDN enables some packet filtering rules to be enforced in network forwarding elements (e.g., switch) by controlling their packet forwarding rules. By taking advantage of this capability of SDN, it is possible to optimize the process of security service enforcement in the I2NSF system. For example, for efficient firewall services, simple packet filtering can be performed by SDN forwarding elements (e.g., switches), and complicated packet filtering based on packet payloads can be performed by a firewall NSF. This optimized firewall using

both SDN forwarding elements and a firewall NSF is more efficient than a firewall where SDN forwarding elements forward all the packets to a firewall NSF for packet filtering. This is because packets to be filtered out can be early dropped by SDN forwarding elements without consuming further network bandwidth due to the forwarding of the packets to the firewall NSF.

Figure 6 shows an I2NSF framework [RFC8329] with SDN networks to support network-based security services. In this system, the enforcement of security policy rules is divided into the SDN forwarding elements (e.g., a switch running as either a hardware middle box or a software virtual switch) and NSFs (e.g., a firewall running in a form of a VNF [ETSI-NFV]). Note that NSFs are created or removed by the NFV Management and Orchestration (MANO) [ETSI-NFV-MANO], performing the lifecycle management of NSFs as VNFs. Refer to Section 8 for the detailed discussion of the NSF lifecycle management in the NFV MANO for I2NSF. For security policy enforcement (e.g., packet filtering), the Security Controller instructs the SDN Controller via NSF-Facing Interface so that SDN forwarding elements can perform the required security services with flow tables under the supervision of the SDN Controller.

As an example, let us consider two different types of security rules: Rule A is a simple packet filtering rule that checks only the IP address and port number of a given packet, whereas rule B is a time-consuming packet inspection rule for analyzing whether an attached file being transmitted over a flow of packets contains malware. Rule A can be translated into packet forwarding rules of SDN forwarding elements and thus be enforced by these elements. In contrast, rule B cannot be enforced by forwarding elements, but it has to be enforced by NSFs with anti-malware capability. Specifically, a flow of packets is forwarded to and reassembled by an NSF to reconstruct the attached file stored in the flow of packets. The NSF then analyzes the file to check the existence of malware. If the file contains malware, the NSF drops the packets.

In an I2NSF framework with SDN, the Security Controller can analyze given security policy rules and automatically determine which of the given security policy rules should be enforced by SDN forwarding elements and which should be enforced by NSFs. If some of the given rules requires security capabilities that can be provided by SDN forwarding elements, then the Security Controller instructs the SDN Controller via NSF-Facing Interface so that SDN forwarding elements can enforce those security policy rules with flow tables under the supervision of the SDN Controller. Or if some rules require security capabilities that cannot be provided by SDN forwarding elements but by NSFs, then the Security Controller instructs relevant NSFs to enforce those rules.

The distinction between software-based SDN forwarding elements and NSFs, which can both run as VNFs, may be necessary for some management purposes in this system. Note that an SDN forwarding element (i.e., switch) is a specific type of VNF rather than an NSF because an NSF is for security services rather than for packet forwarding. For this distinction, we can take advantage of the NFV MANO where there is a subsystem that maintains the descriptions of the capabilities each VNF can offer [ETSI-NFV-MANO]. This subsystem can determine whether a given software element (VNF instance) is an NSF or a virtualized SDN switch. For example, if a VNF instance has anti-malware capability according to the description of the VNF, it could be considered as an NSF. A VNF onboarding system [VNF-ONBOARDING] can be used as such a subsystem that maintains the descriptions of each VNF to tell whether a VNF instance is for an NSF or for a virtualized SDN switch.

For the support of SFC in the I2NSF framework with SDN, as shown in Figure 6, network forwarding elements (e.g., switch) can play the role of either SFC Classifier or SFF, which are explained in Section 6. Classifier and SFF have an NSF-Facing Interface with Security Controller. This interface is used to update security service function chaining information for traffic flows. For example, when it needs to update an SFP for a traffic flow in an SDN network, as shown in Figure 6, SFF (denoted as Switch-3) asks Security Controller to update the SFP for the traffic flow (needing another security service as an NSF) via NSF-Facing Interface. This update lets Security Controller ask Classifier (denoted as Switch-2) to update the mapping between the traffic flow and SFP in Classifier via NSF-Facing Interface.

The following subsections introduce three use cases from [RFC8192] for cloud-based security services: (i) firewall system, (ii) deep packet inspection system, and (iii) attack mitigation system.

7.1. Firewall: Centralized Firewall System

A centralized network firewall can manage each network resource and apply common rules to individual network elements (e.g., switch). The centralized network firewall controls each forwarding element, and firewall rules can be added or deleted dynamically.

A time-based firewall can be enforced with packet filtering rules and a time span (e.g., work hours). With this time-based firewall, a time-based security policy can be enforced, as explained in Section 4. For example, employees at a company are allowed to access social networking service websites during lunch time or after work hours.

7.2. Deep Packet Inspection: Centralized VoIP/VoLTE Security System

A centralized VoIP/VoLTE security system can monitor each VoIP/VoLTE flow and manage VoIP/VoLTE security rules, according to the configuration of a VoIP/VoLTE security service called VoIP Intrusion Prevention System (IPS). This centralized VoIP/VoLTE security system controls each switch for the VoIP/VoLTE call flow management by manipulating the rules that can be added, deleted or modified dynamically.

The centralized VoIP/VoLTE security system can cooperate with a network firewall to realize VoIP/VoLTE security service. Specifically, a network firewall performs the basic security check of an unknown flow's packet observed by a switch. If the network firewall detects that the packet is an unknown VoIP call flow's packet that exhibits some suspicious patterns, then it triggers the VoIP/VoLTE security system for more specialized security analysis of the suspicious VoIP call packet.

7.3. Attack Mitigation: Centralized DDoS-attack Mitigation System

A centralized DDoS-attack mitigation can manage each network resource and configure rules to each switch for DDoS-attack mitigation (called DDoS-attack Mitigator) on a common server. The centralized DDoS-attack mitigation system defends servers against DDoS attacks outside the private network, that is, from public networks [RFC8612][dots-architecture].

Servers are categorized into stateless servers (e.g., DNS servers) and stateful servers (e.g., web servers). For DDoS-attack mitigation, the forwarding of traffic flows in switches can be dynamically configured such that malicious traffic flows are handled by the paths separated from normal traffic flows in order to minimize the impact of those malicious traffic on the servers. This flow path separation can be done by a flow forwarding path management scheme [dots-architecture][AVANT-GUARD]. This management should consider the load balance among the switches for the defense against DDoS attacks.

So far this section has described the three use cases for network-based security services using the I2NSF framework with SDN networks. To support these use cases in the proposed data-driven security service framework, YANG data models described in [consumer-facing-inf-dm], [nsf-facing-inf-dm], and [registration-inf-dm] can be used as Consumer-Facing Interface, NSF-Facing Interface, and Registration Interface, respectively, along with RESTCONF [RFC8040] and NETCONF [RFC6241].

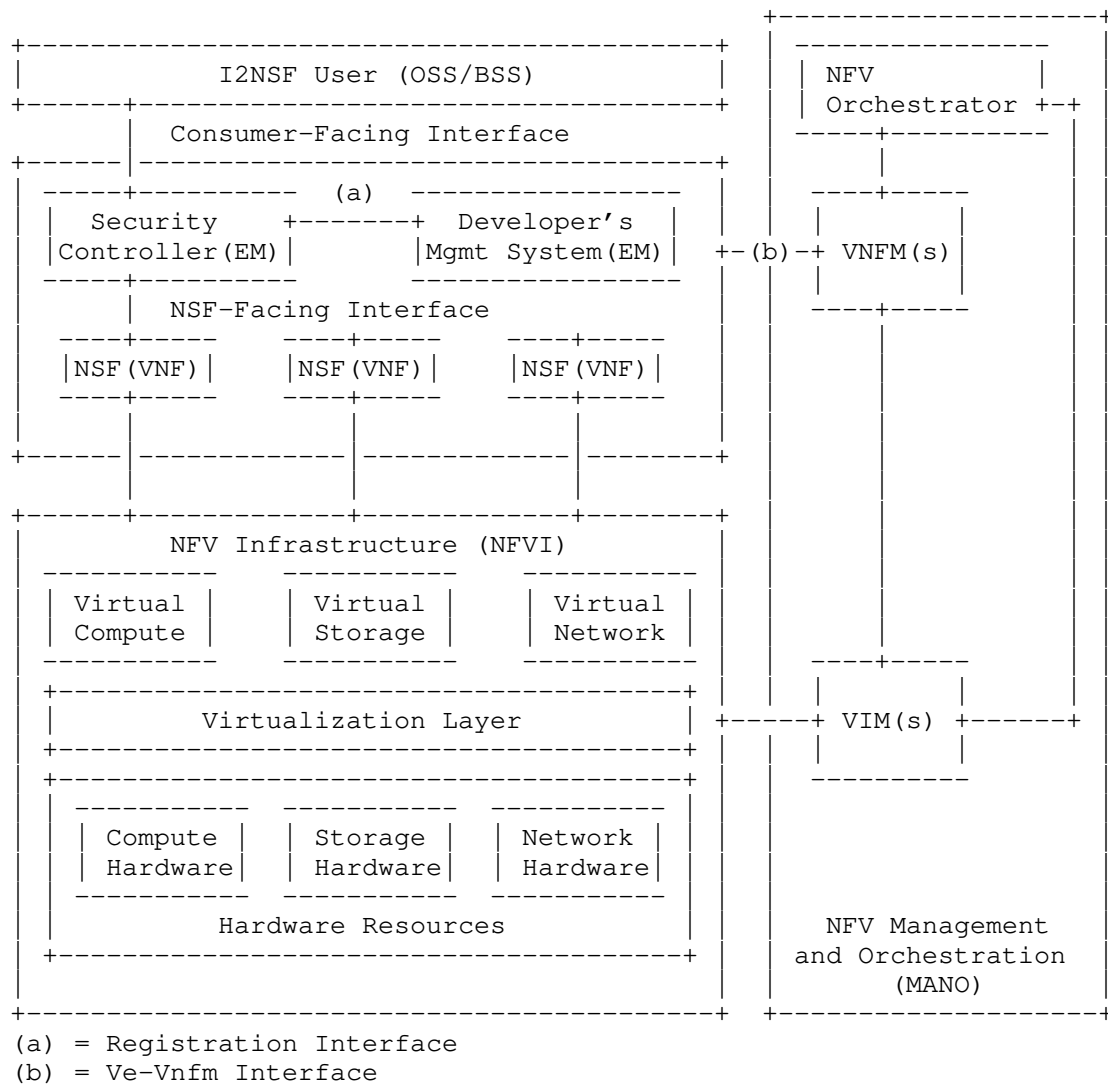


Figure 7: I2NSF Framework Implementation with respect to the NFV Reference Architectural Framework

8. I2NSF Framework with NFV

This section discusses the implementation of the I2NSF framework using Network Functions Virtualization (NFV).

NFV is a promising technology for improving the elasticity and efficiency of network resource utilization. In NFV environments,

NSFs can be deployed in the forms of software-based virtual instances rather than physical appliances. Virtualizing NSFs makes it possible to rapidly and flexibly respond to the amount of service requests by dynamically increasing or decreasing the number of NSF instances. Moreover, NFV technology facilitates flexibly including or excluding NSFs from multiple security solution vendors according to the changes on security requirements. In order to take advantages of the NFV technology, the I2NSF framework can be implemented on top of an NFV infrastructure as show in Figure 7.

Figure 7 shows an I2NSF framework implementation based on the NFV reference architecture that the European Telecommunications Standards Institute (ETSI) defines [ETSI-NFV]. The NSFs are deployed as VNFs in Figure 7. The Developer's Management System (DMS) in the I2NSF framework is responsible for registering capability information of NSFs into the Security Controller. However, those NSFs are created or removed by a virtual network function manager (VNFM) in the NFV MANO that performs the lifecycle management of VNFs. Note that the lifecycle management of VNFs is out of scope for I2NSF. The Security Controller controls and monitors the configurations (e.g., function parameters and security policy rules) of VNFs via the NSF-Facing Interface along with the NSF monitoring capability [nsf-facing-inf-dm][nsf-monitoring-dm]. Both the DMS and Security Controller can be implemented as the Element Managements (EMs) in the NFV architecture. Finally, the I2NSF User can be implemented as OSS/BSS (Operational Support Systems/Business Support Systems) in the NFV architecture that provides interfaces for users in the NFV system.

The operation procedure in the I2NSF framework based on the NFV architecture is as follows:

1. The VNFM has a set of virtual machine (VM) images of NSFs, and each VM image can be used to create an NSF instance that provides a set of security capabilities. The DMS initially registers a mapping table of the ID of each VM image and the set of capabilities that can be provided by an NSF instance created from the VM image into the Security Controller.
2. If the Security Controller does not have any instantiated NSF that has the set of capabilities required to meet the security requirements from users, it searches the mapping table (registered by the DMS) for the VM image ID corresponding to the required set of capabilities.
3. The Security Controller requests the DMS to instantiate an NSF with the VM image ID via VNFM.

4. When receiving the instantiation request, the VNFM first asks the NFV orchestrator for the permission required to create the NSF instance, requests the VIM to allocate resources for the NSF instance, and finally creates the NSF instance based on the allocated resources.
5. Once the NSF instance has been created by the VNFM, the DMS performs the initial configurations of the NSF instance and then notifies the Security Controller of the NSF instance.
6. After being notified of the created NSF instance, the Security Controller delivers low-level security policy rules to the NSF instance for policy enforcement.

We can conclude that the I2NSF framework can be implemented based on the NFV architecture framework. Note that the registration of the capabilities of NSFs is performed through the Registration Interface and the lifecycle management for NSFs (VNFs) is performed through the Ve-Vnfm interface between the DMS and VNFM, as shown in Figure 7.

9. Security Considerations

The same security considerations for the I2NSF framework [RFC8329] are applicable to this document.

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

The role of the DMS is to provide an I2NSF system with the software packages or images for NSF execution. The DMS must not access NSFs in activated status. An inside attacker or a supply chain attacker at the DMS can seriously weaken the I2NSF system's security. A malicious DMS is relevant to an insider attack, and a compromised DMS is relevant to a supply chain attack. A malicious (or compromised) DMS could register an NSF of its choice in response to a capability request by the Security Controller. As a result, a malicious DMS can attack the I2NSF system by providing malicious NSFs with arbitrary capabilities to include potentially controlling those NSFs in real time. An unwitting DMS could be compromised and the infrastructure of the DMS could be coerced into distributing modified NSFs as well.

To deal with these types of threats, an I2NSF system should not use NSFs from an untrusted DMS or without prior testing. The practices by which these packages are downloaded and loaded into the system are out of scope for I2NSF.

I2NSF system operators should audit and monitor interactions with DMSs. Additionally, the operators should monitor the running NSFs

through the I2NSF NSF Monitoring Interface [nsf-monitoring-dm] as part of the I2NSF NSF-Facing Interface. Note that the mechanics for monitoring the DMSs are out of scope for I2NSF.

10. Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea MSIT (Ministry of Science and ICT) (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This work has been partially supported by the European Commission under Horizon 2020 grant agreement no. 700199 "Securing against intruders and other threats through a NFV-enabled environment (SHIELD)". This support does not imply endorsement.

11. Contributors

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

- o Hyoungshick Kim (Sungkyunkwan University)
- o Jinyong Tim Kim (Sungkyunkwan University)
- o Hyunsik Yang (Soongsil University)
- o Younghan Kim (Soongsil University)
- o Jung-Soo Park (ETRI)
- o Se-Hui Lee (Korea Telecom)
- o Mohamed Boucadair (Orange)

12. References

12.1. Normative References

[AVANT-GUARD]

Shin, S., Yegneswaran, V., Porras, P., and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks", ACM CCS, November 2013.

[consumer-facing-inf-dm]

Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares,
"I2NSF Consumer-Facing Interface YANG Data Model", draft-
ietf-i2nsf-consumer-facing-interface-dm-06 (work in
progress), July 2019.

[dots-architecture]

Mortensen, A., Reddy, T., Andreasen, F., Teague, N., and
R. Compton, "Distributed-Denial-of-Service Open Threat
Signaling (DOTS) Architecture", draft-ietf-dots-
architecture-14 (work in progress), May 2019.

[ETSI-NFV]

"Network Functions Virtualisation (NFV); Architectural
Framework", Available:
[https://www.etsi.org/deliver/etsi_gs/
nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf](https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf), October
2013.

[ITU-T.Y.3300]

"Framework of Software-Defined Networking",
Available: <https://www.itu.int/rec/T-REC-Y.3300-201406-I>,
June 2014.

[NFV-Terminology]

"Network Functions Virtualisation (NFV); Terminology for
Main Concepts in NFV", Available:
[https://www.etsi.org/deliver/etsi_gs/
NFV/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf](https://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf),
December 2014.

[nsf-facing-inf-dm]

Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin,
"I2NSF Network Security Function-Facing Interface YANG
Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-07
(work in progress), July 2019.

[nsf-monitoring-dm]

Jeong, J., Chung, C., Hares, S., Xia, L., and H. Birkholz,
"I2NSF NSF Monitoring YANG Data Model", draft-ietf-i2nsf-
nsf-monitoring-data-model-01 (work in progress), July
2019.

[ONF-SDN-Architecture]

"SDN Architecture (Issue 1.1)", Available:
[https://www.opennetworking.org/wp-
content/uploads/2014/10/TR-
521_SDN_Architecture_issue_1.1.pdf](https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf), June 2016.

[registration-inf-dm]

Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-ietf-i2nsf-registration-interface-dm-05 (work in progress), July 2019.

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

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, July 2017.

[RFC8300] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", RFC 8300, January 2018.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

[RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open Threat Signaling (DOTS) Requirements", RFC 8612, May 2019.

12.2. Informative References

[ETSI-NFV-MANO]

"Network Functions Virtualisation (NFV); Management and Orchestration", Available:
https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf,
December 2014.

[i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-08 (work in progress), July 2019.

[ITU-T.X.800]

"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[opsawg-firewalls]

Baker, F. and P. Hoffman, "On Firewalls in Internet Security", draft-ietf-opsawg-firewalls-01 (work in progress), October 2012.

[policy-translation]

Jeong, J., Yang, J., Chung, C., and J. Kim, "Security Policy Translation in Interface to Network Security Functions", draft-yang-i2nsf-security-policy-translation-04 (work in progress), July 2019.

[tls-esni]

Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-04 (work in progress), July 2019.

[VNF-ONBOARDING]

"VNF Onboarding", Available:
<https://wiki.opnfv.org/display/mano/VNF+Onboarding>,
November 2016.

Appendix A. Changes from draft-ietf-i2nsf-applicability-17

The following changes have been made from draft-ietf-i2nsf-applicability-17:

- o In Section 4, a high-level security policy XML file in Figure 2 and the corresponding low-level security policy XML file Figure 3 are constructed using the Consumer-Facing Interface data model and the NSF-Facing data model, respectively.
- o For the applicability of I2NSF to the real world, Section 5 is added to support the Intent-based Security Services using I2NSF. This section explains the security policy translation based on an I2NSF User's intents on the required security services. Figure 4 shows the architecture and procedure of the I2NSF security policy translator.

Authors' Addresses

Jaehoon Paul Jeong
Department of Computer Science and Engineering
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>

Sangwon Hyun
Department of Computer Engineering
Myongji University
116 Myongji-ro, Cheoin-gu
Yongin 17058
Republic of Korea

Phone: +82 62 230 7473
EMail: shyun@chosun.ac.kr

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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Diego R. Lopez
Telefonica I+D
Jose Manuel Lara, 9
Seville 41013
Spain

Phone: +34 682 051 091
EMail: diego.r.lopez@telefonica.com

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

S. Hares
Huawei
J. Jeong
J. Kim
Sungkyunkwan University
R. Moskowitz
HTT Consulting
Q. Lin
Huawei
July 02, 2018

I2NSF Capability YANG Data Model
draft-ietf-i2nsf-capability-data-model-01

Abstract

This document defines a YANG data model for capabilities that enable an I2NSF user to control various Network Security Functions (NSFs) in the framework for Interface to Network Security Functions (I2NSF).

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. Terminology	4
3.1. Tree Diagrams	4
4. Overview	4
5. The Structure and Objective of NSF Capabilities	6
5.1. Generic Network Security Function Identification	6
5.2. Event Capabilities	6
5.3. Condition Capabilities	7
5.4. Action Capabilities	7
5.5. Resolution Strategy Capabilities	7
5.6. Default Action Capabilities	7
5.7. RPC for Acquiring Appropriate Network Security Function	8
6. Data Model Structure	8
6.1. Network Security Function Identification	8
6.2. Capabilities of Generic Network Security Function	9
6.2.1. Event Capabilities	9
6.2.2. Condition Capabilities	11
6.2.3. Action Capabilities	14
6.2.4. Resolution Strategy Capabilities	15
6.2.5. Content Security Capabilities	15
6.2.6. Attack Mitigation Capabilities	16
6.2.7. RPC for Acquiring Appropriate Network Security Function	17
7. YANG Modules	18
7.1. I2NSF Capability YANG Data Module	18
8. IANA Considerations	52
9. Security Considerations	52
10. Acknowledgments	52
11. Contributors	53
12. References	53
12.1. Normative References	53
12.2. Informative References	53
Appendix A. Example: Extended VoIP-VoLTE Security Function Capabilities Module	55
Appendix B. Example: Configuration XML of Capability Module	56
B.1. Example: Configuration XML of Generic Network Security Function Capabilities	56
B.2. Example: Configuration XML of Extended VoIP/VoLTE Security Function Capabilities Module	58
Appendix C. Changes from draft-ietf-i2nsf-capability-data-	

model-01	59
Authors' Addresses	60

1. Introduction

As the industry becomes more sophisticated and network devices (e.g., Internet of Things, Self-driving vehicles, and VoIP/VoLTE smartphones), service providers have a lot of problems mentioned in [RFC8192]. To resolve these problems, [i2nsf-nsf-cap-im] specifies the information model of the capabilities of Network Security Functions (NSFs).

This document provides a data model using YANG [RFC6020][RFC7950] that defines the capabilities of NSFs to express capabilities of those security devices. This YANG data model is based on the information model for I2NSF NSF capabilities [i2nsf-nsf-cap-im]. The security devices can register their own capabilities into Network Operator Management (Mgmt) System (i.e., Security Controller) with this YANG data model through the registration interface [RFC8329]. After the capabilities of the NSFs are registered, this YANG data model can be used by the I2NSF user or Service Function Forwarder (SFF) [i2nsf-sfc] to acquire appropriate NSFs that can be controlled by the Network Operator Mgmt System.

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules. The "ietf-i2nsf-capability" YANG module defined in this document provides the following features:

- o Configuration of identification for generic network security function policy
- o Configuration of event capabilities for generic network security function policy
- o Configuration of condition capabilities for generic network security function policy
- o Configuration of action capabilities for generic network security function policy
- o Configuration of strategy capabilities for generic network security function policy
- o Configuration of default action capabilities for generic network security function policy
- o RPC for acquiring appropriate network security function according to type of NSF and/or target devices.

2. Requirements Language

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

3. Terminology

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

- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol.
- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] is as follows:

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

4. Overview

This section explains overview how the YANG data model can be used by I2NSF User, Developer's Mgmt System, and SFF. Figure 1 shows capabilities of NSFs in I2NSF Framework. As shown in this figure,

Developer's Mgmt System can register NSFs with capabilities that the device can support. To register NSFs in this way, the Developer's Mgmt System utilizes this standardized capabilities YANG data model through registration interface. Through this registration of capabilities, the a lot of problems [RFC8192] can be resolved. The following shows use cases.

Note [i2nsf-nsf-yang] is used to configure rules of NSFs in I2NSF Framework.

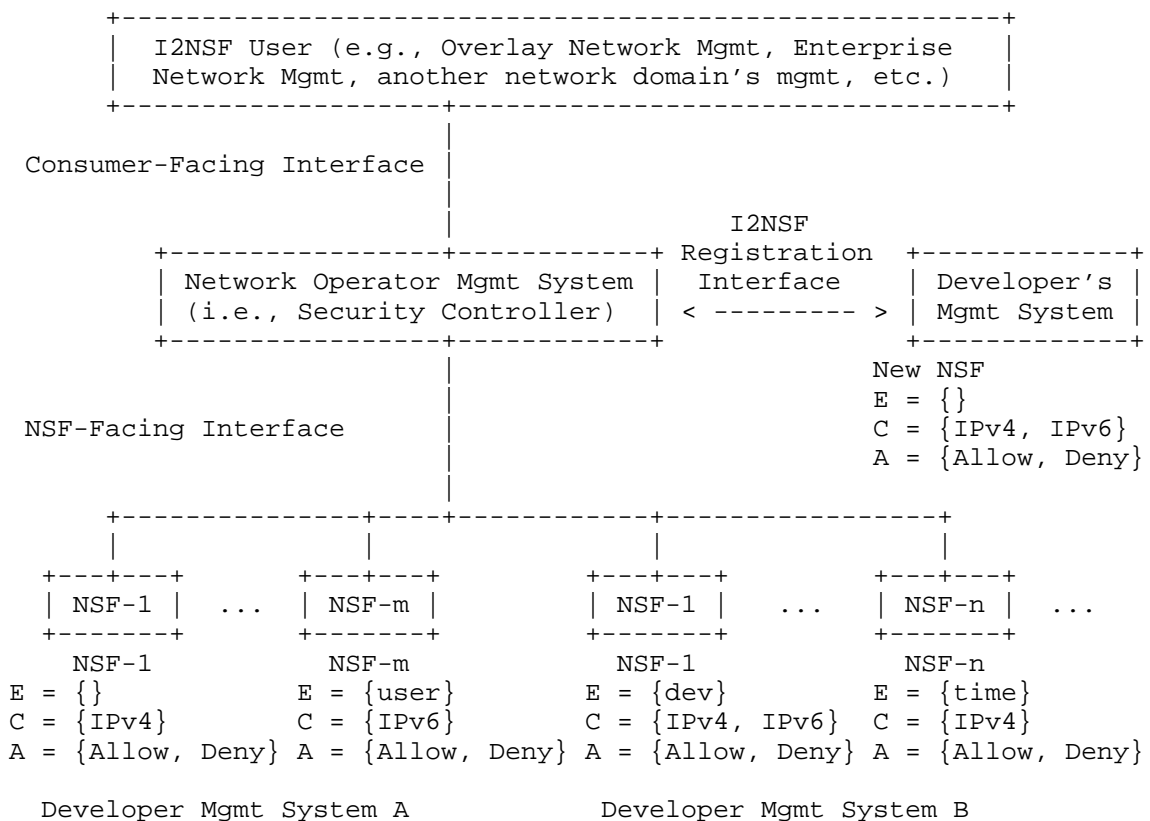


Figure 1: Capabilities of NSFs in I2NSF Framework

- o If I2NSF User wants to apply rules about blocking malicious users, it is a tremendous burden to apply all of these rules to NSFs one by one. This problem can be resolved by standardizing the capabilities of NSFs. If I2NSF User wants to block malicious users with IPv6, I2NSF User sends the rules about blocking the users to Network Operator Mgmt System. When the Network Operator Mgmt System receives the rules, it sends that rules to appropriate

NSFs (i.e., NSF-m in Developer Mgmt System A and NSF-1 in Developer Mgmt System B) which can support the capabilities (i.e., IPv6). Therefore, I2NSF User need not consider NSFs where to apply the rules.

- o If NSFs find the malicious packets, it is a tremendous burden for I2NSF User to apply the rule about blocking the malicious packets to NSFs one by one. This problem can be resolved by standardizing the capabilities of NSFs. If NSFs find the malicious packets with IPv4, they can ask the Network Operator Mgmt System to alter specific rules and/or configurations. When the Network Operator Mgmt System receives the rules for malicious packets, it inspects whether the rules are reasonable and sends the rules to appropriate NSFs (i.e., NSF-1 in Developer Mgmt System A and NSF-1 and NSF-n in Developer Mgmt System B) which can support the capabilities (i.e., IPv4). Therefore, the new rules can be applied to appropriate NSFs without control of I2NSF User.
- o If NSFs of Service Function Chaining (SFC) [i2nsf-sfc] fail, it is a tremendous burden for I2NSF User to reconfigure the policy of SFC immediately. This problem can be resolved by periodically acquiring information of appropriate NSFs of SFC. If SFF needs information of Web Application Firewall for SFC, it can ask the Network Operator Mgmt System to acquire the location information of appropriate Web Application Firewall. When the Network Operator Mgmt System receives requested information from SFF, it sends location information of Web Application Firewall to the SFF. Therefore, the policy about the NSFs of SFC can be periodically updated without control of I2NSF User.

5. The Structure and Objective of NSF Capabilities

5.1. Generic Network Security Function Identification

This shows a identification for generic network security functions. These objects are defined as location information and target device information.

5.2. Event Capabilities

This shows a event capabilities for generic network security functions policy. This is used to specify capabilities about any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of I2NSF Policy Rules, it is used to determine whether the Condition clause of the I2NSF Policy Rule can be evaluated or not. These object of event capabilities is defined as user security event capabilities, device security event capabilities, system

security event capabilities, and time security event capabilities. These object of event capabilities can be extended according to specific vendor event features.

5.3. Condition Capabilities

This shows a condition capabilities for generic network security functions policy. This is used to specify capabilities about a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to determine whether or not the set of Actions in that (imperative) I2NSF Policy Rule can be executed or not. These object of condition capabilities is defined as packet security condition capabilities, packet payload security condition capabilities, target security condition capabilities, user security condition capabilities, context condition capabilities, and generic context condition capabilities. These object of condition capabilities can be extended according to specific vendor condition features.

5.4. Action Capabilities

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

5.5. Resolution Strategy Capabilities

This shows a resolution strategy capabilities for generic network security functions policy. This can be used to specify capabilities how to resolve conflicts that occur between the actions of the same or different policy rules that are matched and contained in this particular NSF. These objects are defined as first-matching-rule capability and last-matching-rule capability. These objects can be extended according to specific vendor resolution strategy features.

5.6. Default Action Capabilities

This shows a default action policy for generic network security functions. This can be used to specify capabilities about a predefined action when no other alternative action was matched by the currently executing I2NSF Policy Rule.

5.7. RPC for Acquiring Appropriate Network Security Function

This shows a RPC for acquiring an appropriate network security function according to type of NSF and/or target devices. If the SFF [i2nsf-sfc] does not have the location information of network security functions that it should send in own cache table, this can be used to acquire the information. These objects are defined as input data (i.e., NSF type and target devices) and output data (i.e., location information of NSF).

6. Data Model Structure

This section shows an overview of a structure tree of capabilities for generic network security functions, as defined in the [i2nsf-nsf-cap-im].

6.1. Network Security Function Identification

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

```

module: ietf-i2nsf-capability
  +--rw nsf* [nsf-name]
    +--rw nsf-name                string
    +--rw nsf-type?              nsf-type
    +--rw nsf-address
      | +--rw (nsf-address-type)?
      |   +--:(ipv4-address)
      |   | +--rw ipv4-address    inet:ipv4-address
      |   +--:(ipv6-address)
      |   | +--rw ipv6-address    inet:ipv6-address
    +--rw target-device
      | +--rw pc?                boolean
      | +--rw mobile-phone?      boolean
      | +--rw voip-volte-phone?  boolean
      | +--rw tablet?           boolean
      | +--rw iot?              boolean
      | +--rw vehicle?          boolean
    +--rw generic-nsf-capabilities
      | +--rw net-sec-capabilities
      |   uses net-sec-caps
    +--rw complete-nsf-capabilities
      | +--rw con-sec-control-capabilities
      |   uses i2nsf-con-sec-control-caps
      +--rw attack-mitigation-capabilities
        uses i2nsf-attack-mitigation-control-caps

```

Figure 2: Data Model Structure for NSF-Identification

This draft also utilizes the concepts originated in Basile, Liroy, Pitscheider, and Zhao[2015] concerning conflict resolution, use of external data, and target device. The authors are grateful to Cataldo for pointing out this excellent work.

The nsf-type object can be used for configuration about type of a NSF. The types of NSF consists of Network Firewall, Web Application Firewall, Anti-Virus, IDS, IPS, and DDoS Mitigator. The nsf-address object can be used for configuration about location of a NSF. The target-device object can be used for configuration about target devices. We will add additional type of a NSF for more generic network security functions.

6.2. Capabilities of Generic Network Security Function

The data model for Generic NSF capabilities has the following structure:

```

+--rw generic-nsf-capabilities
  +--rw net-sec-capabilities
    uses i2nsf-net-sec-caps

```

Figure 3: Data Model Structure for Capabilities of Network Security Function

6.2.1. Event Capabilities

The data model for event capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
  +--rw net-sec-capabilities
    +--rw time
      +--rw time-zone
      |   +--rw time-zone-offset?  boolean
      +--rw time-interval
        +--rw absolute-time-inteval
          +--rw start-time?  boolean
          +--rw end-time?    boolean
        +--rw periodic-time-inteval
          +--rw day?        boolean
          +--rw month?      boolean
    +--rw event
      +--rw usr-event
        +--rw usr-sec-event-content?  boolean
        +--rw usr-sec-event-format
        |   +--rw unknown?  boolean
        |   +--rw guid?     boolean

```



```

| | | |--rw sys-sec-event-type
| | | | |--rw unknown? boolean
| | | | |--rw audit-log-written-to? boolean
| | | | |--rw audit-log-cleared? boolean
| | | | |--rw policy-created? boolean
| | | | |--rw policy-edited? boolean
| | | | |--rw policy-deleted? boolean
| | | | |--rw policy-executed? boolean
| | |--rw time-event
| | | |--rw time-sec-event-begin? boolean
| | | |--rw time-sec-event-end? boolean
| | | |--rw time-sec-event-time-zone? boolean
|--rw condition
| ...
+--rw action
| ...
+--rw resolution-strategy
| ...

```

Figure 4: Data Model Structure for Event Capabilities of Network Security Function

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

6.2.2. Condition Capabilities

The data model for condition capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
| +--rw net-sec-capabilities
| | +--rw time
| | | +--rw time-zone
| | | | +--rw time-zone-offset? boolean
| | | +--rw time-interval
| | | | +--rw absolute-time-interval
| | | | | +--rw start-time? boolean
| | | | | +--rw end-time? boolean
| | | | +--rw periodic-time-interval
| | | | | +--rw day? boolean
| | | | | +--rw month? boolean

```

```

+--rw event
|   ...
+--rw condition
|   +--rw packet-security-condition
|   |   +--rw packet-security-mac-condition
|   |   |   +--rw pkt-sec-cond-mac-dest?          boolean
|   |   |   +--rw pkt-sec-cond-mac-src?          boolean
|   |   |   +--rw pkt-sec-cond-mac-8021q?        boolean
|   |   |   +--rw pkt-sec-cond-mac-ether-type?    boolean
|   |   |   +--rw pkt-sec-cond-mac-tci?          string
|   |   +--rw packet-security-ipv4-condition
|   |   |   +--rw pkt-sec-cond-ipv4-header-length?    boolean
|   |   |   +--rw pkt-sec-cond-ipv4-tos?              boolean
|   |   |   +--rw pkt-sec-cond-ipv4-total-length?    boolean
|   |   |   +--rw pkt-sec-cond-ipv4-id?              boolean
|   |   |   +--rw pkt-sec-cond-ipv4-fragment?        boolean
|   |   |   +--rw pkt-sec-cond-ipv4-fragment-offset?  boolean
|   |   |   +--rw pkt-sec-cond-ipv4-ttl?              boolean
|   |   |   +--rw pkt-sec-cond-ipv4-protocol?        boolean
|   |   |   +--rw pkt-sec-cond-ipv4-src?              boolean
|   |   |   +--rw pkt-sec-cond-ipv4-dest?            boolean
|   |   |   +--rw pkt-sec-cond-ipv4-ipopts?          boolean
|   |   |   +--rw pkt-sec-cond-ipv4-sameip?          boolean
|   |   |   +--rw pkt-sec-cond-ipv4-geoip?           boolean
|   |   +--rw packet-security-ipv6-condition
|   |   |   +--rw pkt-sec-cond-ipv6-dscp?            boolean
|   |   |   +--rw pkt-sec-cond-ipv6-ecn?              boolean
|   |   |   +--rw pkt-sec-cond-ipv6-traffic-class?    boolean
|   |   |   +--rw pkt-sec-cond-ipv6-flow-label?       boolean
|   |   |   +--rw pkt-sec-cond-ipv6-payload-length?   boolean
|   |   |   +--rw pkt-sec-cond-ipv6-next-header?      boolean
|   |   |   +--rw pkt-sec-cond-ipv6-hop-limit?        boolean
|   |   |   +--rw pkt-sec-cond-ipv6-src?              boolean
|   |   |   +--rw pkt-sec-cond-ipv6-dest?            boolean
|   |   +--rw packet-security-tcp-condition
|   |   |   +--rw pkt-sec-cond-tcp-src-port?          boolean
|   |   |   +--rw pkt-sec-cond-tcp-dest-port?         boolean
|   |   |   +--rw pkt-sec-cond-tcp-seq-num?           boolean
|   |   |   +--rw pkt-sec-cond-tcp-ack-num?           boolean
|   |   |   +--rw pkt-sec-cond-tcp-window-size?       boolean
|   |   |   +--rw pkt-sec-cond-tcp-flags?             boolean
|   |   +--rw packet-security-udp-condition
|   |   |   +--rw pkt-sec-cond-udp-src-port?          boolean
|   |   |   +--rw pkt-sec-cond-udp-dest-port?         boolean
|   |   |   +--rw pkt-sec-cond-udp-length?            boolean
|   |   +--rw packet-security-icmp-condition
|   |   |   +--rw pkt-sec-cond-icmp-type?             boolean
|   |   |   +--rw pkt-sec-cond-icmp-code?             boolean

```

```

|         +--rw pkt-sec-cond-icmp-seg-num?    boolean
+--rw packet-payload-condition
|   +--rw pkt-payload-content?    boolean
+--rw acl-number?                  boolean
+--rw application-condition
|   +--rw application-object?    boolean
|   +--rw application-group?     boolean
|   +--rw application-label?     boolean
|   +--rw category
|       +--rw application-category?    boolean
+--rw target-condition
|   +--rw device-sec-context-cond?    boolean
+--rw users-condition
|   +--rw user
|       +--rw (user-name)?
|       |   +--:(tenant)
|       |   |   +--rw tenant?    boolean
|       |   +--:(vn-id)
|       |       +--rw vn-id?    boolean
|   +--rw group
|       +--rw (group-name)?
|       |   +--:(tenant)
|       |   |   +--rw tenant?    boolean
|       |   +--:(vn-id)
|       |       +--rw vn-id?    boolean
|       +--rw security-grup    boolean
+--rw url-category-condition
|   +--rw pre-defined-category?    boolean
|   +--rw user-defined-category?    boolean
+--rw context-condition
|   +--rw temp?    string
+--rw gen-context-condition
|   +--rw geographic-location
|       +--rw src-geographic-location?    boolean
|       +--rw dest-geographic-location?    boolean
+--rw action
|   ...
+--rw resolution-strategy
    ...

```

Figure 5: Data Model Structure for Condition Capabilities of Network Security Function

These objects are defined as capabilities of packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. These objects can be extended according to

specific vendor condition features. We will add additional condition objects for more generic network security functions.

6.2.3. Action Capabilities

The data model for action capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
  +--rw net-sec-capabilities
    +--rw time
      +--rw time-zone
      |   +--rw time-zone-offset?  boolean
      +--rw time-interval
      |   +--rw absolute-time-interval
      |   |   +--rw start-time?  boolean
      |   |   +--rw end-time?    boolean
      |   +--rw periodic-time-interval
      |       +--rw day?        boolean
      |       +--rw month?     boolean
    +--rw event
    |   ...
    +--rw condition
    |   ...
    +--rw action
    |   +--rw rule-log?          boolean
    |   +--rw session-log?      boolean
    |   +--rw ingress-action
    |   |   +--rw ingress-action-type
    |   |   |   +--rw pass?      boolean
    |   |   |   +--rw drop?      boolean
    |   |   |   +--rw reject?    boolean
    |   |   |   +--rw alert?     boolean
    |   |   |   +--rw mirror?    boolean
    |   +--rw egress-action
    |   |   +--rw egress-action-type
    |   |   |   +--rw invoke-signaling?  boolean
    |   |   |   +--rw tunnel-encapsulation?  boolean
    |   |   |   +--rw forwarding?         boolean
    |   |   |   +--rw redirection?        boolean
    +--rw resolution-strategy
    ...
  
```

Figure 6: Data Model Structure for Action Capabilities of Network Security Function

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

6.2.4. Resolution Strategy Capabilities

The data model for resolution strategy capabilities has the following structure:

```

+---rw i2nsf-net-sec-caps
  +---rw net-sec-capabilities
    +---rw time
      |   +---rw time-zone
      |   |   +---rw time-zone-offset?   boolean
      |   +---rw time-interval
      |   |   +---rw absolute-time-interval
      |   |   |   +---rw start-time?   boolean
      |   |   |   +---rw end-time?     boolean
      |   |   +---rw periodic-time-interval
      |   |   |   +---rw day?         boolean
      |   |   |   +---rw month?      boolean
    +---rw event
      |   ...
    +---rw condition
      |   ...
    +---rw action
      |   ...
    +---rw resolution-strategy
      +---rw first-matching-rule?   boolean
      +---rw last-matching-rule?    boolean

```

Figure 7: Data Model Structure for Resolution Strategy Capabilities of Network Security Function

These objects are defined capabilities as first-matching-rule and last-matching-rule. These objects can be extended according to specific vendor resolution strategy features. We will add additional resolution strategy objects for more generic network security functions.

6.2.5. Content Security Capabilities

The data model for content security capabilities has the following structure:

```
+--rw complete-nsf-capabilities
  +--rw con-sec-control-capabilities
    |   +--rw anti-virus?          boolean
    |   +--rw ips?                boolean
    |   +--rw ids?                boolean
    |   +--rw url-filter?         boolean
    |   +--rw data-filter?        boolean
    |   +--rw mail-filter?        boolean
    |   +--rw sql-filter?         boolean
    |   +--rw file-blocking?      boolean
    |   +--rw file-isolate?       boolean
    |   +--rw pkt-capture?        boolean
    |   +--rw application-behavior? boolean
    |   +--rw voip-volte?         boolean
  +--rw attack-mitigation-capabilities
    ...
```

Figure 8: Data Model Structure for Content Security Capabilities of Network Security Function

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

6.2.6. Attack Mitigation Capabilities

The data model for attack mitigation capabilities has the following structure:

```

+--rw complete-nsf-capabilities
  ...
+--rw attack-mitigation-capabilities
  +--rw (attack-mitigation-control-type)?
    +--:(ddos-attack)
      +--rw (ddos-attack-type)?
        +--:(network-layer-ddos-attack)
          +--rw network-layer-ddos-attack-types
            +--rw syn-flood-attack?          boolean
            +--rw udp-flood-attack?          boolean
            +--rw icmp-flood-attack?          boolean
            +--rw ip-fragment-flood-attack?  boolean
            +--rw ipv6-related-attack?        boolean
          +--:(app-layer-ddos-attack)
            +--rw app-layer-ddos-attack-types
              +--rw http-flood-attack?        boolean
              +--rw https-flood-attack?        boolean
              +--rw dns-flood-attack?          boolean
              +--rw dns-amp-flood-attack?      boolean
              +--rw ssl-flood-attack?          boolean
        +--:(single-packet-attack)
          +--rw (single-packet-attack-type)?
            +--:(scan-and-sniff-attack)
              +--rw ip-sweep-attack?          boolean
              +--rw port-scanning-attack?      boolean
            +--:(malformed-packet-attack)
              +--rw ping-of-death-attack?      boolean
              +--rw teardrop-attack?            boolean
            +--:(special-packet-attack)
              +--rw oversized-icmp-attack?      boolean
              +--rw tracert-attack?              boolean

```

Figure 9: Data Model Structure for Attack Mitigation Capabilities of Network Security Function

Attack mitigation is composed of a number of GNSFs; each one protects against a specific type of network attack. Attack Mitigation security is a type of GNSF, which summarizes a well-defined set of security Capabilities.

6.2.7. RPC for Acquiring Appropriate Network Security Function

The data model for RPC for Acquiring Appropriate Network Security Function has the following structure:

```

rpcs:
  +---x call-appropriate-nsf
    +---w input
      +---w nsf-type          nsf-type
      +---w target-device
      +---w pc?                boolean
      +---w mobile-phone?      boolean
      +---w voip-volte-phone?  boolean
      +---w tablet?            boolean
      +---w iot?                boolean
      +---w vehicle?           boolean
    +--ro output
      +--ro nsf-address
        +--ro (nsf-address-type)?
          +--:(ipv4-address)
            | +--ro ipv4-address    inet:ipv4-address
          +--:(ipv6-address)
            | +--ro ipv6-address    inet:ipv6-address

```

Figure 10: RPC for Acquiring Appropriate Network Security Function

This shows a RPC for acquiring an appropriate network security function according to type of NSF and/or target devices. If the SFF [i2nsf-sfc] does not have the location information of network security functions that it should send in own cache table, this can be used to acquire the information. These objects are defined as input data (i.e., NSF type and target devices) and output data (i.e., location information of NSF).

7. YANG Modules

7.1. I2NSF Capability YANG Data Module

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

<CODE BEGINS> file "ietf-i2nsf-capability@2018-07-02.yang"

```

module ietf-i2nsf-capability {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability";
  prefix
    i2nsf-capability;

  import ietf-inet-types{
    prefix inet;
  }

```

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

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

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

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

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

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

  Editor: Jinyong Tim Kim
  <mailto:timkim@skku.edu>";

description
  "This module describes a capability model
  for I2NSF devices.";

revision "2018-07-02" {
  description "The fifth revision";
  reference
    "draft-ietf-i2nsf-capability-00";
}

grouping i2nsf-nsf-location {
  description
    "This provides a location for capabilities.";
  container nsf-address {
    description
      "This is location information for capabilities.";
    choice nsf-address-type {
      description
        "nsf address type: ipv4 and ipv4";
      case ipv4-address {
        description
          "ipv4 case";
        leaf ipv4-address {
```

```
        type inet:ipv4-address;
        mandatory true;
        description
            "nsf address type is ipv4";
    }
}
case ipv6-address {
    description
        "ipv6 case";
    leaf ipv6-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "nsf address type is ipv6";
    }
}
}
}
}

typedef nsf-type {
    type enumeration {
        enum network-firewall {
            description
                "If type of a NSF is Network Firewall.";
        }

        enum web-app-firewall {
            description
                "If type of a NSF is Web Application
                Firewall.";
        }

        enum anti-virus {
            description
                "If type of a NSF is Anti-Virus";
        }

        enum ids {
            description
                "If type of a NSF is IDS.";
        }

        enum ips {
            description
                "If type of a NSF is IPS.";
        }
    }
}
```

```
        enum ddos-mitigator {
            description
                "If type of a NSF is DDoS Mitigator.";
        }
    }
    description
        "This is used for type of NSF.";
}

grouping i2nsf-it-resources {
    description
        "This provides a link between capabilities
        and IT resources. This has a list of IT resources
        by name.";
    container target-device {
        description
            "it-resources";

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

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

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

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

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



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

grouping capabilities-information {
    description
        "This includes information of capabilities.";

    leaf nsf-type {
        type nsf-type;
        description
            "This is type of NSF.";
    }
    uses i2nsf-nsf-location;
    uses i2nsf-it-resources;
}

grouping i2nsf-net-sec-caps {
    description
        "i2nsf-net-sec-caps";
    container net-sec-capabilities {
        description
            "net-sec-capabilities";

        container time {
            description
                "This is capabilities for time";
            container time-zone {
                description
                    "This can be used to apply rules
                    according to time zone";

                leaf time-zone-offset {
                    type boolean;
                    description
                        "This is offset for UTC time zone";
                }
            }

            container time-interval {
                description
                    "This can be used to apply rules
```

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

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

    container usr-event {
        description "TBD";
    }
}
```

```
leaf usr-sec-event-content {
    type boolean;
    description
        "This is a mandatory string that contains the content
        of the UserSecurityEvent. The format of the content
        is specified in the usrSecEventFormat class
        attribute, and the type of event is defined in the
        usrSecEventType class attribute. An example of the
        usrSecEventContent attribute is a string hrAdmin,
        with the usrSecEventFormat set to 1 (GUID) and the
        usrSecEventType attribute set to 5 (new logon).";
}

container usr-sec-event-format {
    description
        "This is a mandatory uint 8 enumerated integer, which
        is used to specify the data type of the
        usrSecEventContent attribute. The content is
        specified in the usrSecEventContent class attribute,
        and the type of event is defined in the
        usrSecEventType class attribute. An example of the
        usrSecEventContent attribute is string hrAdmin,
        with the usrSecEventFormat attribute set to 1 (GUID)
        and the usrSecEventType attribute set to 5
        (new logon).";
    leaf unknown {
        type boolean;
        description
            "If SecEventFormat is unknown";
    }
    leaf guid {
        type boolean;
        description
            "If SecEventFormat is GUID
            (Generic Unique Identifier)";
    }
    leaf uuid {
        type boolean;
        description
            "If SecEventFormat is UUID
            (Universal Unique Identifier)";
    }
    leaf uri {
        type boolean;
        description
            "If SecEventFormat is URI
            (Uniform Resource Identifier)";
    }
}
```

```
    leaf fqdn {
        type boolean;
        description
            "If SecEventFormat is FQDN
            (Fully Qualified Domain Name)";
    }
    leaf fqpn {
        type boolean;
        description
            "If SecEventFormat is FQPN
            (Fully Qualified Path Name)";
    }
}

container usr-sec-event-type {
    leaf unknown {
        type boolean;
        description
            "If usrSecEventType is unknown";
    }
    leaf user-created {
        type boolean;
        description
            "If usrSecEventType is new user
            created";
    }
    leaf user-grp-created {
        type boolean;
        description
            "If usrSecEventType is new user
            group created";
    }
    leaf user-deleted {
        type boolean;
        description
            "If usrSecEventType is user
            deleted";
    }
    leaf user-grp-deleted {
        type boolean;
        description
            "If usrSecEventType is user
            group deleted";
    }
    leaf user-logon {
        type boolean;
        description
            "If usrSecEventType is user
```

```
        logon";
    }
    leaf user-logout {
        type boolean;
        description
            "If usrSecEventType is user
            logout";
    }
    leaf user-access-request {
        type boolean;
        description
            "If usrSecEventType is user
            access request";
    }
    leaf user-access-granted {
        type boolean;
        description
            "If usrSecEventType is user
            granted";
    }
    leaf user-access-violation {
        type boolean;
        description
            "If usrSecEventType is user
            violation";
    }
    description
        "This is a mandatory uint 8 enumerated integer, which
        is used to specify the type of event that involves
        this user. The content and format are specified in
        the usrSecEventContent and usrSecEventFormat class
        attributes, respectively. An example of the
        usrSecEventContent attribute is string hrAdmin,
        with the usrSecEventFormat attribute set to 1 (GUID)
        and the usrSecEventType attribute set to 5
        (new logon).";
}

}
container dev-event {
    description "TBD";

    leaf dev-sec-event-content {
        type boolean;
        mandatory true;
        description
            "This is a mandatory string that contains the content
```

```
    of the DeviceSecurityEvent. The format of the
    content is specified in the devSecEventFormat class
    attribute, and the type of event is defined in the
    devSecEventType class attribute. An example of the
    devSecEventContent attribute is alarm, with the
    devSecEventFormat attribute set to 1 (GUID), the
    devSecEventType attribute set to 5 (new logon).";
}
```

```
container dev-sec-event-format {
  description
    "This is a mandatory uint 8 enumerated integer,
    which is used to specify the data type of the
    devSecEventContent attribute.";

  leaf unknown {
    type boolean;
    description
      "If SecEventFormat is unknown";
  }
  leaf guid {
    type boolean;
    description
      "If SecEventFormat is GUID
      (Generic Unique Identifier)";
  }
  leaf uuid {
    type boolean;
    description
      "If SecEventFormat is UUID
      (Universal Unique Identifier)";
  }
  leaf uri {
    type boolean;
    description
      "If SecEventFormat is URI
      (Uniform Resource Identifier)";
  }
  leaf fqdn {
    type boolean;
    description
      "If SecEventFormat is FQDN
      (Fully Qualified Domain Name)";
  }
  leaf fqpn {
    type boolean;
    description
      "If SecEventFormat is FQPN
```

```
        (Fully Qualified Path Name)";
    }
}

container dev-sec-event-type {
  description
    "This is a mandatory uint 8 enumerated integer,
    which is used to specify the type of event
    that was generated by this device.";

  leaf unknown {
    type boolean;
    description
      "If devSecEventType is unknown";
  }
  leaf comm-alarm {
    type boolean;
    description
      "If devSecEventType is communications
      alarm";
  }
  leaf quality-of-service-alarm {
    type boolean;
    description
      "If devSecEventType is quality of service
      alarm";
  }
  leaf process-err-alarm {
    type boolean;
    description
      "If devSecEventType is processing error
      alarm";
  }
  leaf equipment-err-alarm {
    type boolean;
    description
      "If devSecEventType is equipment error
      alarm";
  }
  leaf environmental-err-alarm {
    type boolean;
    description
      "If devSecEventType is environmental error
      alarm";
  }
}
```

```
    container dev-sec-event-type-severity {
      description
        "This is a mandatory uint 8 enumerated integer,
        which is used to specify the perceived
        severity of the event generated by this
        Device.";

      leaf unknown {
        type boolean;
        description
          "If devSecEventType is unknown";
      }
      leaf cleared {
        type boolean;
        description
          "If devSecEventTypeSeverity is cleared";
      }
      leaf indeterminate {
        type boolean;
        description
          "If devSecEventTypeSeverity is
          indeterminate";
      }
      leaf critical {
        type boolean;
        description
          "If devSecEventTypeSeverity is critical";
      }
      leaf major {
        type boolean;
        description
          "If devSecEventTypeSeverity is major";
      }
      leaf minor {
        type boolean;
        description
          "If devSecEventTypeSeverity is minor";
      }
      leaf warning {
        type boolean;
        description
          "If devSecEventTypeSeverity is warning";
      }
    }
  }
  container sys-event {
    description "TBD";
```



```
leaf sys-sec-event-content {
    type boolean;
    description
        "This is a mandatory string that contains a content
        of the SystemSecurityEvent. The format of a content
        is specified in a sysSecEventFormat class attribute,
        and the type of event is defined in the
        sysSecEventType class attribute. An example of the
        sysSecEventContent attribute is string sysadmin3,
        with the sysSecEventFormat attribute set to 1(GUID),
        and the sysSecEventType attribute set to 2
        (audit log cleared).";
}

container sys-sec-event-format {
    description
        "This is a mandatory uint 8 enumerated integer, which
        is used to specify the data type of the
        sysSecEventContent attribute.";

    leaf unknown {
        type boolean;
        description
            "If SecEventFormat is unknown";
    }
    leaf guid {
        type boolean;
        description
            "If SecEventFormat is GUID
            (Generic Unique Identifier)";
    }
    leaf uuid {
        type boolean;
        description
            "If SecEventFormat is UUID
            (Universal Unique Identifier)";
    }
    leaf uri {
        type boolean;
        description
            "If SecEventFormat is URI
            (Uniform Resource Identifier)";
    }
    leaf fqdn {
        type boolean;
        description
            "If SecEventFormat is FQDN
            (Fully Qualified Domain Name)";
    }
}
```

```
    }
    leaf fqpn {
        type boolean;
        description
            "If SecEventFormat is FQPN
            (Fully Qualified Path Name)";
    }
}

container sys-sec-event-type {
    description
        "This is a mandatory uint 8 enumerated integer, which
        is used to specify the type of event that involves
        this device.";

    leaf unknown {
        type boolean;
        description
            "If sysSecEventType is unknown";
    }
    leaf audit-log-written-to {
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that audit log is written to";
    }
    leaf audit-log-cleared {
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that audit log is cleared";
    }
    leaf policy-created {
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that policy is created";
    }
    leaf policy-edited{
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that policy is edited";
    }
    leaf policy-deleted{
        type boolean;
        description
            "If sysSecEventTypeSeverity
```

```
        is that policy is deleted";
    }
    leaf policy-executed{
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that policy is executed";
    }
}
}
container time-event {
    description "TBD";

    leaf time-sec-event-begin {
        type boolean;
        description
            "This is a mandatory DateTime attribute, and
            represents the beginning of a time period.
            It has a value that has a date and/or a time
            component (as in the Java or Python libraries).";
    }

    leaf time-sec-event-end {
        type boolean;
        description
            "This is a mandatory DateTime attribute, and
            represents the end of a time period. It has
            a value that has a date and/or a time component
            (as in the Java or Python libraries). If this is
            a single event occurrence, and not a time period
            when the event can occur, then the
            timeSecEventPeriodEnd attribute may be ignored.";
    }

    leaf time-sec-event-time-zone {
        type boolean;
        description
            "This is a mandatory string attribute, and defines a
            time zone that this event occurred in using the
            format specified in ISO8601.";
    }
}

}

container condition {
    description
        " This is abstract. A condition is defined as a set
```

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

```
container packet-security-condition {  
  description "TBD";
```

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

```
    leaf pkt-sec-cond-mac-dest {  
      type boolean;  
      description  
        "The MAC destination address (6 octets long).";  
    }
```

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

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

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

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

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

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

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

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

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

    leaf pkt-sec-cond-ipv4-fragment {
```

```
type boolean;
description
  "IP fragmentation is an Internet Protocol (IP)
  process that breaks datagrams into smaller pieces
  (fragments), so that packets may be formed that
  can pass through a link with a smaller maximum
  transmission unit (MTU) than the original
  datagram size.";
}

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

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

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

leaf pkt-sec-cond-ipv4-src {
  type boolean;
  description
    "Defines the IPv4 Source Address.";
}

leaf pkt-sec-cond-ipv4-dest {
  type boolean;
  description
    "Defines the IPv4 Destination Address.";
}

leaf pkt-sec-cond-ipv4-iptopts {
  type boolean;
```

```
    description
      "With the ipopts keyword you can check if
       a specific ip option is set. Ipopts has
       to be used at the beginning of a rule.";
  }

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

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

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

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

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

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

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

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

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

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

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

leaf pkt-sec-cond-ipv6-dest {
    type boolean;
```



```
        description
          "The IPv6 address of the destination node(s).";
      }
}

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

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

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

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

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

  leaf pkt-sec-cond-tcp-window-size {
    type boolean;
    description
      "The size of the receive window, which specifies
      the number of windows size units (by default,bytes)
      (beyond the segment identified by the sequence
```

```
        number in the acknowledgment field) that the sender
        of this segment is currently willing to receive.";
    }

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

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

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

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

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

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

    leaf pkt-sec-cond-icmp-type {
        type boolean;
    }
}
```

```
        description
            "ICMP type, see Control messages.";
    }

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

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

container packet-payload-condition {
    description "TBD";
    leaf pkt-payload-content {
        type boolean;
        description
            "The content keyword is very important in
            signatures. Between the quotation marks you
            can write on what you would like the
            signature to match.";
    }
}

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

container application-condition {
    description
        "TBD";

    leaf application-object {
        type boolean;
        description
            "This is application object.";
    }
    leaf application-group {
        type boolean;
        description
            "This is application group.";
    }
}
```

```
    }
    leaf application-label {
        type boolean;
        description
            "This is application label.";
    }
    container category {
        description
            "TBD";
        leaf application-category {
            type boolean;
            description
                "TBD";
        }
    }
}

container target-condition {
    description "TBD";

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

container users-condition {
    description "TBD";

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

        choice user-name {
```

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

case tenant {
  description
    "Tenant information.";

  leaf tenant {
    type boolean;
    description
      "User's tenant information.";
  }
}

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

  leaf vn-id {
    type boolean;
    description
      "User's VN-ID information.";
  }
}
}

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

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

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

```
        leaf tenant {
            type boolean;
            description
                "User's tenant information.";
        }
    }

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

        leaf vn-id {
            type boolean;
            description
                "User's VN-ID information.";
        }
    }
}

leaf security-grup {
    type boolean;
    mandatory true;
    description
        "security-grup.";
}

}

container url-category-condition {
    description
        "TBD";
    leaf pre-defined-category {
        type boolean;
        description
            "This is pre-defined-category.";
    }
    leaf user-defined-category {
        type boolean;
        description
            "This user-defined-category.";
    }
}

container context-condition {
    description "TBD";
    leaf temp {
        type string;
        description
            "This is temp for context condition.";
    }
}
```

```
    }  
  }  
  container gen-context-condition {  
    description "TBD";  
  
    container geographic-location {  
      description  
        "The location where network traffic is associated  
        with. The region can be the geographic location  
        such as country, province, and city,  
        as well as the logical network location such as  
        IP address, network section, and network domain.";  
  
      leaf src-geographic-location {  
        type boolean;  
        description  
          "This is mapped to ip address. We can acquire  
          source region through ip address stored the  
          database.";  
      }  
      leaf dest-geographic-location {  
        type boolean;  
        description  
          "This is mapped to ip address. We can acquire  
          destination region through ip address stored  
          the database.";  
      }  
    }  
  }  
}  
container action {  
  description  
    "An action is used to control and monitor aspects of  
    flow-based NSFs when the event and condition clauses  
    are satisfied. NSFs provide security functions by  
    executing various Actions. Examples of I2NSF Actions  
    include providing intrusion detection and/or protection,  
    web and flow filtering, and deep packet inspection  
    for packets and flows."  
  
  leaf rule-log {  
    type boolean;  
    description  
      "rule-log";  
  }  
  leaf session-log {  
    type boolean;  
    description
```

```
    "session-log";
  }

  container ingress-action {
    description "TBD";

    container ingress-action-type {
      description
        "Ingress action type: permit, deny, and mirror.";
      leaf pass {
        type boolean;
        description
          "If ingress action is pass";
      }
      leaf drop {
        type boolean;
        description
          "If ingress action is drop";
      }
      leaf reject {
        type boolean;
        description
          "If ingress action is reject";
      }
      leaf alert {
        type boolean;
        description
          "If ingress action is alert";
      }
      leaf mirror {
        type boolean;
        description
          "If ingress action is mirror";
      }
    }
  }

  container egress-action {
    description "TBD";

    container egress-action-type {
      description
        "Egress-action-type: invoke-signaling,
        tunnel-encapsulation, and forwarding.";
      leaf invoke-signaling {
        type boolean;
        description
          "If egress action is invoke signaling";
      }
    }
  }
}
```



```
        leaf tunnel-encapsulation {
            type boolean;
            description
                "If egress action is tunnel encapsulation";
        }
        leaf forwarding {
            type boolean;
            description
                "If egress action is forwarding";
        }
        leaf redirection {
            type boolean;
            description
                "If egress action is redirection";
        }
    }
}

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

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

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

}

grouping i2nsf-con-sec-control-caps {
    description
        "i2nsf-con-sec-control-caps";

    container con-sec-control-capabilities {
        description
            "content-security-control-capabilities";
    }
}
```

```
    leaf anti-virus {
        type boolean;
        description
            "antivirus";
    }
    leaf ips {
        type boolean;
        description
            "ips";
    }

    leaf ids {
        type boolean;
        description
            "ids";
    }

    leaf url-filter {
        type boolean;
        description
            "url-filter";
    }
    leaf data-filter {
        type boolean;
        description
            "data-filter";
    }
    leaf mail-filter {
        type boolean;
        description
            "mail-filter";
    }
    leaf sql-filter {
        type boolean;
        description
            "sql-filter";
    }
    leaf file-blocking {
        type boolean;
        description
            "file-blocking";
    }
    leaf file-isolate {
        type boolean;
        description
            "file-isolate";
    }
    leaf pkt-capture {
```

```
        type boolean;
        description
            "pkt-capture";
    }
    leaf application-behavior {
        type boolean;
        description
            "application-behavior";
    }
    leaf voip-volte {
        type boolean;
        description
            "voip-volte";
    }
}

}

grouping i2nsf-attack-mitigation-control-caps {
    description
        "i2nsf-attack-mitigation-control-caps";

    container attack-mitigation-capabilities {
        description
            "attack-mitigation-capabilities";
        choice attack-mitigation-control-type {
            description
                "attack-mitigation-control-type";
            case ddos-attack {
                description
                    "ddos-attack";
                choice ddos-attack-type {
                    description
                        "ddos-attack-type";
                    case network-layer-ddos-attack {
                        description
                            "network-layer-ddos-attack";
                        container network-layer-ddos-attack-types {
                            description
                                "network-layer-ddos-attack-type";
                            leaf syn-flood-attack {
                                type boolean;
                                description
                                    "syn-flood-attack";
                            }
                            leaf udp-flood-attack {
                                type boolean;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
        description
            "udp-flood-attack";
    }
    leaf icmp-flood-attack {
        type boolean;
        description
            "icmp-flood-attack";
    }
    leaf ip-fragment-flood-attack {
        type boolean;
        description
            "ip-fragment-flood-attack";
    }
    leaf ipv6-related-attack {
        type boolean;
        description
            "ip-fragment-flood-attack";
    }
}
}
case app-layer-ddos-attack {
    description
        "app-layer-ddos-attack";
    container app-layer-ddos-attack-types {
        description
            "app-layer-ddos-attack-types";
        leaf http-flood-attack {
            type boolean;
            description
                "http-flood-attack";
        }
        leaf https-flood-attack {
            type boolean;
            description
                "https-flood-attack";
        }
        leaf dns-flood-attack {
            type boolean;
            description
                "dns-flood-attack";
        }
        leaf dns-amp-flood-attack {
            type boolean;
            description
                "dns-amp-flood-attack";
        }
        leaf ssl-flood-attack {
            type boolean;
        }
    }
}
```

```
        description
          "ssl-flood-attack";
      }
    }
  }
}

case single-packet-attack {
  description
    "single-packet-attack";
  choice single-packet-attack-type {
    description
      "single-packet-attack-type";
    case scan-and-sniff-attack {
      description
        "scan-and-sniff-attack";
      leaf ip-sweep-attack {
        type boolean;
        description
          "ip-sweep-attack";
      }
      leaf port-scanning-attack {
        type boolean;
        description
          "port-scanning-attack";
      }
    }
  }
  case malformed-packet-attack {
    description
      "malformed-packet-attack";
    leaf ping-of-death-attack {
      type boolean;
      description
        "ping-of-death-attack";
    }
    leaf teardrop-attack {
      type boolean;
      description
        "teardrop-attack";
    }
  }
}
case special-packet-attack {
  description
    "special-packet-attack";
  leaf oversized-icmp-attack {
    type boolean;
    description
```

```

        "oversized-icmp-attack";
    }
    leaf tracert-attack {
        type boolean;
        description
            "tracert-attack";
    }
}
}
}
}
}

list nsf {
    key "nsf-name";
    description
        "nsf-name";
    leaf nsf-name {
        type string;
        mandatory true;
        description
            "nsf-name";
    }
}

uses capabilities-information;

container generic-nsf-capabilities {
    description
        "generic-nsf-capabilities";
    uses i2nsf-net-sec-caps;
}

container complete-nsf-capabilities {
    description
        "generic-nsf-capabilities";
    uses i2nsf-con-sec-control-caps;
    uses i2nsf-attack-mitigation-control-caps;
}

}

rpc call-appropriate-nsf {
    description
        "We can acquire appropriate NSF that we want

```

If we give type of NSF that we want to use,
we acquire the location information of NSF";

```
input {  
  leaf nsf-type {  
    type nsf-type;  
    mandatory true;  
    description  
      "This is used to acquire NSF  
      This is mandatory";  
  }  
  uses i2nsf-it-resources;  
}  
output {  
  uses i2nsf-nsf-location;  
}  
}
```

<CODE ENDS>

Figure 11: YANG Data Module of I2NSF Capability

8. IANA Considerations

No IANA considerations exist for this document at this time. URL will be added.

9. Security Considerations

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

10. Acknowledgments

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

11. Contributors

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

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

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.
- [RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, July 2017.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

12.2. Informative References

- [i2nsf-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-01 (work in progress), April 2018.

`[i2nsf-nsf-yang]`

Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin,
"I2NSF Network Security Function-Facing Interface YANG
Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-00
(work in progress), March 2018.

`[i2nsf-sfc]`

Hyun, S., Jeong, J., Park, J., and S. Hares, "Service
Function Chaining-Enabled I2NSF Architecture", draft-hyun-
i2nsf-nsf-triggered-steering-05 (work in progress), March
2018.

`[i2nsf-terminology]`

Hares, S., Strassner, J., Lopez, D., Xia, L., and H.
Birkholz, "Interface to Network Security Functions (I2NSF)
Terminology", draft-ietf-i2nsf-terminology-05 (work in
progress), January 2018.

`[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-12
(work in progress), April 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. Example: Extended VoIP-VoLTE Security Function Capabilities Module

This section gives a simple example of how VoIP-VoLTE Security Function Capabilities module could be extended.

```
module
ex-voip-volte-cap {
  namespace "http://example.com/voip-volte-cap";
  prefix "voip-volte-cap";

  import ietf-i2nsf-capability {
    prefix capa;
  }

  augment "/capa:nsf/capa:generic-nsf-capabilities/"
    + "capa:net-sec-control-capabilities/"
    + "capa:condition/capa:condition-type" {
    case voice-condition {
      leaf sip-header-method {
        type boolean;
        description
          "SIP header method.";
      }

      leaf sip-header-uri {
        type boolean;
        description
          "SIP header URI.";
      }

      leaf sip-header-from {
        type boolean;
        description
          "SIP header From.";
      }

      leaf sip-header-to {
        type boolean;
        description
          "SIP header To.";
      }

      leaf sip-header-expire-time {
        type boolean;
        description
          "SIP header expire time.";
      }
    }
  }
```

```
    }  
    leaf sip-header-user-agent {  
      type boolean;  
      description  
        "SIP header user agent.";  
    }  
  }  
}
```

Figure 12: Example: Extended VoIP-VoLTE Security Function Capabilities Module

Appendix B. Example: Configuration XML of Capability Module

This section gives a xml examples for a configuration of Capability module according to a requirement.

B.1. Example: Configuration XML of Generic Network Security Function Capabilities

This section gives a xml example for generic network security function capability configuration according to a requirement.

Requirement: Register packet filter according to requirements.

1. The location of the NSF is 221.159.112.150.
2. The NSF can obtain the best effect if the packet was generated by PC or IoT.
3. The NSF can apply policies according to time.
4. The NSF should be able to block the source packets or destination packets with IPv4 address.
5. The NSF should be able to pass, reject, or alert packets.
6. Here is XML example for the generic network security function capability configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running />
    </target>
    <config>
      <nsf xmlns="urn:ietf:params:xml:ns:yang:" +
          "ietf-i2nsf-capability">
        <nsf-name>Huawei-Firewall</nsf-name>
        <nsf-address>
          <ipv4-address>221.159.112.150</ipv4-address>
        </nsf-address>
        <target-device>
          <pc>true</pc>
        </target-device>
        <target-device>
          <iot>true</iot>
        </target-device>
        <generic-nsf-capabilities>
          <net-sec-control-capabilities>
            <nsf-capabilities-name>ipv4-packet-filter</nsf-capabilities-name>
            <time-zone>
              <start-time>true</start-time>
              <end-time>true</end-time>
            </time-zone>
            <condition>
              <packet-security-ipv4-condition>
                <pkt-sec-cond-ipv4-src>true</pkt-sec-cond-ipv4-src>
                <pkt-sec-cond-ipv4-dest>true</pkt-sec-cond-ipv4-dest>
              </packet-security-ipv4-condition>
            </condition>
            <action>
              <ingress-action-type>
                <pass>true</pass>
                <reject>true</reject>
                <alert>true</alert>
              </ingress-action-type>
            </action>
          </net-sec-control-capabilities>
        </generic-nsf-capabilities>
      </nsf>
    </config>
  </edit-config>
</rpc>
```

Figure 13: Example: Configuration XML for Generic Network Security Function Capability

B.2. Example: Configuration XML of Extended VoIP/VoLTE Security Function Capabilities Module

This section gives a xml example for extended VoIP-VoLTE security function capabilities (See Figure 12) configuration according to a requirement.

Requirement: Register VoIP/VoLTe security function according to requirements.

1. The location of the NSF is 221.159.112.151.
2. The NSF can obtain the best effect if the packet was generated by VoIP-VoLTE phone.
3. The NSF should be able to block the malicious sip packets with user agent.
4. The NSF should be able to pass, reject, or alert packets.

Here is XML example for the VoIP-VoLTE security function capabilities configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
    <running />
  </target>
<config>
  <nsf xmlns="urn:ietf:params:xml:ns:yang:" +
    "ietf-i2nsf-capability">
    <nsf-name>Cisco-VoIP-VoLTE</nsf-name>
    <nsf-address>
      <ipv4-address>221.159.112.151</ipv4-address>
    </nsf-address>
    <generic-nsf-capabilities>
      <net-sec-control-capabilities>
        <nsc-capabilities-name>sip-packet-filter<nsc-capabilities-name>
        <condition>
          <sip-header-user-agent>true</sip-header-user-agent>
        </condition>
        <action>
          <ingress-action-type>
            <pass>true</pass>
            <reject>true</reject>
            <alert>true</alert>
          </ingress-action-type>
        </action>
      </net-sec-control-capabilities>
    </generic-nsf-capabilities>
  </nsf>
</config>
</edit-config>
</rpc>
```

Figure 14: Example: Configuration XML for Extended VoIP/VoLTE
Security Function Capabilities

Appendix C. Changes from draft-ietf-i2nsf-capability-data-model-01

The following changes are made from draft-ietf-i2nsf-capability-data-model-00:

1. We have clarified and simplified capabilities.
2. We added additional condition capabilities for application and url.
3. We replaced unnecessary leaf-list component to leaf component.

4. We replaced the list component to the container component for net-sec-capabilities.
5. We modified the choice-case structure into a container structure to allow for the selection of multiple catalogues for condition and action clauses.
6. We added complete-nsf-capabilities such as content capabilities and attack mitigation capabilities.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

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

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

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

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

Phone: +1-248-968-9809
EMail: rgm@htt-consult.com

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

EMail: linqiushi@huawei.com

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

J. Jeong
E. Kim
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei
July 2, 2018

I2NSF Consumer-Facing Interface YANG Data Model
draft-ietf-i2nsf-consumer-facing-interface-dm-01

Abstract

This document describes a YANG data model for the Consumer-Facing Interface between an Interface to Network Security Functions (I2NSF) User and Security Controller in an I2NSF system in a Network Functions Virtualization (NFV) environment. The data model is required for enabling different users of a given I2NSF system to define, manage, and monitor security policies for specific flows within an administrative domain.

Status of This Memo

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
4. Data Modeling for Security Policies for Consumer-Facing Interface	3
5. YANG Data Model for Security Policies for Consumer-Facing Interface	8
6. Security Considerations	37
7. Acknowledgments	37
8. Contributors	37
9. References	37
9.1. Normative References	37
9.2. Informative References	37
Appendix A. Changes from draft-ietf-i2nsf-consumer-facing-interface-dm-00	39
Appendix B. Use Case: Policy Instance Example for VoIP/VoLTE Security Services	39
Appendix C. Policy Instance YANG Example for VoIP/VoLTE Security Services	41
Appendix D. Example XML output for VoIP service	51
Authors' Addresses	52

1. Introduction

This document provides a YANG [RFC6020] data model that defines the required data for the Consumer-Facing Interface between an Interface to Network Security Functions (I2NSF) User and Security Controller in an I2NSF system [i2nsf-framework] in a Network Functions Virtualization (NFV) environment. The data model is required for enabling different users of a given I2NSF system to define, manage and monitor security policies for specific flows within an administrative domain. This document defines a YANG data model based on the information model of I2NSF Consumer-Facing Interface [client-facing-inf-im].

Data models are defined at a lower level of abstraction and provide many details. They provide details about the implementation of a protocol's specification, e.g., rules that explain how to map managed objects onto lower-level protocol constructs. Since conceptual models can be implemented in different ways, multiple data models can be derived by a single information model.

The efficient and flexible provisioning of network functions by NFV leads to a rapid advance in the network industry. As practical applications, network security functions (NSFs), such as firewall, intrusion detection system (IDS)/intrusion protection system (IPS), and attack mitigation, can also be provided as virtual network functions (VNF) in the NFV system. By the efficient virtual technology, these VNFs might be automatically provisioned and dynamically migrated based on real-time security requirements. This document presents a YANG data model to implement security functions based on NFV.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC3444].

3. Terminology

This document uses the terminology described in [i2nsf-terminology][client-facing-inf-im][client-facing-inf-req].

4. Data Modeling for Security Policies for Consumer-Facing Interface

The main objective of this data model is to fully transform the information model [client-facing-inf-im] into a YANG data model that can be used for delivering control and management messages via the Consumer-Facing Interface between an I2NSF User and Security Controller for the I2NSF User's high-level security policies.

The semantics of the data model must be aligned with the information model of the Consumer-Facing Interface. The transformation of the information model was performed so that this YANG data model can facilitate the efficient delivery of the control or management messages.

This data model is designed to support the I2NSF framework that can be extended according to the security needs. In other words, the model design is independent of the content and meaning of specific policies as well as the implementation approach. This document

suggests a VoIP/VoLTE security service as a use case for policy rule generation.

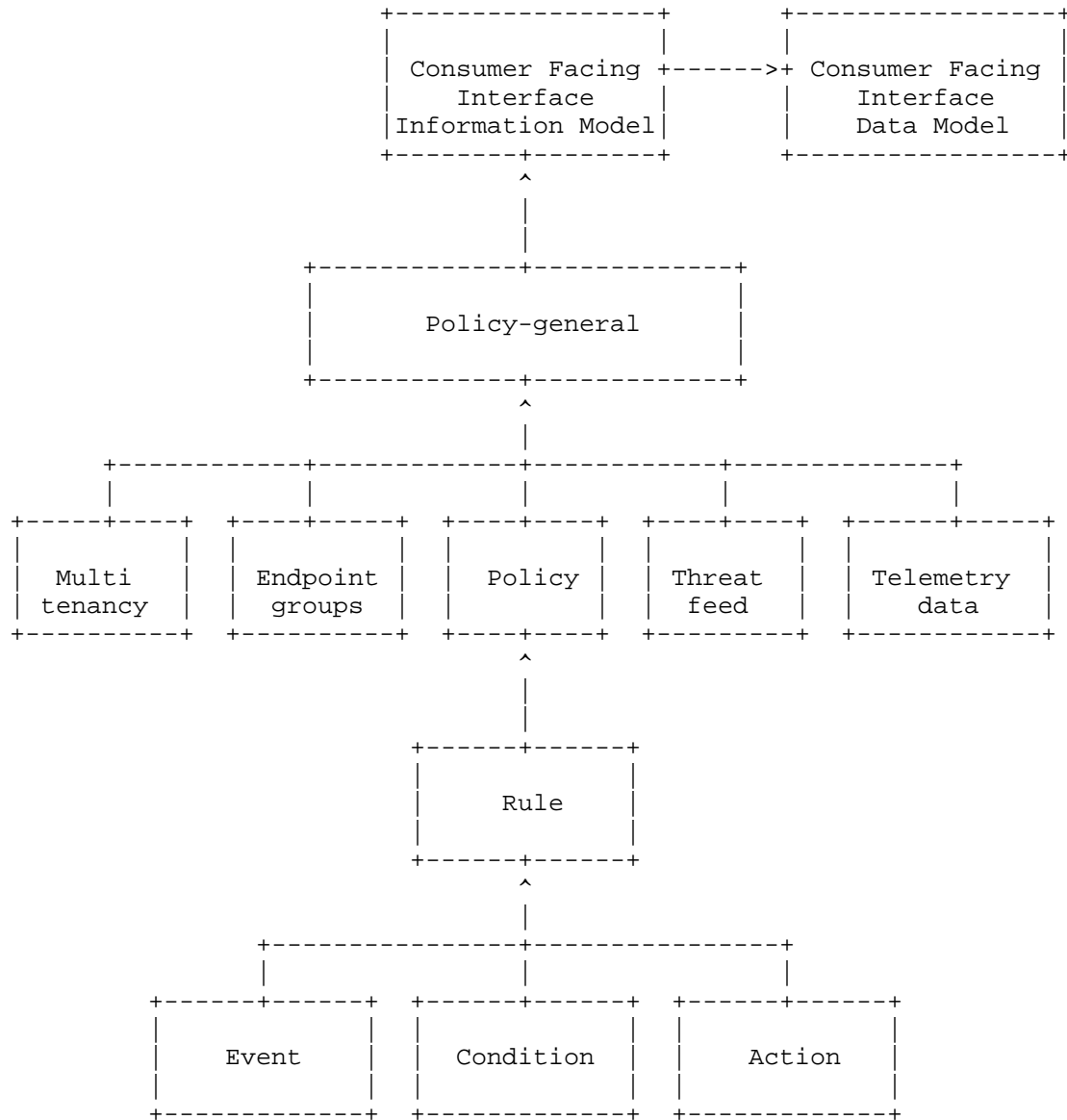


Figure 1: High-level-abstraction for Consumer Facing Interface

Multi-tenancy in this document enables multiple administrative domains in order to manage application resources. An Enterprise

organization may have multiple tenants or departments such as HR, finance, and legal. Thus, we need an object which defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. You can think of it as a way to assign policy users to a job function or a set of permissions within the organization. The policy-role object SHALL have Name, Date and access-profile to grant or deny permissions for the purpose of security policy management.

```

module: policy-general
  +--rw policy
    |   +--rw rule* [rule-id]
    |   |   +--rw rule-id                uint16
    |   |   +--rw name?                  string
    |   |   +--rw date?                  yang:date-and-time
    |   |   +--rw case?                  string
    |   |   +--rw event* [event-id]
    |   |   |   +--rw event-id            string
    |   |   |   +--rw name?              string
    |   |   |   +--rw date?              yang:date-and-time
    |   |   |   +--rw event-type?        string
    |   |   |   +--rw time-information?   string
    |   |   |   +--rw event-map-group?    -> /threat-feed/event-map-group
    |   |   |   |   /event-map-group-id
    |   |   |   +--rw enable?            boolean
    |   |   +--rw condition* [condition-id]
    |   |   |   +--rw condition-id        string
    |   |   |   +--rw source?            string
    |   |   |   +--rw destination?       string
    |   |   |   +--rw match?             boolean
    |   |   |   +--rw match-direction?   string
    |   |   |   +--rw exception?         string
    |   |   +--rw policy-action* [policy-action-id]
    |   |   |   +--rw policy-action-id    string
    |   |   |   +--rw name?              string
    |   |   |   +--rw date?              yang:date-and-time
    |   |   |   +--rw primary-action?    string
    |   |   |   +--rw secondary-action?  string
    |   |   |   +--rw owner?            string
    |   +--rw multi-tenancy
    |   |   +--rw policy-domain* [policy-domain-id]
    |   |   |   +--rw policy-domain-id    uint16
    |   |   |   +--rw name                string
    |   |   |   +--rw address?           string
    |   |   |   +--rw contact            string
    |   |   |   +--rw date                yang:date-and-time
    |   |   +--rw policy-tenant* [policy-tenant-id]
    |   |   |   +--rw policy-tenant-id    uint16

```

```

+--rw name string
+--rw date yang:date-and-time
+--rw domain? -> /multi-tenancy
/policy-domain
/policy-domain-id
+--rw authentication-method? -> /multi-tenancy
/policy-mgmt-auth-method
/policy-mgmt-auth-method-id
+--rw policy-role* [policy-role-id]
+--rw policy-role-id uint16
+--rw name string
+--rw date yang:date-and-time
+--rw access-profile string
+--rw policy-user* [policy-user-id]
+--rw policy-user-id uint16
+--rw name string
+--rw date yang:date-and-time
+--rw password string
+--rw email string
+--rw scope-type? string
+--rw scope-reference? string
+--rw role string
+--rw policy-mgmt-auth-method* [policy-mgmt-auth-method-id]
+--rw policy-mgmt-auth-method-id uint16
+--rw name string
+--rw date yang:date-and-time
+--rw authentication-method enumeration
+--rw mutual-authentication boolean
+--rw token-server inet:ipv4-address
+--rw certificate-server inet:ipv4-address
+--rw single-sign-on-server inet:ipv4-address
+--rw endpoint-group
+--rw meta-data-source* [meta-data-source-id]
+--rw meta-data-source-id uint16
+--rw name string
+--rw date yang:date-and-time
+--rw tag-type? boolean
+--rw tag-server-information? inet:ipv4-address
+--rw tag-application-protocol? string
+--rw tag-server-credential? string
+--rw user-group* [user-group-id]
+--rw user-group-id uint16
+--rw name? string
+--rw date? yang:date-and-time
+--rw group-type? enumeration
+--rw meta-data-server? inet:ipv4-address
+--rw group-member? string
+--rw risk-level? uint16

```

```

+--rw device-group* [device-group-id]
|   +--rw device-group-id      uint16
|   +--rw name?                string
|   +--rw date?                yang:date-and-time
|   +--rw group-type?          enumeration
|   +--rw meta-data-server?    inet:ipv4-address
|   +--rw group-member?        string
|   +--rw risk-level?          uint16
+--rw application-group* [application-group-id]
|   +--rw application-group-id  uint16
|   +--rw name?                string
|   +--rw date?                yang:date-and-time
|   +--rw group-type?          enumeration
|   +--rw meta-data-server?    inet:ipv4-address
|   +--rw group-member?        string
|   +--rw risk-level?          uint16
+--rw location-group* [location-group-id]
|   +--rw location-group-id     uint16
|   +--rw name?                string
|   +--rw date?                yang:date-and-time
|   +--rw group-type?          enumeration
|   +--rw meta-data-server?    inet:ipv4-address
|   +--rw group-member?        string
|   +--rw risk-level?          uint16
+--rw threat-feed
|   +--rw threat-feed* [threat-feed-id]
|   |   +--rw threat-feed-id    uint16
|   |   +--rw name?            string
|   |   +--rw date?            yang:date-and-time
|   |   +--rw feed-type        enumeration
|   |   +--rw feed-server?     inet:ipv4-address
|   |   +--rw feed-priority?   uint16
|   +--rw custom-list* [custom-list-id]
|   |   +--rw custom-list-id    uint16
|   |   +--rw name?            string
|   |   +--rw date?            yang:date-and-time
|   |   +--rw list-type        enumeration
|   |   +--rw list-property     enumeration
|   |   +--rw list-content?     string
|   +--rw malware-scan-group* [malware-scan-group-id]
|   |   +--rw malware-scan-group-id  uint16
|   |   +--rw name?                string
|   |   +--rw date?                yang:date-and-time
|   |   +--rw signature-server?    inet:ipv4-address
|   |   +--rw file-types?          string
|   |   +--rw malware-signatures?  string
|   +--rw event-map-group* [event-map-group-id]
|   |   +--rw event-map-group-id    uint16

```

```

|      +--rw name?                string
|      +--rw date?                yang:date-and-time
|      +--rw security-events?    string
|      +--rw threat-map?         string
+--rw telemetry-data
  +--rw telemetry-data* [telemetry-data-id]
    +--rw telemetry-data-id      uint16
    +--rw name?                  string
    +--rw date?                  yang:date-and-time
    +--rw logs?                  boolean
    +--rw syslogs?               boolean
    +--rw snmp?                  boolean
    +--rw sflow?                 boolean
    +--rw netflow?               boolean
    +--rw interface-stats?       boolean
  +--rw telemetry-source* [telemetry-source-id]
    +--rw telemetry-source-id    uint16
    +--rw name?                  string
    +--rw date?                  yang:date-and-time
    +--rw source-type?           enumeration
    +--rw nsf-source?            inet:ipv4-address
    +--rw nsf-credentials?       string
    +--rw collection-interval?   uint16
    +--rw collection-method?     enumeration
    +--rw heartbeat-interval?    uint16
    +--rw qos-marking?           uint16
  +--rw telemetry-destination* [telemetry-destination-id]
    +--rw telemetry-destination-id uint16
    +--rw name?                  string
    +--rw date?                  yang:date-and-time
    +--rw collector-source?       inet:ipv4-address
    +--rw collector-credentials? string
    +--rw data-encoding?          string
    +--rw data-transport?         enumeration

```

Figure 2: Generic Data Model for Security Policies for cf Interface

5. YANG Data Model for Security Policies for Consumer-Facing Interface

This section describes a YANG data model for Consumer-Facing Interface, based on the information model of Consumer-Facing Interface to security controller [client-facing-inf-im].

```

<CODE BEGINS> file "policy-general.yang"
module ietf-policy-general {
  namespace

```



```
    "urn:ietf:params:xml:ns:yang:ietf-policy-general";
prefix
  cf-interface;

import ietf-yang-types{
  prefix yang;
}

import ietf-inet-types{
  prefix inet;
}

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

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

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

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

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

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

revision "2018-07-02"{
  description "fourth revision";
  reference
    "draft-kumar-i2nsf-client-facing-interface-im-04";
}

//Groupings
container policy {
  description
    "This object is a policy instance to have
    complete information such as where and when
    a policy need to be applied.";

  list rule {
    key "rule-id";
```

```
leaf rule-id {
  type uint16;
  description
    "This is ID for rules.";
}
description
  "This is a container for rules.";
leaf name {
  type string;
  description
    "This field identifies the name of this object.";
}

leaf date {
  type yang:date-and-time;
  description
    "Date this object was created or last
    modified";
}

leaf case {
  type string;
  description
    "to identify whether the rule belongs to
    web filter or enterprise mode.";
}

list event {
  key "event-id";
  description
    "This represents the security event of a
    policy-rule.";

  leaf event-id {
    type string;
    mandatory true;
    description
      "This represents the event-id.";
  }

  leaf name {
    type string;
    description
      "This field identifies the name of this object.";
  }

  leaf date {
    type yang:date-and-time;
```

```
    description
      "Date this object was created or last
      modified";
  }

  leaf event-type {
    type string;
    description
      "This field identifies the event of
      policy enforcement trigger type.";
  }

  leaf time-information {
    type string;
    description
      "This field contains time calendar such as
      BEGIN-TIME and END-TIME for one time
      enforcement or recurring time calendar for
      periodic enforcement.";
  }

  leaf event-map-group {
    type leafref {
      path "/threat-feed/event-map-group/event-map-group-id";
    }
    description
      "This field contains security events or threat
      map in order to determine when a policy need
      to be activated. This is a reference to
      Evnet-Map-Group.";
  }

  leaf enable {
    type boolean;
    description
      "This determines whether the condition
      matches the security event or not.";
  }
}

list condition {
  key "condition-id";
  description
    "This represents the condition of a
    policy-rule.";

  leaf condition-id {
    type string;
```

```
    description
    "This represents the condition-id.";
}

leaf source {
    type string;
    description
    "This field identifies the source of
    the traffic. This could be reference to
    either 'Policy Endpoint Group' or
    'Threat-Feed' or 'Custom-List' if Security
    Admin wants to specify the source; otherwise,
    the default is to match all traffic.";
}

leaf destination {
    type string;
    description
    "This field identifies the source of
    the traffic. This could be reference to
    either 'Policy Endpoint Group' or
    'Threat-Feed' or 'Custom-List' if Security
    Admin wants to specify the source; otherwise,
    the default is to match all traffic.";
}

leaf match {
    type boolean;
    description
    "This field identifies the match criteria used to
    evaluate whether the specified action need to be
    taken or not. This could be either a Policy-
    Endpoint-Group identifying a Application set or a
    set of traffic rules.";
}

leaf match-direction {
    type string;
    description
    "This field identifies if the match criteria is
    to evaluated for both direction of the traffic or
    only in one direction with default of allowing in
    the other direction for stateful match conditions.
    This is optional and by default rule should apply
    in both directions.";
}

leaf exception {
```

```
    type string;
    description
      "This field identifies the exception
      consideration when a rule is evaluated for a
      given communication. This could be reference to
      Policy-Endpoint-Group object or set of traffic
      matching criteria.";
  }
}

list policy-action {
  key "policy-action-id";

  leaf policy-action-id {
    type string;
    mandatory true;
    description
      "this represents the policy-action-id.";
  }
  description
    "This object represents actions that a
    Security Admin wants to perform based on
    a certain traffic class.";

  leaf name {
    type string;
    description
      "The name of the policy-action object.";
  }

  leaf date {
    type yang:date-and-time;
    description
      "When the object was created or last
      modified.";
  }

  leaf primary-action {
    type string;
    description
      "This field identifies the action when a rule
      is matched by NSF. The action could be one of
      'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
      'AUTHENTICATE-SESSION', 'IPS', 'APP-FIREWALL', etc.";
  }

  leaf secondary-action {
    type string;
  }
}
```

```
    description
      "This field identifies additional actions if
      a rule is matched. This could be one of 'LOG',
      'SYSLOG', 'SESSION-LOG', etc.";
  }

  leaf owner {
    type string;
    description
      "This field defines the owner of this
      policy. Only the owner is authorized to
      modify the contents of the policy.";
  }
}

container multi-tenancy {
  description
    "The descriptions of multi-tenancy.";

  list policy-domain {
    key "policy-domain-id";

    leaf policy-domain-id {
      type uint16;
      description
        "This represents the list of domains.";
    }
    description
      "this represent the list of policy domains";
    leaf name {
      type string;
      mandatory true;
      description
        "Name of the organization or customer representing
        this domain.";
    }

    leaf address {
      type string;
      description
        "address of an organization or customer.";
    }

    leaf contact {
      type string;
    }
  }
}
```

```
    mandatory true;
    description
        "contact information of the organization
        or customer.";
}

leaf date {
    type yang:date-and-time;
    mandatory true;
    description
        "The date when this account was created
        or last modified.";
}

list policy-tenant {
    key "policy-tenant-id";
    leaf policy-tenant-id {
        type uint16;
        description
            "The policy tenant id.";
    }
    description
        "This represents the list of tenants";

    leaf name {
        type string;
        mandatory true;
        description
            "Name of the Department or Division within
            an organization.";
    }

    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
            "Date this account was created or last modified.";
    }

    leaf domain {
        type leafref {
            path "/multi-tenancy/policy-domain/policy-domain-id";
        }
        description
            "This field identifies the domain to which this
            tenant belongs. This should be reference to a
            'Policy-Domain' object.";
    }
}
}
```

```
    leaf authentication-method {
        type leafref {
            path "/multi-tenancy/policy-mgmt-auth-method/policy-mgmt-auth-meth
od-id";
        }

        description
            "Authentication method to be used for this domain.
            It should be a reference to a 'policy-mgmt-auth-method'
            object.";
    }
}

list policy-role {
    key "policy-role-id";

    leaf policy-role-id {
        type uint16;
        mandatory true;
        description
            "This defines a set of permissions assigned
            to a user in an organization that want to manage
            its own Security Policies.";
    }
    description
        "This represents the list of policy roles.";

    leaf name {
        type string;
        mandatory true;
        description
            "This field identifies name of the role.";
    }

    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
            "Date this role was created or last modified.";
    }

    leaf access-profile {
        type string;
        mandatory true;
        description
            "This field identifies the access profile for the
            role. The profile grants or denies access to policy
            objects. Multiple access profiles can be
            concatenated together.";
    }
}
```



```
    }  
  }  
  
  list policy-user {  
    key "policy-user-id";  
  
    leaf policy-user-id {  
      type uint16;  
      description  
        "This represents the policy-user-id.";  
    }  
    description  
      "This represents the list of policy users.";  
    leaf name {  
      type string;  
      mandatory true;  
      description  
        "The name of a user.";  
    }  
  
    leaf date {  
      type yang:date-and-time;  
      mandatory true;  
      description  
        "Date this user was created or last modified";  
    }  
  
    leaf password {  
      type string;  
      mandatory true;  
      description  
        "User password for basic authentication";  
    }  
  
    leaf email {  
      type string;  
      mandatory true;  
      description  
        "The email account of a user";  
    }  
  
    leaf scope-type {  
      type string;  
      description  
        "identifies whether a user has domain-wide  
        or tenant-wide privileges";  
    }  
  }
```

```
    leaf scope-reference {
        type string;
        description
            "This references policy-domain or policy-tenant
            to identify the scope.";
    }

    leaf role {
        type string;
        mandatory true;
        description
            "This references policy-role to define specific
            permissions";
    }
}

list policy-mgmt-auth-method {
    key "policy-mgmt-auth-method-id";

    leaf policy-mgmt-auth-method-id {
        type uint16;
        description
            "This represents the authentication method id.";
    }
    description
        "The descriptions of policy management
        authentication methods.";
    leaf name {
        type string;
        mandatory true;
        description
            "name of the authentication method";
    }

    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
            "date when the authentication method
            was created";
    }

    leaf authentication-method {
        type enumeration{
            enum password{
                description
                    "password-based authentication.";
            }
        }
    }
}
```

```
        enum token{
            description
                "token-based authentication.";
        }
        enum certificate{
            description
                "certificate-based authentication.";
        }
    }
    mandatory true;
    description
        "The description of authentication method;
        token-based, password, certificate,
        single-sign-on";
}

leaf mutual-authentication {
    type boolean;
    mandatory true;
    description
        "To identify whether the authentication
        is mutual";
}

leaf token-server {
    type inet:ipv4-address;
    mandatory true;
    description
        "The token-server information if the
        authentication method is token-based";
}

leaf certificate-server {
    type inet:ipv4-address;
    mandatory true;
    description
        "The certificate-server information if
        the authentication method is certificate-based";
}

leaf single-sing-on-server {
    type inet:ipv4-address;
    mandatory true;
    description
        "The single-sign-on-server information
        if the authentication method is
        single-sign-on-based";
}
```

```
    }  
  }  
  container endpoint-group {  
    description  
      "A logical entity in their business  
      environment, where a security policy  
      is to be applied.";  
  
    list meta-data-source {  
      key "meta-data-source-id";  
      leaf meta-data-source-id {  
        type uint16;  
        mandatory true;  
        description  
          "This represents the meta-data source id.";  
      }  
      description  
        "This represents the meta-data source.";  
  
      leaf name {  
        type string;  
        mandatory true;  
        description  
          "This identifies the name of the  
          meta-datas-ource.";  
      }  
  
      leaf date {  
        type yang:date-and-time;  
        mandatory true;  
        description  
          "This identifies the date this object was  
          created or last modified.";  
      }  
  
      leaf tag-type {  
        type boolean;  
        description  
          "This identifies the group type; user group,  
          app group or device group.";  
      }  
  
      leaf tag-server-information {  
        type inet:ipv4-address;  
        description  
          "The description of suthentication method;  
          token-based, password, certificate,  
          single-sign-on";  
      }  
    }  
  }  
}
```

```
    }

    leaf tag-application-protocol {
        type string;
        description
            "This field identifies the protocol e.g. LDAP,
            Active Directory, or CMDB";
    }

    leaf tag-server-credential {
        type string;
        description
            "This field identifies the credential
            information needed to access the tag server";
    }
}

list user-group{
    key "user-group-id";

    leaf user-group-id {
        type uint16;
        mandatory true;
        description
            "This represents the the user group id.";
    }
    description
        "This represents the user group.";

    leaf name {
        type string;
        description
            "This field identifies the name of user-group.";
    }

    leaf date {
        type yang:date-and-time;
        description
            "when this user-group was created or last modified.";
    }

    leaf group-type {
        type enumeration{
            enum user-tag{
                description
                    "The user group is based on user-tag.";
            }
            enum user-name{
```

```
        description
            "The user group is based on user-name.";
    }
    enum ip-address{
        description
            "The user group is based on ip-address.";
    }
}

description
    "This describes the group type; User-tag,
    User-name or IP-address.";
}

leaf meta-data-server {
    type inet:ipv4-address;
    description
        "This references metadata source";
}

leaf group-member {
    type string;
    description
        "This describes the user-tag information";
}

leaf risk-level {
    type uint16;
    description
        "This represents the threat level; valid range
        may be 0 to 9.";
}
}

list device-group {
    key "device-group-id";
    leaf device-group-id {
        type uint16;
        description
            "This represents a device group id.";
    }
    description
        "This represents a device group.";
    leaf name {
        type string;
        description
            "This field identifies the name of
            a device-group.";
    }
}
```

```
}
leaf date {
  type yang:date-and-time;
  description
    "The date when this group was create or
    last modified.";
}

leaf group-type {
  type enumeration{
    enum device-tag{
      description
        "The device group is based on device-tag.";
    }
    enum device-name{
      description
        "The device group is based on device-name.";
    }
    enum ip-address{
      description
        "The device group is based on ip-address.";
    }
  }
  description
    "This describes the group type; device-tag,
    device-name or IP-address.";
}

leaf meta-data-server {
  type inet:ipv4-address;
  description
    "This references meta-data-source
    object.";
}

leaf group-member {
  type string;
  description
    "This describes the device-tag, device-name or
    IP-address information";
}

leaf risk-level {
  type uint16;
  description
    "This represents the threat level; valid range
    may be 0 to 9.";
}
```

```
}  
  
list application-group{  
  key "application-group-id";  
  leaf application-group-id {  
    type uint16;  
    description  
      "This represents an application group id.";  
  }  
  description  
    "This represents an application group.";  
  leaf name {  
    type string;  
    description  
      "This field identifies the name of  
      an application group";  
  }  
  
  leaf date {  
    type yang:date-and-time;  
    description  
      "The date when this group was created or  
      last modified.";  
  }  
  
  leaf group-type {  
    type enumeration{  
      enum application-tag{  
        description  
          "The application group is based on application-tag.";  
      }  
      enum device-name{  
        description  
          "The application group is based on application-name.";  
      }  
      enum ip-address{  
        description  
          "The application group is based on ip-address.";  
      }  
    }  
    description  
      "This identifies the group type;  
      application-tag, application-name or  
      IP-address.";  
  }  
  
  leaf meta-data-server {  
    type inet:ipv4-address;
```



```
        description
            "This references meta-data-source
            object.";
    }

    leaf group-member {
        type string;
        description
            "This describes the application-tag,
            application-name or IP-address information";
    }

    leaf risk-level {
        type uint16;
        description
            "This represents the threat level; valid range
            may be 0 to 9.";
    }
}

list location-group{
    key "location-group-id";
    leaf location-group-id {
        type uint16;
        description
            "This represents a location group id.";
    }
    description
        "This represents a location group.";

    leaf name {
        type string;
        description
            "This field identifies the name of
            a location group";
    }

    leaf date {
        type yang:date-and-time;
        description
            "The date when this group was created or
            last modified.";
    }

    leaf group-type {
        type enumeration{
            enum location-tag{
```

```
        description
            "The location group is based on location-tag.";
    }
    enum location-name{
        description
            "The location group is based on location-name.";
    }
    enum ip-address{
        description
            "The location group is based on ip-address.";
    }
}
description
    "This identifies the group type;
    location-tag, location-name or
    IP-address.";
}

leaf meta-data-server {
    type inet:ipv4-address;
    description
        "This references meta-data-source
        object.";
}

leaf group-member {
    type string;
    description
        "This describes the location-tag,
        location-name or IP-address information";
}

leaf risk-level {
    type uint16;
    description
        "This represents the threat level; valid range
        may be 0 to 9.";
}
}

container threat-feed {
    description
        "this describes the list of threat-feed.";

    list threat-feed {
        key "threat-feed-id";
        leaf threat-feed-id {
```

```
type uint16;
mandatory true;
description
    "This represents the threat-feed-id.";
}
description
    "This represents the threat feed within the
    threat-prevention-list.";
leaf name {
    type string;
    description
        "Name of the theat feed.";
}

leaf date {
    type yang:date-and-time;
    description
        "when the threat-feed was created.";
}

leaf feed-type {
    type enumeration {
        enum unknown {
            description
                "feed-type is unknown.";
        }
        enum ip-address {
            description
                "feed-type is IP address.";
        }
        enum url {
            description
                "feed-type is URL.";
        }
    }
    mandatory true;
    description
        "This determined whether the feed-type is IP address
        based or URL based.";
}

leaf feed-server {
    type inet:ipv4-address;
    description
        "this contains threat feed server information.";
}

leaf feed-priority {
```

```
        type uint16;
        description
            "this describes the priority of the threat from
            0 to 5, where 0 means the threat is minimum and
            5 meaning the maximum.";
    }
}

list custom-list {
    key "custom-list-id";
    leaf custom-list-id {
        type uint16;
        description
            "this describes the custom-list-id.";
    }
    description
        "this describes the threat-prevention custom list.";
    leaf name {
        type string;
        description
            "Name of the custom-list.";
    }

    leaf date {
        type yang:date-and-time;
        description
            "when the custom list was created.";
    }

    leaf list-type {
        type enumeration {
            enum unknown {
                description
                    "list-type is unknown.";
            }
            enum ip-address {
                description
                    "list-type is IP address.";
            }
            enum mac-address {
                description
                    "list-type is MAC address.";
            }
            enum url {
                description
                    "list-type is URL.";
            }
        }
    }
}
```

```
        mandatory true;
        description
            "This determined whether the feed-type is IP address
            based or URL based.";
    }

    leaf list-property {
        type enumeration {
            enum unknown {
                description
                    "list-property is unknown.";
            }
            enum blacklist {
                description
                    "list-property is blacklist.";
            }
            enum whitelist {
                description
                    "list-property is whitelist.";
            }
        }
        mandatory true;
        description
            "This determined whether the list-type is blacklist
            or whitelist.";
    }

    leaf list-content {
        type string;
        description
            "This describes the contents of the custom-list.";
    }
}

list malware-scan-group {
    key "malware-scan-group-id";
    leaf malware-scan-group-id {
        type uint16;
        mandatory true;
        description
            "This is the malware-scan-group-id.";
    }
    description
        "This represents the malware-scan-group.";
    leaf name {
        type string;
        description
            "Name of the malware-scan-group.";
    }
}
```

```
    }

    leaf date {
        type yang:date-and-time;
        description
            "when the malware-scan-group was created.";
    }

    leaf signature-server {
        type inet:ipv4-address;
        description
            "This describes the signature server of the
            malware-scan-group.";
    }

    leaf file-types {
        type string;
        description
            "This contains a list of file types needed to
            be scanned for the virus.";
    }

    leaf malware-signatures {
        type string;
        description
            "This contains a list of malware signatures or hash.";
    }
}

list event-map-group {
    key "event-map-group-id";
    leaf event-map-group-id {
        type uint16;
        mandatory true;
        description
            "This is the event-map-group-id.";
    }
    description
        "This represents the event map group.";

    leaf name {
        type string;
        description
            "Name of the event-map.";
    }

    leaf date {
        type yang:date-and-time;
```

```
        description
            "when the event-map was created.";
    }

    leaf security-events {
        type string;
        description
            "This contains a list of security events.";
    }

    leaf threat-map {
        type string;
        description
            "This contains a list of threat levels.";
    }
}

container telemetry-data {
    description
        "Telemetry provides visibility into the network
        activities which can be tapped for further
        security analytics, e.g., detecting potential
        vulnerabilities, malicious activities, etc.";

    list telemetry-data {
        key "telemetry-data-id";

        leaf telemetry-data-id {
            type uint16;
            mandatory true;
            description
                "This is ID for telemetry-data-id.";
        }
        description
            "This is ID for telemetry-data.";

        leaf name {
            type string;
            description
                "Name of the telemetry-data object.";
        }

        leaf date {
            type yang:date-and-time;
            description
                "This field states when the telemery-data
                object was created.";
        }
    }
}
```

```
    }

    leaf logs {
        type boolean;
        description
            "This field identifies whether logs
            need to be collected.";
    }

    leaf syslogs {
        type boolean;
        description
            "This field identifies whether System logs
            need to be collected.";
    }

    leaf snmp {
        type boolean;
        description
            "This field identifies whether 'SNMP traps' and
            'SNMP alarms' need to be collected.";
    }

    leaf sflow {
        type boolean;
        description
            "This field identifies whether 'sFlow' data
            need to be collected.";
    }

    leaf netflow {
        type boolean;
        description
            "This field identifies whether 'NetFlow' data
            need to be collected.";
    }

    leaf interface-stats {
        type boolean;
        description
            "This field identifies whether 'Interface' data
            such as packet bytes and counts need to be
            collected.";
    }
}

list telemetry-source {
    key "telemetry-source-id";
```



```
leaf telemetry-source-id {
  type uint16;
  mandatory true;
  description
    "This is ID for telemetry-source-id.";
}
description
  "This is ID for telemetry-source.";

leaf name {
  type string;
  description
    "This identifies the name of this object.";
}

leaf date {
  type yang:date-and-time;
  description
    "Date this object was created or last modified";
}

leaf source-type {
  type enumeration {
    enum network-nsf {
      description
        "NSF telemetry source type is network-nsf.";
    }

    enum firewall-nsf {
      description
        "NSF telemetry source type is firewall-nsf.";
    }

    enum ids-nsf {
      description
        "NSF telemetry source type is ids-nsf.";
    }

    enum ips-nsf {
      description
        "NSF telemetry source type is ips-nsf.";
    }

    enum proxy-nsf {
      description
        "NSF telemetry source type is proxy-nsf.";
    }

    enum other-nsf {
      description
        "NSF telemetry source type is other-nsf.";
    }
  }
}
```

```
    }
    description
        "This should have one of the following type of
        the NSF telemetry source: NETWORK-NSF,
        FIREWALL-NSF, IDS-NSF, IPS-NSF,
        PROXY-NSF, VPN-NSF, DNS, ACTIVE-DIRECTORY,
        IP Reputation Authority, Web Reputation
        Authority, Anti-Malware Sandbox, Honey Pot,
        DHCP, Other Third Party, ENDPOINT";
    }

    leaf nsf-source {
        type inet:ipv4-address;
        description
            "This field contains information such as
            IP address and protocol (UDP or TCP) port
            number of the NSF providing telemetry data.";
    }

    leaf nsf-credentials {
        type string;
        description
            "This field contains username and password
            to authenticate with the NSF.";
    }

    leaf collection-interval {
        type uint16;
        units seconds;
        default 5000;
        description
            "This field contains time in milliseconds
            between each data collection. For example,
            a value of 5000 means data is streamed to
            collector every 5 seconds. Value of 0 means
            data streaming is event-based";
    }

    leaf collection-method {
        type enumeration {
            enum unknown {
                description
                    "collection-method is unknown.";
            }
            enum push-based {
                description
                    "collection-method is PUSH-based.";
            }
        }
    }
```

```
        enum pull-based {
            description
                "collection-method is PULL-based.";
        }
    }
    description
        "This field contains a method of collection,
        i.e., whether it is PUSH-based or PULL-based.";
}
leaf heartbeat-interval {
    type uint16;
    units seconds;
    description
        "time in seconds the source sends telemetry
        heartbeat.";
}

leaf qos-marking {
    type uint16;
    description
        "DSCP value must be contained in this field.";
}
}

list telemetry-destination {
    key "telemetry-destination-id";

    leaf telemetry-destination-id {
        type uint16;
        description
            "this represents the telemetry-destination-id";
    }
    description
        "This object contains information related to
        telemetry destination. The destination is
        usually a collector which is either a part of
        Security Controller or external system
        such as Security Information and Event
        Management (SIEM).";

    leaf name {
        type string;
        description
            "This identifies the name of this object.";
    }

    leaf date {
        type yang:date-and-time;
    }
}
```

```
    description
      "Date this object was created or last
      modified";
  }

  leaf collector-source {
    type inet:ipv4-address;
    description
      "This field contains information such as
      IP address and protocol (UDP or TCP) port
      number for the collector's destination.";
  }

  leaf collector-credentials {
    type string;
    description
      "This field contains the username and
      password for the collector.";
  }

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

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

Figure 3: YANG for policy-general

6. Security Considerations

The data model for the I2NSF Consumer-Facing Interface is derived from the I2NSF Consumer-Facing Interface Information Model [client-facing-inf-im], so the same security considerations with the information model should be included in this document. The data model needs to support a mechanism to protect Consumer-Facing Interface to Security Controller.

7. 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).

This document has greatly benefited from inputs by Hyoungshick Kim, Mahdi F. Dachmehchi, Seungjin Lee, Jinyong Tim Kim, and Daeyoung Hyun.

8. Contributors

I2NSF is a group effort. The following people actively contributed to the consumer facing interface data model, and are considered co-authors: o Hyoungshick Kim (Sungkyunkwan University) o Seungjin Lee (Sungkyunkwan University)

9. References

9.1. Normative References

[RFC3444] Pras, A., "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

9.2. Informative References

[client-facing-inf-im]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-kumar-i2nsf-client-facing-interface-im-04 (work in progress), July 2017.

- [client-facing-inf-req]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislaamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-03 (work in progress), July 2017.
- [i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-08 (work in progress), October 2017.
- [i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Birkholz, H., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-terminology-04 (work in progress), July 2017.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Appendix A. Changes from draft-ietf-i2nsf-consumer-facing-interface-dm-00

The following changes have been made from draft-jeong-i2nsf-consumer-facing-interface-dm-05:

- o In Section 3, the high-level abstraction of the consumer facing interface has been added.
- o The overall organization of the YANG data model and its data types have also been reviewed and corrected, and produced the corresponding data tree as shown in the Section 5.
- o Overall editorial errors have been corrected.

Appendix B. Use Case: Policy Instance Example for VoIP/VoLTE Security Services

A common scenario for VoIP/VoLTE policy enforcement could be that a malicious call is made to a benign user of any telecommunication company. For example, imagine a case where a company "A" employs a hacker with a malicious attempt to hack a user's phone with malware. The company "A" is located in a country, such as Africa, and uses the user's hacked phone to call the company. The hacked user is unaware of the company "A" so complains about the international call that was made to the company "B", which is the user's telecommunications company. The company "A" charges the company "B" for the international call. The company "B" cannot charge the user for the call, and has no choice but to pay the company "A". The following shows the example data tree model for the VoIP/VoLTE services. Multi-tenancy, endpoint groups, threat prevention, and telemetry data components are general part of the tree model, so we can just modify the policy instance in order to generate and enforce high-level policies. The policy-calendar can act as a scheduler to set the start and end time to block calls which uses suspicious ids, or calls from other countries.

```

module: policy-voip
  +-rw policy-voip
  |   +-rw rule-voip* [rule-voip-id]
  |   |   +-rw rule-voip-id          uint16
  |   |   +-rw name?                 string
  |   |   +-rw date?                 yang:date-and-time
  |   |   +-rw event* [event-id]
  |   |   |   +-rw event-id          string
  |   |   |   +-rw name?              string
  |   |   |   +-rw date?              yang:date-and-time
  |   |   |   +-rw event-type?       string

```

```

+--rw Time-Information?      string
+--rw event-map-group?      -> /threat-feed/event-map-group
                               /event-map-group-id
+--rw enable?                boolean
+--rw condition* [condition-id]
+--rw condition-id           string
+--rw source-caller?         -> /threat-feed/threat-feed
                               /threat-feed-id
+--rw destination-callee?   -> /threat-feed/custom-list
                               /custom-list-id
+--rw match?                 boolean
+--rw match-direction?      string
+--rw exception?             string
+--rw action* [action-id]
+--rw action-id              string
+--rw name?                  string
+--rw date?                  yang:date-and-time
+--rw primary-action?        string
+--rw secondary-action?      string
+--rw precedence?            uint16
+--rw owner* [owner-id]
+--rw owner-id               string
+--rw name?                  string
+--rw date?                  yang:date-and-time
+--rw threat-feed
+--rw threat-feed* [threat-feed-id]
+--rw threat-feed-id         uint16
+--rw name?                  string
+--rw date?                  yang:date-and-time
+--rw feed-type               enumeration
+--rw feed-server?           inet:ipv4-address
+--rw feed-priority?         uint16
+--rw custom-list* [custom-list-id]
+--rw custom-list-id         uint16
+--rw name?                  string
+--rw date?                  yang:date-and-time
+--rw list-type               enumeration
+--rw list-property           enumeration
+--rw list-content?          string
+--rw malware-scan-group* [malware-scan-group-id]
+--rw malware-scan-group-id  uint16
+--rw name?                  string
+--rw date?                  yang:date-and-time
+--rw signature-server?      inet:ipv4-address
+--rw file-types?            string
+--rw malware-signatures?    string
+--rw event-map-group* [event-map-group-id]
+--rw event-map-group-id     uint16

```



```

+--rw name?                string
+--rw date?                yang:date-and-time
+--rw security-events?     string
+--rw threat-map?          string

```

Figure 4: Policy Instance Example for VoIP/VoLTE Security Services

Appendix C. Policy Instance YANG Example for VoIP/VoLTE Security Services

The following YANG data model is a policy instance for VoIP/VoLTE security services. The policy-calendar can act as a scheduler to set the start time and end time to block malicious calls which use suspicious IDs, or calls from other countries.

```

<CODE BEGINS> file "ietf-i2nsf-cf-interface-voip.yang"

module ietf-policy-voip {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-policy-voip";
  prefix
    "cf-interface";

  import ietf-yang-types {
    prefix yang;
  }

  import ietf-inet-types {
    prefix inet;
  }

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

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

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

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

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

```

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

revision "2018-07-02" {
  description "sixth revision";
  reference
    "draft-kumar-i2nsf-client-facing-interface-im-04";
}

container policy-voip {
  description
    "This object is a policy instance to have
    complete information such as where and when
    a policy need to be applied.";
  list rule-voip {
    key "rule-voip-id";
    leaf rule-voip-id {
      type uint16;
      mandatory true;
      description
        "This is ID for rules.";
    }
    description
      "This is a container for rules.";
    leaf name {
      type string;
      description
        "This field identifies the name of this object.";
    }
    leaf date {
      type yang:date-and-time;
      description
        "Date this object was created or last
        modified";
    }
  }
  list event {
    key "event-id";
    description
      "This represents the security event of a
      policy-rule.";
    leaf event-id {
      type string;
      mandatory true;
      description
        "This represents the event-id.";
    }
    leaf name {
```

```
        type string;
        description
            "This field identifies the name of this object.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "Date this object was created or last
            modified";
    }
    leaf event-type {
        type string;
        description
            "This field identifies the event event type
            .";
    }
    leaf Time-Information {
        type string;
        description
            "This field contains time calendar such as
            BEGIN-TIME and END-TIME for one time
            enforcement or recurring time calendar for
            periodic enforcement.";
    }
    leaf event-map-group {
        type leafref{
            path "/threat-feed/event-map-group/event-map-group-id";
        }
        description
            "This field contains security events or threat
            map in order to determine when a policy need
            to be activated. This is a reference to
            Evnet-Map-Group.";
    }
    leaf enable {
        type boolean;
        description
            "This determines whether the condition
            matches the security event or not.";
    }
}
list condition {
    key "condition-id";
    description
        "This represents the condition of a
        policy-rule.";
    leaf condition-id {
        type string;
    }
}
```

```
    description
      "This represents the condition-id.";
  }
  leaf source-caller {
    type leafref {
      path "/threat-feed/threat-feed/threat-feed-id";
    }
    description
      "This field identifies the source of
      the traffic. This could be reference to
      either 'Policy Endpoint Group' or
      'Threat-Feed' or 'Custom-List' if Security
      Admin wants to specify the source; otherwise,
      the default is to match all traffic.";
  }
  leaf destination-callee {
    type leafref {
      path "/threat-feed/custom-list/custom-list-id";
    }
    description
      "This field identifies the source of
      the traffic. This could be reference to
      either 'Policy Endpoint Group' or
      'Threat-Feed' or 'Custom-List' if Security
      Admin wants to specify the source; otherwise,
      the default is to match all traffic.";
  }
  leaf match {
    type boolean;
    description
      "This field identifies the match criteria used to
      evaluate whether the specified action need to be
      taken or not. This could be either a Policy-
      Endpoint-Group identifying a Application set or a
      set of traffic rules.";
  }
  leaf match-direction {
    type string;
    description
      "This field identifies if the match criteria is
      to evaluated for both direction of the traffic or
      only in one direction with default of allowing in
      the other direction for stateful match conditions.
      This is optional and by default rule should apply
      in both directions.";
  }
  leaf exception {
    type string;
```

```
        description
            "This field identifies the exception
            consideration when a rule is evaluated for a
            given communication. This could be reference to
            Policy-Endpoint-Group object or set of traffic
            matching criteria.";
    }
}
list action {
    key "action-id";
    leaf action-id {
        type string;
        mandatory true;
        description
            "this represents the policy-action-id.";
    }
    description
        "This object represents actions that a
        Security Admin wants to perform based on
        a certain traffic class.";
    leaf name {
        type string;
        description
            "The name of the policy-action object.";
    }

    leaf date {
        type yang:date-and-time;
        description
            "When the object was created or last
            modified.";
    }

    leaf primary-action {
        type string;
        description
            "This field identifies the action when a rule
            is matched by NSF. The action could be one of
            'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
            'AUTHENTICATE-SESSION', 'IPS', 'APP-FIREWALL', etc.";
    }

    leaf secondary-action {
        type string;
        description
            "This field identifies additional actions if
            a rule is matched. This could be one of 'LOG',
            'SYSLOG', 'SESSION-LOG', etc.";
    }
}
```

```
    leaf precedence {
        type uint16;
        description
            "This field identifies the precedence
            assigned to this rule by Security Admin.
            This is helpful in conflict resolution
            when two or more rules match a given
            traffic class.";
    }
}
list owner {
    key "owner-id";
    leaf owner-id {
        type string;
        mandatory true;
        description
            "this represents the owner-id.";
    }
    description
        "This field defines the owner of this policy.
        Only the owner is authorized to modify the
        contents of the policy.";
    leaf name {
        type string;
        description
            "The name of the owner.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "When the object was created or last
            modified.";
    }
}
}
container threat-feed {
    description
        "this describes the list of threat-feed.";

    list threat-feed {
        key "threat-feed-id";
        leaf threat-feed-id {
            type uint16;
            mandatory true;
            description
                "This represents the threat-feed-id.";
        }
        description

```

```
    "This represents the threat feed within the
    threat-prevention-list.";
  leaf name {
    type string;
    description
      "Name of the theat feed.";
  }

  leaf date {
    type yang:date-and-time;
    description
      "when the threat-feed was created.";
  }

  leaf feed-type {
    type enumeration {
      enum unknown {
        description
          "feed-type is unknown.";
      }
      enum ip-address {
        description
          "feed-type is IP address.";
      }
      enum url {
        description
          "feed-type is URL.";
      }
    }
    mandatory true;
    description
      "This determined whether the feed-type is IP address
      based or URL based.";
  }

  leaf feed-server {
    type inet:ipv4-address;
    description
      "this contains threat feed server information.";
  }

  leaf feed-priority {
    type uint16;
    description
      "this describes the priority of the threat from
      0 to 5, where 0 means the threat is minimum and
      5 meaning the maximum.";
  }
```

```
}

list custom-list {
  key "custom-list-id";
  leaf custom-list-id {
    type uint16;
    description
      "this describes the custom-list-id.";
  }
  description
    "this describes the threat-prevention custom list.";
  leaf name {
    type string;
    description
      "Name of the custom-list.";
  }

  leaf date {
    type yang:date-and-time;
    description
      "when the custom list was created.";
  }

  leaf list-type {
    type enumeration {
      enum unknown {
        description
          "list-type is unknown.";
      }
      enum ip-address {
        description
          "list-type is IP address.";
      }
      enum mac-address {
        description
          "list-type is MAC address.";
      }
      enum url {
        description
          "list-type is URL.";
      }
    }
    mandatory true;
    description
      "This determined whether the feed-type is IP address
      based or URL based.";
  }
}
```



```
leaf list-property {
  type enumeration {
    enum unknown {
      description
        "list-property is unknown.";
    }
    enum blacklist {
      description
        "list-property is blacklist.";
    }
    enum whitelist {
      description
        "list-property is whitelist.";
    }
  }
  mandatory true;
  description
    "This determined whether the list-type is blacklist
    or whitelist.";
}

leaf list-content {
  type string;
  description
    "This describes the contents of the custom-list.";
}
}

list malware-scan-group {
  key "malware-scan-group-id";
  leaf malware-scan-group-id {
    type uint16;
    mandatory true;
    description
      "This is the malware-scan-group-id.";
  }
  description
    "This represents the malware-scan-group.";
  leaf name {
    type string;
    description
      "Name of the malware-scan-group.";
  }

  leaf date {
    type yang:date-and-time;
    description
      "when the malware-scan-group was created.";
  }
}
```

```
    }

    leaf signature-server {
        type inet:ipv4-address;
        description
            "This describes the signature server of the
            malware-scan-group.";
    }

    leaf file-types {
        type string;
        description
            "This contains a list of file types needed to
            be scanned for the virus.";
    }

    leaf malware-signatures {
        type string;
        description
            "This contains a list of malware signatures or hash.";
    }
}

list event-map-group {
    key "event-map-group-id";
    leaf event-map-group-id {
        type uint16;
        mandatory true;
        description
            "This is the event-map-group-id.";
    }
    description
        "This represents the event map group.";

    leaf name {
        type string;
        description
            "Name of the event-map.";
    }

    leaf date {
        type yang:date-and-time;
        description
            "when the event-map was created.";
    }

    leaf security-events {
        type string;
    }
}
```

```

        description
            "This contains a list of security events.";
    }

    leaf threat-map {
        type string;
        description
            "This contains a list of threat levels.";
    }
}
}
}

```

<CODE ENDS>

Figure 5: Policy Instance YANG Example for VoIP Security Services

Appendix D. Example XML output for VoIP service

In this section, we present an XML example for VoIP service. Here, we are going to drop calls commin from a country with an Ip from South Africa that is classified as malicious.

```

<?xml version="1.1" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:restconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <i2nsf-cf-interface-voip-req nc:operation="create">
        <policy-voip>
          <rule-voip>
            <rule-voip-id>01</rule-voip-id>
            <rule-voip-name>voip-policy-example</rule-voip-name>
            <rule-voip-date>2017.10.25/20:30:32</rule-voip-date>
            <event>
              <event-id>01</event-id>
              <event-name>voip_call</event-name>
              <event-date>2017.10.25/20:30:32</event-date>
              <event-type>malicious</event-type>
              <time-information>
                <begin-time>22:00</begin-time>
                <end-time>08:00</end-time>
              </time-information>
              <event-map-group>19</event-map-group>
            </event>
          </rule-voip>
        </policy-voip>
      </i2nsf-cf-interface-voip-req>
    </config>
  </edit-config>
</rpc>

```

```

        <enable>True</enable>
    </event>
    <condition>
        <condition-id>01</condition-id>
        <source-caller>105.176.0.0</source-caller>
        <destination-callee>192.168.171.35</destination-callee>
        <match-direction>default</match-direction>
        <exeption>00</exeption>
    </condition>
    <action>
        <action-id>01</action-id>
        <action-name>action-voip</action-name>
        <action-date>2017.10.25/20:30:32</action-date>
        <primary-action>DENY</primary-action>
        <secondary-action>LOG</secondary-action>
    </action>
    <precedence>none</precedence>
    <owner>
        <owner-id>01</owner-id>
        <name>i2nsf-admin</name>
    </owner>
    </rule-voip>
</policy-voip>
</i2nsf-cf-interface-voip-req>
</config>
</edit-config>
</rpc>

```

Figure 6: An XML example for VoIP service

Authors' Addresses

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

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

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

Phone: +82 31 299 4104
EMail: eskim86@skku.edu
URI: <http://seclab.skku.edu/people/eunsoo-kim/>

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

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

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

EMail: rkkumar@juniper.net

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 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.

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
3.1. Tree Diagrams	4
4. The Structure and Objective of I2NSF Security Policy	4
4.1. I2NSF Security Policy Rule	4
4.2. Event Clause	4
4.3. Condition Clause	4
4.4. Action Clause	5
5. Data Model Structure	5
5.1. I2NSF Security Policy Rule	5
5.2. Event Clause	7
5.3. Condition Clause	8
5.4. Action Clause	10
6. YANG Module	12
6.1. IETF NSF-Facing Interface YANG Data Module	12
7. Security Considerations	46
8. Acknowledgments	46
9. Contributors	47
10. References	47
10.1. Normative References	47
10.2. Informative References	47
Appendix A. Changes from draft-ietf-i2nsf-nsf-facing-interface-dm-01	49
Authors' Addresses	49

1. Introduction

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

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

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

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

- o configuration of I2NSF security policy rule for generic network security function policy
- o configuration of event clause for generic network security function policy
- o configuration of condition clause for generic network security function policy
- o 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]:

- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol.
- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [i2rs-rib-data-model] is as follows:

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

4. 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].

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

5.1. I2NSF Security Policy Rule

The data model for 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 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.dunbar@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 enable.";
}

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 enable.";
    }

    leaf during {
        type uint16;
    }
}
```

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

leaf-list policy-event-clause-aggr {
    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-aggr {
    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-aggr {
    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 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-iptables {
        type string;
        description
            "With the iptables keyword you can check if
            a specific ip option is set. Iptables 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 receive.";
    }

    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.";
  }
}
```



```
    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:

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

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [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

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei
J. Jeong
Sungkyunkwan University
July 2, 2018

Information Model for Consumer-Facing Interface to Security Controller
draft-kumar-i2nsf-client-facing-interface-im-06

Abstract

This document defines an information model for Consumer-Facing interface to Security Controller based on the requirements identified in [I-D.ietf-i2nsf-client-facing-interface-req]. The information model defines various managed objects and relationship among these objects needed to build the interface. The information model is organized based on the "Event-Condition-Event" (ECA) policy model defined by a capability information model for Interface to Network Security Functions (I2NSF) [I-D.ietf-i2nsf-capability].

Status of This Memo

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions Used in this Document	3
3. Information Model for Policy	4
3.1. Event Sub-Model	6
3.1.1. Event-Map-Group	6
3.2. Condition Sub-Model	7
3.3. Action Sub-Model	8
4. Information Model for Multi Tenancy	9
4.1. Policy-Domain	10
4.2. Policy-Tenant	10
4.3. Policy-Role	10
4.4. Policy-User	11
4.5. Policy-Management-Authentication-Method	11
5. Information Model for Policy Endpoint Groups	12
5.1. Tag-Source	13
5.2. User-Group	13
5.3. Device-Group	14
5.4. Application-Group	14
5.5. Location-Group	15
6. Information Model for Threat Prevention	15
6.1. Threat-Feed	16
6.2. Custom-List	16
6.3. Malware-Scan-Group	17
7. Information Model for Telemetry Data	17
7.1. Telemetry-Data	17
7.2. Telemetry-Source	18
7.3. Telemetry-Destination	19
8. Role-Based Access Control (RBAC)	19
9. Security Considerations	20
10. IANA Considerations	21
11. Acknowledgments	21

12. Contributors	21
13. Informative References	21
Appendix A. Changes from draft-kumar-i2nsf-client-facing- interface-im-05	22
Authors' Addresses	22

1. Introduction

Interface to Network Security Functions (I2NSF) defines a Consumer-Facing Interface to deliver high-level security policies to Security Controller [RFC8192][RFC8329] for security enforcement in Network Security Functions (NSFs).

The Consumer-Facing interface would be built using a set of objects, with each object capturing a unique set of information from Security Admin (i.e., I2NSF User [RFC8329]) needed to express a Security Policy. An object may have relationship with various other objects to express a complete set of requirement. An information model captures the managed objects and relationship among these objects. The information model proposed in this document is in accordance with interface requirements as defined in [I-D.ietf-i2nsf-client-facing-interface-req].

An NSF Capability model is proposed in [I-D.ietf-i2nsf-capability] as the basic model for both the NSF-Facing interface and Consumer-Facing interface security policy model of this document. The information model proposed in this document is structured in accordance with the "Event-Condition-Event" (ECA) policy model.

[RFC3444] explains differences between an information and data model. This document use the guidelines in [RFC3444] to define an information model for Consumer-Facing interface in this document. A data model, which represents an implementation of the proposed information model in a specific data representation language, will be defined in a separate document.

Figure 1: High-level-abstraction of Consumer-Facing Interface

2. Conventions Used in this Document

BSS: Business Support System
 CLI: Command Line Interface
 CMDB: Configuration Management Database

Controller: Security Controller or Management System

CRUD: Create, Retrieve, Update, Delete

FW: Firewall

GUI: Graphical User Interface

IDS: Intrusion Detection System

IPS: Intrusion Prevention System

LDAP: Lightweight Directory Access Protocol

NSF: Network Security Function, defined by
[I-D.ietf-i2nsf-terminology]

OSS: Operations Support System

RBAC: Role-Based Access Control

SIEM: Security Information and Event Management

URL: Universal Resource Locator

vNSF: NSF being instantiated on Virtual Machines

3. Information Model for Policy

A Policy object represents a mechanism to express a Security Policy by Security Admin (i.e., I2NSF User) using Consumer-Facing interface toward Security Controller; the policy would be enforced on an NSF. The Policy object SHALL have following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Multi-Tenancy: The multi-tenant environment information in which the policy is applied. The Rules in the Policy can refer to sub-objects (e.g., domain, tenant, role, and user) of it. It can be either a reference to a Multi-Tenancy object defined in another place, or a concrete object. See details in Section 4.

End-Group: This field contains a list of logical entities in the business environment where a Security Policy is to be applied. It can be referenced by the Condition objects in

a Rule, e.g., Source, Destination, Match, etc. It can be either a reference to an End-Group object defined in other place, or a concrete object. See details in Section 5.

Threat-Feed: This field represents threat feed such as Botnet servers, GeoIP, and Malware signature. This information can be referenced by the Rule Action object directly to execute the threat mitigation. See details in Section 6.

Telemetry-Data: This field represents the telemetry collection related information that the Rule Action object can refer to about how to collect the interested telemetry information, for example, what type of telemetry to collect, where the telemetry source is, where to send the telemetry information. See details in Section 7.

Rules: This field contains a list of rules. If the rule does not have a user-defined precedence, then any conflict must be manually resolved.

Owner: This field defines the owner of this policy. Only the owner is authorized to modify the contents of the policy.

A policy is a container of Rules. In order to express a Rule, a Rule must have complete information such as where and when a policy needs to be applied. This is done by defining a set of managed objects and relationship among them. A Policy Rule may be related segmentation, threat mitigation or telemetry data collection from an NSF in the network, which will be specified as the sub-model of the policy model in the subsequent sections.

The rule object SHALL have the following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event: This field includes the information to determine whether the Rule Condition can be evaluated or not. See details in Section 3.1.

Condition: This field contains all the checking conditions to apply to the objective traffic. See details in Section 3.2.

Action: This field identifies the action taken when a rule is matched. There is always an implicit action to drop

traffic if no rule is matched for a traffic type. See details in Section 3.3.

Precedence: This field identifies the precedence assigned to this rule by Security Admin. This is helpful in conflict resolution when two or more rules match a given traffic class.

3.1. Event Sub-Model

The Event Object contains information related to scheduling a Rule. The Rule could be activated based on a time calendar or security event including threat level changes.

Event object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event-Type: This field identifies whether the event of triggering policy enforcement is "ADMIN-ENFORCED", "TIME-ENFORCED" or "EVENT-ENFORCED".

Time-Information: This field contains a time calendar such as "BEGIN-TIME" and "END-TIME" for one time enforcement or recurring time calendar for periodic enforcement.

Event-Map-Group: This field contains security events or threat map in order to determine when a policy needs to be activated. This is a reference to Event-Map-Group defined later.

3.1.1. Event-Map-Group

This object represents an event map containing security events and threat levels used for dynamic policy enforcement. The Event-Map-Group object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Security-Events: This contains a list of security events used for purpose for Security Policy definition.

Threat-Map: This contains a list of threat levels used for purpose for Security Policy definition.

3.2. Condition Sub-Model

This object represents Conditions that Security Admin wants to apply the checking on the traffic in order to determine whether the set of actions in the Rule can be executed or not.

The Condition object SHALL have following information:

Source: This field identifies the source of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry-Source for telemetry collection policy.

Destination: This field identifies the destination of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry-Destination for telemetry collection policy.

Match: This field identifies the match criteria used to evaluate whether the specified action needs to be taken or not. This could be either a Policy-Endpoint-Group identifying an Application set or a set of traffic rules.

Match-Direction: This field identifies whether the match criteria is to be evaluated for both directions or only one direction of the traffic with a default of allowing the other direction for stateful match conditions. This is optional and by default a rule should apply to both directions.

Exception: This field identifies the exception consideration when a rule is evaluated for a given communication. This could be a reference to "Policy-Endpoint-Group" object or set of traffic matching criteria.

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

The variable and value can be source and destination IP address, for example, and they have logical operator in between to check whether they match the condition criteria set by a security admin. For

Example: If condition A AND B is true: THEN execute actions ENDIF
 where A denotes a destination address, and B denotes a blacklisted IP address. The operator AND is the logical AND operation.

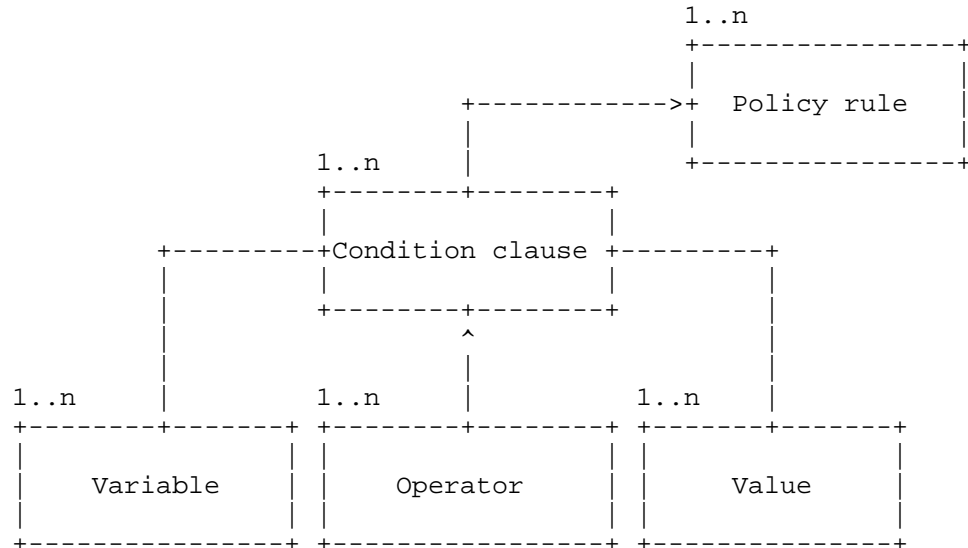


Figure 2: Condition-clause Diagram

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

3.3. Action Sub-Model

This object represents actions that Security Admin wants to perform based on certain traffic class.

The Action object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Primary-Action: This field identifies the action when a rule is matched by an NSF. The action could be one of "PERMIT", "DENY", "DROP-CONNECTION", "AUTHENTICATE-CONNECTION", "MIRROR", "REDIRECT", "NETFLOW", "COUNT", "ENCRYPT", "DECRYPT", "THROTTLE", "MARK", or "INSTANTIATE-NSF".

Secondary-Action: Security Admin can also specify additional actions if a rule is matched. This could be one of "LOG", "SYSLOG", or "SESSION-LOG".

4. Information Model for Multi Tenancy

Multi-tenancy is an important aspect of any application that enables multiple administrative domains in order to manage application resources. An Enterprise organization may have multiple tenants or departments such as Human Resources (HR), Finance, and Legal, with each tenant having a need to manage their own Security Policies. In a Service Provider, a tenant could represent a Customer that wants to manage its own Security Policies.

There are multiple managed objects that constitute multi-tenancy aspects. This section lists these objects and any relationship among these objects. Below diagram shows an example of multi-tenancy in an Enterprise domain.

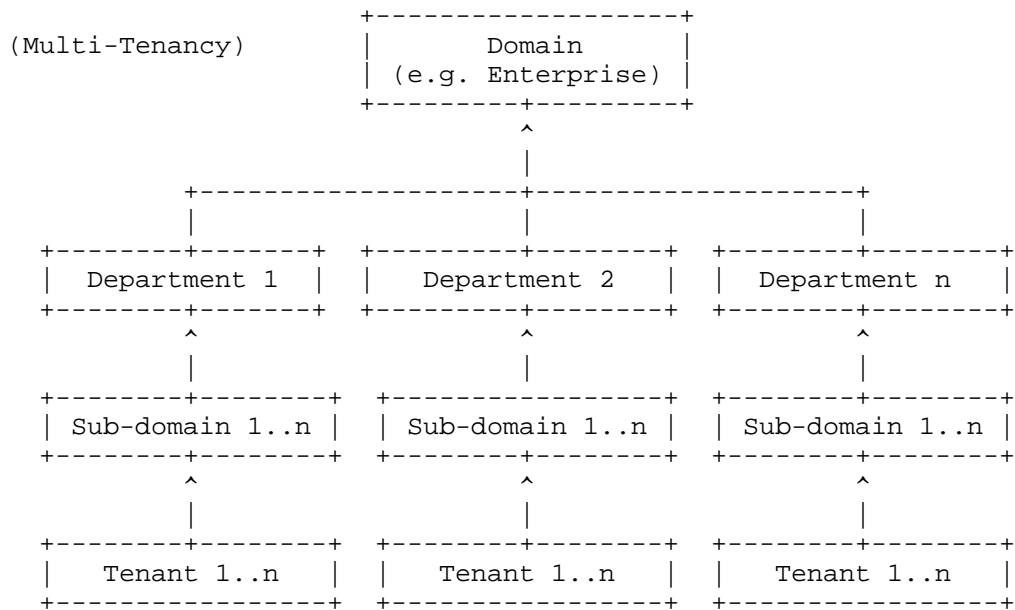


Figure 3: Diagram for Multi-tenancy

4.1. Policy-Domain

This object defines a boundary for the purpose of policy management within a Security Controller. This may vary based on how the Security Controller is deployed and hosted. For example, if an Enterprise hosts a Security Controller in their network; the domain in this case could just be the one that represents that Enterprise. But if a Cloud Service Provider hosts managed services, then a domain could represent a single customer of that Provider. Multi-tenancy model should be able to work in all such environments.

The Policy-Domain object SHALL have following information:

Name: Name of the organization or customer representing this domain.

Address: Address of the organization or customer.

Contact: Contact information of the organization or customer.

Date: Date when this account was created or last modified.

Authentication-Method: Authentication method to be used for this domain. It should be a reference to a 'Policy-Management-Authentication-Method' object.

4.2. Policy-Tenant

This object defines an entity within an organization. The entity could be a department or business unit within an Enterprise organization that would like to manage its own Policies due to regulatory compliance or business reasons.

The Policy-Tenant object SHALL have following information:

Name: Name of the Department or Division within an organization.

Date: Date when this account was created or last modified.

Domain: This field identifies the domain to which this tenant belongs. This should be a reference to a Policy-Domain object.

4.3. Policy-Role

This object defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. It

provides a convenient way to assign policy users to a job function or a set of permissions within the organization.

The Policy-Role object SHALL have the following information:

Name: This field identifies the name of the role.

Date: Date when this role was created or last modified.

Access-Profile: This field identifies the access profile for the role. The profile grants or denies the permissions to access Endpoint Groups for the purpose of policy management or may restrict certain operations related to policy managements.

4.4. Policy-User

This object represents a unique identity within an organization. The identity authenticates with Security Controller using credentials such as a password or token in order to perform policy management. A user may be an individual, system, or application requiring access to Security Controller.

The Policy-User object SHALL have the following information:

Name: Name of a user.

Date: Date when this user was created or last modified.

Password: User password for basic authentication.

Email: E-mail address of the user.

Scope-Type: This field identifies whether the user has domain-wide or tenant-wide privileges.

Scope-Reference: This field should be a reference to either a Policy-Domain or a Policy-Tenant object.

Role: This field should be a reference to a Policy-Role object that defines the specific permissions.

4.5. Policy-Management-Authentication-Method

This object represents authentication schemes supported by Security Controller.

This Policy-Management-Authentication-Method object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Authentication-Method: This field identifies the authentication methods. It could be a password-based, token-based, certificate-based or single sign-on authentication.

Mutual-Authentication: This field indicates whether mutual authentication is mandatory or not.

Token-Server: This field stores the information about server that validates the token submitted as credentials.

Certificate-Server: This field stores the information about server that validates certificates submitted as credentials.

Single Sign-on-Server: This field stores the information about server that validates user credentials.

5. Information Model for Policy Endpoint Groups

The Policy Endpoint Group is a very important part of building User-construct based policies. Security Admin would create and use these objects to represent a logical entity in their business environment, where a Security Policy is to be applied.

There are multiple managed objects that constitute a Policy Endpoint Group. This section lists these objects and relationship among these objects.

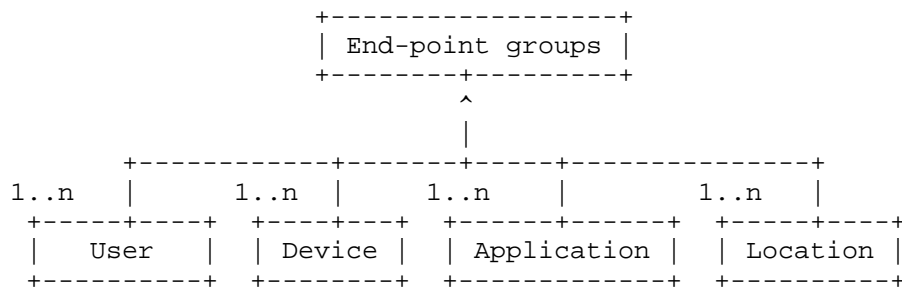


Figure 4: Endpoint

5.1. Tag-Source

This object represents information source for tag. The tag in a group must be mapped to its corresponding contents to enforce a Security Policy.

Tag-Source object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Tag-Type: This field identifies the Endpoint Group type. It can be a User-Group, App-Group, Device-Group or Location-Group.

Tag-Source-Server: This field identifies information related to the source of the tag such as IP address and UDP/TCP port information.

Tag-Source-Application: This field identifies the protocol, e.g., LDAP, Active Directory, or CMDB used to communicate with a server.

Tag-Source-Credentials: This field identifies the credential information needed to access the server.

5.2. User-Group

This object represents a user group based on either tag or other information.

The User-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the user group is based on User-tag, User-name or IP-address.

Metadata-Server: This field should be a reference to a Metadata-Source object.

Group-Member: This field is a list of User-tag, User-names or IP addresses based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.3. Device-Group

This object represents a device group based on either tag or other information.

The Device-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the device group is based on Device-tag or Device-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Device-tag, Device-name or IP address based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.4. Application-Group

This object represents an application group based on either tag or other information.

The Application-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the application group is based on App-tag or App-name, or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Application-tag Application-name or IP address and port information based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.5. Location-Group

This object represents a location group based on either tag or other information.

The 'Location-Group' object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the location group is based on Location-tag, Location-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Location-tag, Location-name or IP addresses based on Group-Type.

Risk Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

6. Information Model for Threat Prevention

The threat prevention plays an important part in the overall security posture by reducing the attack surfaces. This information could come in the form of threat feeds such as Botnet and GeoIP feeds usually from a third party or external service.

There are multiple managed objects that constitute this category. This section lists these objects and relationship among these objects.

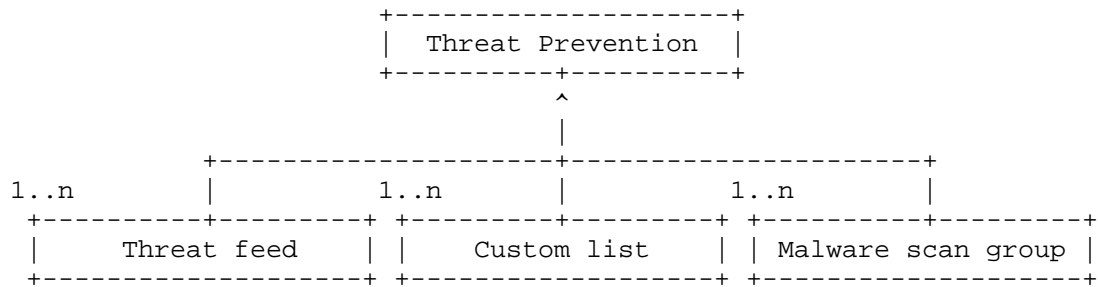


Figure 5: Threat Prevention Diagram

6.1. Threat-Feed

This object represents a threat feed such as Botnet servers and GeoIP.

The Threat-Feed object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Feed-Type: This field identifies whether a feed type is IP address-based or URL-based.

Feed-Server: This field identifies the information about the feed provider, it may be an external service or local server.

Feed-Priority: This field represents the feed priority level to resolve conflict if there are multiple feed sources; the valid range may be 0 to 9.

6.2. Custom-List

This object represents a custom list created for the purpose of defining exception to threat feeds. An organization may want to allow a certain exception to threat feeds obtained from a third party

The Custom-List object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

List-Type: This field identifies whether the list type is IP address-based or URL-based.

List-Property: This field identifies the attributes of the list property, e.g., Blacklist or Whitelist.

List-Content: This field contains contents such as IP addresses or URL names.

6.3. Malware-Scan-Group

This object represents information needed to detect malware. This information could come from a local server or uploaded periodically from a third party.

The Malware-Scan-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Signature-Server: This field contains information about the server from where signatures can be downloaded periodically as updates become available.

File-Types: This field contains a list of file types needed to be scanned for the virus.

Malware-Signatures: This field contains a list of malware signatures or hash values.

7. Information Model for Telemetry Data

Telemetry provides System Admin with the visibility of the network activities which can be tapped for further security analytics, e.g., detecting potential vulnerabilities, malicious activities, etc.

7.1. Telemetry-Data

This object contains information collected for telemetry.

The Telemetry-Data object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Log-Data: This field identifies whether Log data need to be collected.

Syslog-Data: This field identifies whether Syslog data need to be collected.

SNMP-Data: This field identifies whether SNMP traps and alarm data need to be collected.

sFlow-Record: This field identifies whether sFlow records need to be collected.

NetFlow-Record: This field identifies whether NetFlow record need to be collected.

NSF-Stats: This field identifies whether statistics need to be collected from an NSF.

7.2. Telemetry-Source

This object contains information related to telemetry source. The source would be an NSF in the network.

The Telemetry-Source object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Source-Type: This field contains the type of the NSF telemetry source: "NETWORK-NSF", "FIREWALL-NSF", "IDS-NSF", "IPS-NSF", "PROXY-NSF" or "OTHER-NSF".

NSF-Source: This field contains information such as IP address and protocol (UDP or TCP) port number of the NSF providing telemetry data.

NSF-Credentials: This field contains a username and a password used to authenticate the NSF.

Collection-Interval: This field contains time in milliseconds between each data collection. For example, a value of 5,000 means data is streamed to collector every 5 seconds. Value of 0 means data streaming is event-based.

Collection-Method: This field contains a method of collection whether it is PUSH-based or PULL-based.

Heartbeat-Interval: This field contains time in seconds when the source must send telemetry heartbeat.

QoS-Marking: This field contains a DSCP value source marked on its generated telemetry packets.

7.3. Telemetry-Destination

This object contains information related to telemetry destination. The destination is usually a collector which is either a part of Security Controller or external system such as SIEM.

The Telemetry-Destination object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Collector-Source: This field contains the information such as IP address and protocol (UDP or TCP) port number for the collector's destination.

Collector-Credentials: This field contains a username and a password provided by the collector.

Data-Encoding: This field contains the telemetry data encoding, which could in the form of a schema.

Data-Transport: This field contains streaming telemetry data protocols: whether it is gRPC, protocol buffer over UDP, etc.

8. Role-Based Access Control (RBAC)

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

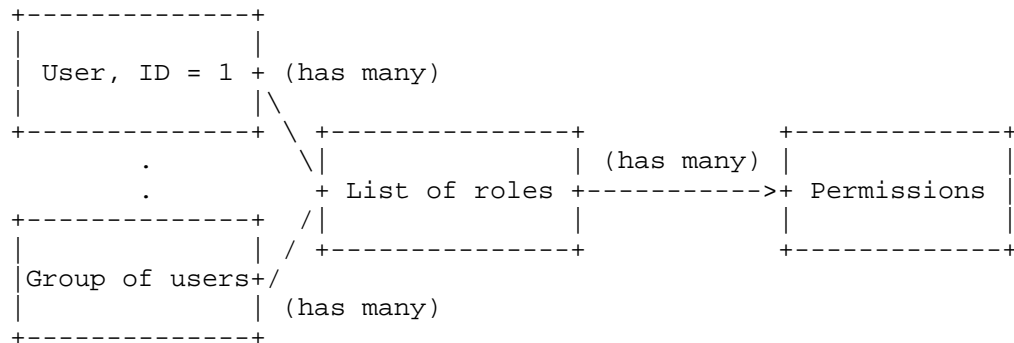


Figure 6: RBAC Diagram

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

Following are the steps required to build RBAC.

1. Defining roles and permissions.
2. Establishing relations among roles and permissions.
3. Defining users.
4. Associating rules with roles and permissions.
5. assigning roles to users.

9. Security Considerations

An information model provides a mechanism to protect Consumer-Facing interface between System Admin (i.e., I2NSF User) and Security Controller. One of the specified mechanism must be used to protect an Enterprise network, data and all resources from external attacks. This information model mandates that the interface must have proper authentication and authorization with Role-Based Access Controls to address the multi-tenancy requirement. The document does not mandate that a particular mechanism should be used because a different organization may have different needs based on their deployment.

10. IANA Considerations

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

11. Acknowledgments

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

12. Contributors

This document is the work of I2NSF working group, greatly benefiting from inputs and suggestions by Kunal Modasiya, Prakash T. Sehsadri and Srinivas Nimmagadda from Juniper Networks. The authors sincerely appreciate their contributions.

The following are contributing authors of this document, who are considered co-authors:

- o Eunsoo Kim (Sungkyunkwan University)
- o Hyounghshick Kim (Sungkyunkwan University)

13. Informative References

[I-D.ietf-i2nsf-capability]

Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-01 (work in progress), April 2018.

[I-D.ietf-i2nsf-client-facing-interface-req]

Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-05 (work in progress), May 2018.

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <<https://www.rfc-editor.org/info/rfc3444>>.
- [RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <<https://www.rfc-editor.org/info/rfc8192>>.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <<https://www.rfc-editor.org/info/rfc8329>>.

Appendix A. Changes from draft-kumar-i2nsf-client-facing-interface-im-05

The following changes have been made from draft-kumar-i2nsf-client-facing-interface-im-05:

- o In Section 3.2, the description and diagram of a condition clause is added.
- o In Section 4, a Multi-tenancy diagram is added.
- o In Section 5, the diagram of a Endpoint group is added.
- o In Section 6, the diagram of a Threat prevention is added.
- o In Section 8, the description and diagram of RBAC is added.
- o References are updated.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rakeshkumarcloud@gmail.com

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Nabil Bitar
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: nabil.bitar@nokia.com

Senad Palislamovic
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: senad.palislamovic@nokia.com

Liang Xia
Huawei
101 Software Avenue
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

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

Phone: +82 31 299 4957

Fax: +82 31 290 7996

Email: pauljeong@skku.edu

URI: <http://iotlab.skku.edu/people-jaehoon-jeong.php>

Interface to Network Security Functions
Internet-Draft
Intended status: Experimental
Expires: January 2, 2019

A. Pastor
D. Lopez
Telefonica I+D
A. Shaw
Hewlett Packard Labs
July 1, 2018

Remote Attestation Procedures for Network Security Functions (NSFs)
through the I2NSF Security Controller
draft-pastor-i2nsf-nsf-remote-attestation-04

Abstract

This document describes the procedures a client can follow to assess the trust on an external NSF platform and its client-defined configuration through the I2NSF Security Controller. The procedure to assess trustworthiness is based on a remote attestation of the platform and the NSFs running on it performed through a Trusted Platform Module (TPM) invoked by the Security Controller.

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 <http://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 January 2, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Establishing Client Trust	4
3.1. First Step: Client-Agnostic Attestation	4
3.2. Second Step: Client-Specific Attestation	5
3.3. Trusted Computing	5
3.4. Topology Attestation	7
4. NSF Attestation Principles	8
4.1. Requirements for a Trusted NSF Platform	9
4.1.1. Trusted Boot	9
4.1.2. Remote Attestation Service	10
4.1.3. Secure Boot	11
5. Remote Attestation Procedures	11
5.1. Trusted Channel with the Security Controller	12
5.2. Security Controller Attestation	14
5.3. Platform Attestation	15
6. Security Considerations	15
7. IANA Considerations	15
8. References	16
8.1. Normative References	16
8.2. Informative References	16
Authors' Addresses	17

1. Introduction

As described in [RFC8192], the use of externally provided NSF implies several additional concerns in security. The most relevant threats associated with a externalized platform are detailed in [I-D.ietf-i2nsf-framework]. As stated there, mutual authentication between the user and the NSF environment and, more importantly, the attestation of the components in this environment by clients, could address these threats and provide an acceptable level of risk. In particular:

- o Client impersonation will be minimized by mutual authentication, and since appropriate records of such authentications will be made available, events are suitable for auditing (as a minimum) in the case of an incident.
- o Attestation of the NSF environment, especially when performed periodically, will allow clients to detect the alteration of the processing components, or the installation of malformed components. Mutual authentication will again provide an audit trail.
- o Attestation relying on independent Trusted Third Parties will alleviate the impact of malicious activity on the side of the provider by issuing the appropriate alarms in the event of any NSF environment manipulation.
- o While it is true that any environment is vulnerable to malicious activity with full physical access (and this is obviously beyond the scope of this document), the application of attestation mechanisms raises the degree of physical control necessary to perform an untraceable malicious modification of the environment.

The client can have a proof that their NSFs and policies are correctly (from the client point of view) enforced by the Security Controller. Taking into account the threats identified in [I-D.ietf-i2nsf-framework], this document first identifies the user expectations regarding remote trust establishment, briefly analyzes Trusted Computing techniques, and finally describes the proposed mechanisms for remote establishment of trust through the Security Controller.

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Establishing Client Trust

From a high-level standpoint, in any I2NSF platform, the client connects and authenticates to the Security Controller, which then initializes the procedures for authentication and authorization (and likely accounting and auditing) to track the loading and unloading of the client's NSFs, addressing the verification of the whole software stack: firmware, (host and guest) Oses, NSFs themselves and, in a virtualized environment, the virtualization system (hypervisors, container frameworks...). Afterwards, user traffic from the client domain goes through the NSF platform that hosts the corresponding NSFs. The user's expectations of the platform behavior are thus twofold:

- o The user traffic will be treated according to the client-specified NSFs, and no other processing will be performed by the Security Controller or the platform itself (e.g. traffic eavesdropping).
- o Each NSF (and its corresponding policies) behaves as configured by the client.

We will refer to the attestation of these two expectations as the "client-agnostic attestation" and the "client-specific attestation". Trusted Computing techniques play a key role in addressing these expectations.

3.1. First Step: Client-Agnostic Attestation

This is the first interaction between a client and a Security Controller: the client wants to attest that he is connected to a genuine Security Controller before continuing with the authentication. In this context, two properties characterize the genuineness of the Security Controller:

1. That the identity of the Security Controller is correct
2. That it will process the client credentials and set up the client NSFs and policies properly.

Once these two properties are proven to the client, the client knows that their credentials will only be used by the Security Controller to set up the execution of their NSFs.

3.2. Second Step: Client-Specific Attestation

From the security enforcement point of view, the client agnostic attestation focuses on the initialization of the execution platform for the NSFs. This second step aims to prove to clients that their security is enforced accordingly with their choices (i.e. NSFs and policies). The attestation can be performed at the initialization of the NSFs, before any user traffic is processed by the NSFs, and optionally during the execution of the NSFs.

Support of static attestation, performed at initialization time, for the execution platform and the NSFs is REQUIRED for a Security Controller managing NSFs, and MUST be performed before any user traffic is redirected through any set of NSFs. The Security Controller MUST provide proof to the client that the instantiated NSFs and policies are the ones chosen.

In addition to the platform and executable component attestation, the infrastructure network topology supporting the NSFs may need to be attested, in order to assess the enforcement of the security policies requested by the client. Whilst platform and NSF attestation can be considered sufficient in I2NSF environments in which network elements are connected following a fairly static configuration, the dynamicity brought by networking techniques such as NFV, SDN and SFC make attestation of dynamic topology network topologies a desirable feature in a number of cases. Depending on the level of assurance desired, the client MAY request the Security Controller proof of the network topology connecting the instantiated NSFs.

Additionally to the NSFs instantiation attestation, a continuous attestation of the Security Controller and the NSF execution MAY be required by a client to ensure their security. The sampling periods for the continuous attestation of NSFs and Controller MAY be different.

3.3. Trusted Computing

In a nutshell, Trusted Computing (TC) aims at answering the following question: "As a user or administrator, how can I have some assurance that a computing system is behaving as it should?". The major enterprise level TC initiative is the Trusted Computing Group [TCG], which has been established for more than a decade, that primarily focuses on developing TC for commodity computers (servers, desktops, laptops, etc.).

The overall scheme proposed by TCG for using Trusted Computing is based on a step-by-step extension of trust, called a Chain of Trust. It uses a transitive mechanism: if a user can trust the first

execution step and each step correctly attests the next executable software for trustworthiness, then a user can trust the system.

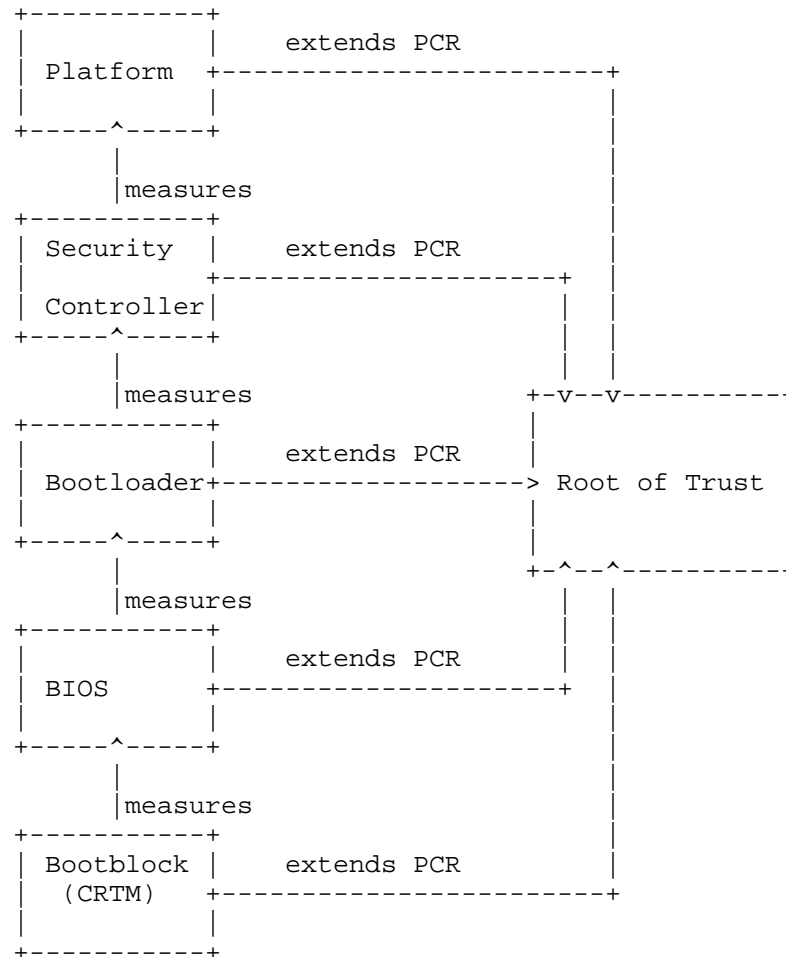


Figure 1: Applying Trusted Computing

Effectively, during the loading of each piece of software, the integrity of each piece of software is measured and stored inside a log that reflects the different boot stages, as illustrated in the figure above. Later, at the request of a user, the platform can present this log (signed with the unique identity of the platform), which can be checked to prove the platform identity and attest the state of the system. The base element for the extension of the Chain of Trust is called the Core Root of Trust.

The TCG has created a standard for the design and usage of a secure crypto-processor to address the storage of keys, general secrets, identities, and platform integrity measurements: the Trusted Platform Module (TPM). When using a TPM as a root of trust, measurements of the software stack are stored in special on-board Platform Configuration Registers (PCRs) on a discrete TPM. There are normally a small number of PCRs that can be used for storing measurements; however, it is not possible to directly write to a PCR. Instead, measurements must be stored using a process called Extending PCRs.

The extend operation can update a PCR by producing a global hash of the concatenated values of the previous PCR value with the new measurement value. The Extend operation allows for an unlimited number of measurements to be captured in a single PCR, since the size of the value is always the same and it retains a verifiable ordered chain of all the previous measurements.

Attestation of the virtualization platform will thus rely on a process of measuring the booted software and storing a chained log of measurements, typically referred to as Trusted Boot. The user will either validate the signed set of measurements with a trusted third party verifier who will assess whether the software configuration is trusted, or the user can check for themselves against their own set of reference digest values (measurements) that they have obtained a priori, and having already known the public endorsement key of the remote Root of Trust.

Trusted Boot should not be confused with a different mechanism known as "Secure Boot", as they both are designed to solve different problems. Secure Boot is a mechanism for a platform owner to lock a platform to only execute particular software. Software components that do not match the configuration digests will not be loaded or executed. This mechanism is particularly useful in preventing malicious software that attempts to install itself in the boot record (a bootkit) from successfully infecting a platform on reboot. A common standard for implementing Secure Boot is described in [UEFI]. Secure Boot only enforces a particular configuration of software, it does not allow a user to attest or quote for a series of measurements.

3.4. Topology Attestation

There are two methods able to attest the deployment of a topology addressing client requirements on a dynamically controlled network infrastructure. The first one assumes the network infrastructure is built by means of SDN-enabled forwarding elements, and the second relies on the application of SFC [RFC7665] to build the NSF processing paths. In both cases, a network topology verifier is

used.

In the first case, a SDN verifier is introduced, and network forwarding elements required to provide attestation features, as described in the previous section, to provide measures on the enforced SDN configuration. The SDN verifier retrieves from the SDN controller the configuration for the attested network elements, challenges them for their SDN configuration, and assesses it is consistent with the expected SDN configuration retrieved from the SDN controller. The SDN verifier on the network elements leverage a TPM, with the network element implementing a regular measured boot.

The second option considers the application of Proof of Transit (POT) [I-D.ietf-sfc-proof-of-transit] to a SFC-based network, where the NSFs act as service functions. A SFC verifier can inject specific packets requesting POT, and verifying it at the egress of the service path to assess a correct topology is being enforced, by means of the cryptographic proof provided by POT.

4. NSF Attestation Principles

Following the general requirements described in [I-D.ietf-i2nsf-framework] the Security Controller will become the essential element to implement the measurements described above, relying on a TPM for the Root of Trust.

A mutual authentication of clients and the Security Controller MUST be performed, establishing the desired level of assurance. This level of assurance will determine how stringent are the requirements for authentication (in both directions), and how detailed the collected measurements and their verification will be. Furthermore, the NSF platform MUST run a TPM, able to collect measurements of the platform itself, the Security Controller, and the NSFs being executed. The availability of a network topology verifier is OPTIONAL, though a client MAY require it to fulfill the required level of assurance. The Security Controller MUST make the attestation measurements available to the client, directly or by means of a Trusted Third Party.

As described in [I-D.ietf-i2nsf-framework], a trusted connection between the client and the Security Controller MUST be established and all traffic to and from the NSF environment MUST flow through this connection

NOTE: The reference to results from WGs such as NEA and SACM is currently under consideration and will be included here.

4.1. Requirements for a Trusted NSF Platform

Although a discrete hardware TPM is RECOMMENDED, relaxed alternatives (such as embedded CPU TPMs, or memory and execution isolation mechanisms) MAY also be applied when the required level of assurance is lower. This reduced level of assurance MUST be communicated to the client by the Security Controller during the initial mutual authentication phase. The Security Controller MUST use a set of capabilities to negotiate the level of assurance with the client.

4.1.1. Trusted Boot

NOTE: This section is derived from the original version of the document, focused on virtual NSFs. Although it seems to be applicable to any modern physical appliance, we must be sure all these considerations are 100% applicable to physical NSFs as well, and provide exceptions when that is not the case. Support from an expert in physical node attestation is required here.

All clients who interact with a Security Controller MUST be able to:

- a. Identify the Security Controller based on the public key of a Root of Trust.
- b. Retrieve a set of measurements of all the base software the Security Controller has booted (i.e. the NSF platform).

This requires that firmware and software MUST be measured before loading, with the resulting value being used to extend the appropriate PCR register. The general usage of PCRs by each software component SHOULD conform to open standards, in order to make verifying attestation reports interoperable, as it is the case of TCG Generic Server Specification [TCGGSS].

The following list describes which PCR registers SHOULD be used during a Trusted Boot process:

- o PCRs 00-03: for use by the CRTM (Initial EEPROM or PC BIOS)
- o PCRs 04-07: for use by the bootloader stages
- o PCRs 08-15: for use by the booted base system

A signed audit log of boot measurements should also be provided. The PCR values can also be used as an identity for dynamically decrypting encrypted blobs on the platform (such as encryption keys or configurations that belong to operating system components). Software can choose to submit pieces of data to be encrypted by the Root of

Trust (which has its own private asymmetric key and PCR registers) and only have it decrypted based on some criteria. These criteria can be that the platform booted into a particular state (e.g. a set of PCR values). Once the desired criteria are described and the sensitive data is encrypted by the root of trust, the data has been sealed to that platform state. The sealed data will only be decrypted when the platform measurements held in the root of trust match the particular state.

Trusted Boot requires the use of a root of trust for safely storing measurements and secrets. Since the Root of Trust is self-contained and isolated from all the software that is measured, it is able to produce a signed set of platform measurements to a local or remote user. However, Trusted Boot does not provide enforcement of a configuration, since the root of trust is a passive component not in the execution path, and is solely used for safe independent storage of secrets and platform measurements. It will respond to attestation requests with the exact measurements that were made during the software boot process. Sealing and unsealing of sensitive data is also a strong advantage of Trusted Boot, since it prevents leakage of secrets in the event of an untrusted software configuration.

4.1.2. Remote Attestation Service

A service **MUST** be present for providing signed attestation report (e.g. the measurements) from the Root of Trust (RoT) to the client. In case of failure to communicate with the service, the client **MUST** assume the service cannot be trusted and seek an alternative Security Controller.

Since some forms of RoT require serialised access (i.e. due to slow access to hardware), latency of getting an attestation report could increase with simultaneous requests. Simultaneous requests could occur if multiple Trusted Third Parties (TTP) request attestation reports at the same time. This **MAY** be improved through batching of requests, in a special manner. In a typical remote attestation protocol, the client sends a random number ("nonce") to the RoT in order to detect any replay attacks. Therefore, caching of an attestation report does not work, since there is the possibility that it may not be a fresh report. The solution is to batch the nonce for each requestor until the RoT is ready for creating the attestation report. The report will be signed by the embedded identity of the RoT to provide data integrity and authenticity, and the report will include all the nonces of the requestors. Regardless of the number of the number of nonces included, the requestor verifying the attestation report **MUST** check to see if the requestor's nonce was included in order to detect replay attacks. In addition to the attestation report containing PCRs, an additional report known as an

SML (Secure Measurement Log) can be returned to the requestor to provide more information on how to verify the report (e.g. how to reproduce the PCR values). The integrity of the SML is protected by a PCR measurement in the RoT. An example of an open standard for responses is [TCGIRSS]. Further details are discussed in Section 5.2.

As part of initial contact, the Security Controller MAY present a list of external TTPs that the client can use to verify it. However, the client MUST assess whether these external verifiers can be trusted. The client can also choose to ignore or discard the presented verifiers.

If available, the network topology verifier MUST be colocated or integrated with the RoT.

Finally, to prevent malicious relaying of attestation reports from a different host, the authentication material of the secure channel (e.g. TLS, IPSec, etc.) SHOULD be bound to the RoT and verified by the connected client, unless the lowest levels of assurance have been chosen and an explicit warning issued. This is also addressed in Section 5.1.

4.1.3. Secure Boot

Using a mechanism such as Secure Boot helps provide strong prevention of software attacks. Furthermore, in combination with a hardware-based TPM, Secure Boot can provide some resilience to physical attacks (e.g. preventing a class of offline attacks and unauthorized system replacement). For NSF providers, it is RECOMMENDED that Secure Boot is employed wherever possible with an appropriate firmware update mechanism, due to the possible threat of software/firmware modifications in either public places or privately with inside attackers.

5. Remote Attestation Procedures

The establishment of trust with the Security Controller and the NSF platform consists of three main phases, which need to be coordinated by the client:

1. Trusted channel with the Security Controller. During this phase, the client securely connects to the Security Controller to avoid that any data can be tampered with or modified by an attacker if the network cannot be considered trusted. The establishment of the trusted channel is completed after the next step.

2. Security Controller attestation. During this phase, the client verifies that the Security Controller components responsible for handling the credentials and for the isolation with respect to other potential clients are behaving correctly. Furthermore, it is verified that the identity of the platform attested is the same of the one presented by the Security Controller during the establishment of the secure connection.
3. Platform attestation. During this step, which can be repeated periodically until the connection is terminated, the Security Controller verifies the integrity of the elements composing the NSF platform. The components responsible for this task have been already attested during the previous phase.

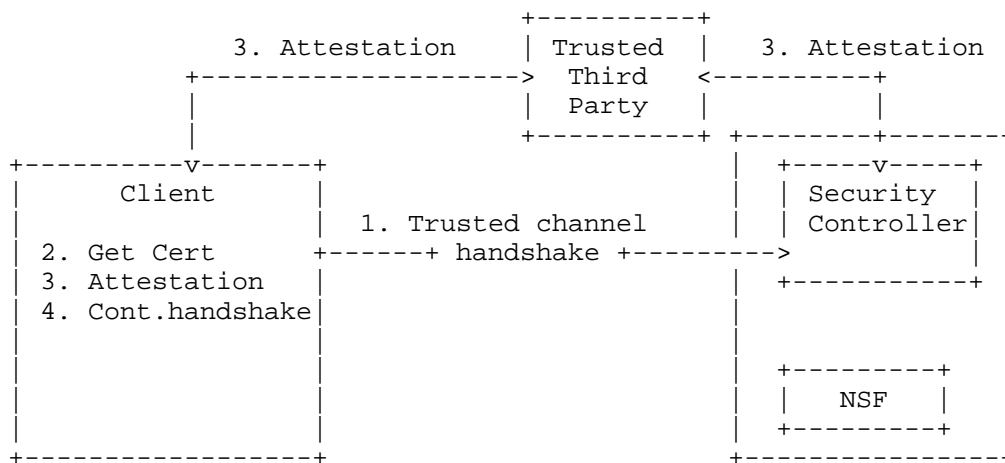


Figure 2: Steps for remote attestation

In the following each step, as depicted in the above figure, is discussed in more detail.

5.1. Trusted Channel with the Security Controller

A trusted channel is an enhanced version of the secured channel that. It adds the requirement of integrity verification of the contacted endpoint by the other peer during the initial handshake to the functionality of the secured channel. However, simply transmitting the integrity measurements over the channel does not guarantee that the platform verified is the channel endpoint. The public key or the

certificate for the secure communication MUST be included as part of the measurements presented by the contacted endpoint during the remote attestation. This way, a malicious platform cannot relay the attestation to another platform as its certificate will not be present in the measurements list of the genuine platform.

In addition, the problem of a potential loss of control of the private key must be addressed (a malicious endpoint could prove the identity of the genuine endpoint). This is done by defining a long-lived Platform Property Certificate. Since this certificate connects the platform identity to the AIK public key, an attacker cannot use a stolen private key without revealing his identity, as it may use the certificate of the genuine endpoint but cannot create a quote with the AIK of the other platform.

Finally, since the platform identity can be verified from the Platform Property Certificate, the information in the certificate to be presented during the establishment of a secure communication is redundant. This allows for the use of self-signed certificates. This would simplify operational procedures in many environments, especially when they are multi-tenant. Thus, in place of certificates signed by trusted CAs, the use of self-signed certificates (which still need to be included in the measurements list) is RECOMMENDED.

The steps required for the establishment of a trusted channel with the Security Controller are as follows:

1. The client begins the trusted channel handshake with the selected Security Controller.
2. The certificate of the Security Controller is collected and used for verifying the binding of the attestation result to the contacted endpoint.
3. The client performs the remote attestation protocol with the Security Controller, either directly or with the help of a Trusted Third Party. The Trusted Third Party MAY perform the verification of attestation quotes on behalf of multiple clients.
4. If the result of the attestation is positive, the application continues the handshake and establishes the trusted channel. Otherwise, it closes the connection.

5.2. Security Controller Attestation

During the establishment of the trusted channel, the client attests the Security Controller by verifying the identity of the contacted endpoint and its integrity. Initially the Security Controller measures all of the hardware and software components involved in the boot process of the NSF platform, in order to build the chain of trust.

Since a client may not have enough functionality to perform the integrity verification of a Security Controller, the client MAY request the status of a Security Controller to be computed by a Trusted Third Party (TTP). This choice has the additional advantage of preventing an attacker from easily determining the software running at the Security Controller.

If the client directly performs the remote attestation, it executes the following steps:

1. Ask the Security Controller to generate an integrity report with the format defined in [TCGIRSS].
2. The Security Controller retrieves the measurements and asks the TPM to sign the PCRs with an Attestation Identity Key (AIK). This signature provides the client with the evidence that the measurements received belong to the Security Controller being attested.
3. Once the integrity report has been generated it is sent back to the client.
4. The client first checks if the integrity report is valid by verifying the quote and the certificate associated to the AIK, and then determines if the Security Controller is behaving as expected (i.e. its software has not been compromised and isolation among the clients connected to it is enforced). As part of the verification, the client also checks that the digest of the certificate, received during the trusted channel handshake, is present among measurements.

If the client has limited computation resources, it may contact a TTP which, in turn, attests the Security Controller and returns the result of the integrity evaluation to the client, following the same steps depicted above.

5.3. Platform Attestation

The main outcome of the Security Controller attestation is to detect whether or not it is correctly configuring the operational environment for NSFs to be managed by the connecting client (the NSF platform, or just platform) in a way that any user traffic is processed only by these NSFs that are part of the platform. Platform attestation, instead, evaluates the integrity of the NSFs running on the platform.

Platform attestation does not imply a validation of the mechanisms the Security Controller can apply to select the appropriate NSFs to enforce the Service Policies applicable to specific flows. The selection of these NSFs is supposed to happen independent of the attestation procedures, and trust on the selection process and the translation of policies into function capabilities has to be based on the trust clients have on the Security Controller being attested as the one that was intended to be used. An attestation of the selection and policy mapping procedures constitute an interesting research problem, but it is out of the scope of this document.

The procedures are essentially similar to the ones described in the previous section. This step MAY be applied periodically if the level of assurance selected by the user requires it.

Attesting NSFs, especially if they are running as virtual machines, can become a costly operation, especially if periodic monitoring is required by the requested level of assurance. There are several proposals to make this feasible, from the proposal of virtual TPMs in [VTPM] to the application of Virtual Machine Introspection through an integrity monitor described by [VMIA].

6. Security Considerations

This document is specifically oriented to security and it is considered along the whole text.

7. IANA Considerations

This document requires no IANA actions.

8. References

8.1. Normative References

- [I-D.ietf-i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-10 (work in progress), November 2017.
- [I-D.ietf-sfc-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof of Transit", draft-ietf-sfc-proof-of-transit-00 (work in progress), May 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015,
<<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017,
<<https://www.rfc-editor.org/info/rfc8192>>.
- [TCG] "Trusted Computing Group (TCG)",
<<https://www.trustedcomputinggroup.org/>>.
- [TCGGSS] "TCG Generic Server Specification, Version 1.0",
<<http://www.trustedcomputinggroup.org/>>.
- [TCGIRSS] "Infrastructure Work Group Integrity Report Schema Specification, Version 1.0",
<<https://www.trustedcomputinggroup.org/>>.

8.2. Informative References

- [UEFI] "UEFI Specification Version 2.2 (Errata D), Tech. Rep.".
- [VMIA] Schiffman, J., Vijayakumar, H., and T. Jaeger, "Verifying System Integrity by Proxy",
<<http://dl.acm.org/citation.cfm?id=2368379>>.

[VTPM] "vTPM:Virtualizing the Trusted Platform Module", <<https://www.usenix.org/legacy/events/sec06/tech/berger.html>>.

Authors' Addresses

Antonio Pastor
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain

Phone: +34 913 128 778
Email: antonio.pastorperales@telefonica.com

Diego R. Lopez
Telefonica I+D
Editor Jose Manuel Lara, 9 (1-B)
Seville, 41013
Spain

Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Adrian L. Shaw
Hewlett Packard Labs
Long Down Avenue
Bristol, BS34 8QZ
UK

Phone: +44 117 316 2877
Email: als@hpe.com

Interface to Network Security Functions (I2NSF)
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2019

L. Xia
Q. Lin
Huawei
October 21, 2018

I2NSF Security Policy Object YANG Data Model
draft-xia-i2nsf-sec-object-dm-01

Abstract

This document describes a set of policy objects which are reusable and can be referenced by variable I2NSF policy rules. And the YANG data models of these policy objects are provided.

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 April 24, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
4. Tree Diagrams	3
5. Policy Object	4
5.1. Address Object and Address Group	4
5.2. Service Object and Service Group	5
5.3. Application Object and Application Group	7
5.4. User Object, User Group and Security Group	9
5.5. Time Range Object	11
5.6. Region Object and Region Group	11
5.7. Domain Object	12
6. I2NSF Security Policy Object YANG Module	13
7. Acknowledgements	46
8. IANA Considerations	46
9. Security Considerations	46
10. References	46
10.1. Normative References	46
10.2. Informative References	46
Authors' Addresses	47

1. Introduction

As described in [RFC8329], provisioning to NSFs can be standardized by using policy rules, and I2NSF uses Event-Condition-Action (ECA) model to represent policy rules. According to [I-D.ietf-i2nsf-terminology], an I2SNF 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 the set of actions in that I2NSF policy rules can be executed or not. Information Model of NSFs Capabilities [I-D.ietf-i2nsf-capability] describes attributes of different condition subclasses. When configuring I2NSF condition clause by attributes or features, it is common to see that the same value of an attribute or the same value set of several attributes are configured for many times. And modifications of the policy rules are also very tedious and time-consuming.

To facilitate the provisioning of NSF instances, this document describes a set of policy objects which are reusable. These policy objects can then be referenced in the condition clause of variable I2NSF policy rules. A policy object consists of a name attribute that identifies itself and one or several attributes that are typically used together to represent a certain condition. For example, protocol type and port number are usually used together to represent a certain service. Each policy object should be predefined

and named in order to be used in I2NSF policy rules. By defining policy objects, the creation and maintenance of policy rules are greatly simplified.

- o A policy object can be referenced in different policy rules as required to provide re-usability. And a policy rule can reference several policy objects.
- o The modification of a policy object will be propagated to the I2NSF policy rules that reference this object. No modification should be made to the related policy rules.

According to [I-D.ietf-i2nsf-terminology], there are two kinds of I2NSF policy rules, I2NSF Directly Consumable Policy Rule and I2NSF Indirectly Consumable Policy Rule. The former one can be executed by a network device without translating its content or structure, while the latter one can not be executed by a network device without first translating its content or structure. In this document, policy objects are defined for I2NSF directly consumable policy rules.

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 terms defined in [I-D.ietf-i2nsf-terminology] and [RFC7950].

4. Tree Diagrams

Tree diagram defined in [RFC8340] is used to represent the policy objects defined in this document. The meaning of the symbols used in the tree diagrams of following sections and the syntax are as follows:

- o Groupings, offset by 2 spaces, and identified by the keyword "grouping" followed by the name of the grouping and a colon (":") character.
- o Each node in the tree is prefaced with "+--". Schema nodes that are children of another node are offset from the parent by 3 spaces.
- o Brackets "[" and "]" enclose list keys.

- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" means state data (read-only), and "-u" indicates the use of a predefined grouping.
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Curly brackets and a question mark "{...}?" are combined to represent the features that node depends on.

5. Policy Object

This document defines policy objects that are commonly used. Figure 1 shows all the defined policy objects and their relationships.

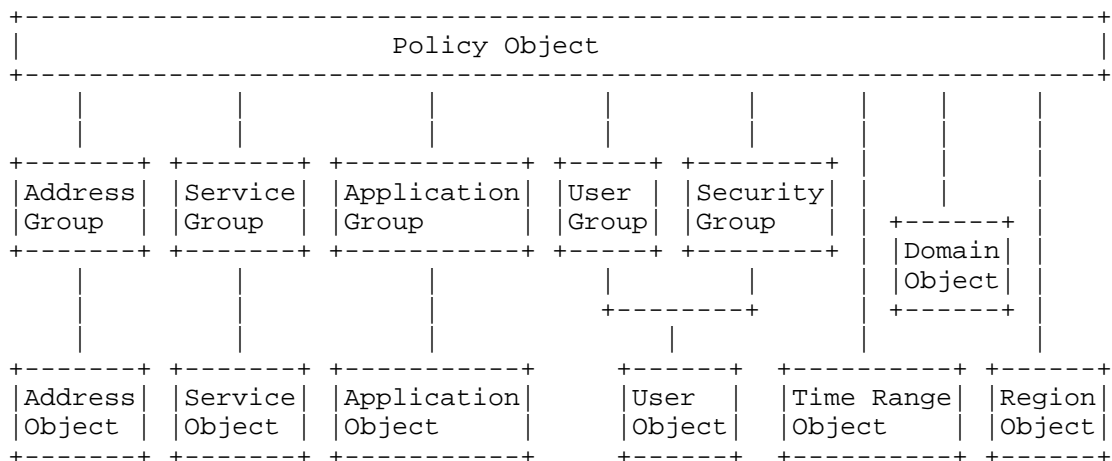


Figure 1: The Policy Objects Overview

5.1. Address Object and Address Group

An address object is identified by a unique name, which contains a set of IPv4/IPv6 addresses or MAC addresses. Several address objects can be organized into an address group object.

This document defines groupings for address objects and address groups.

The tree diagram of address object is:

grouping address-objects:

```

+--rw address-object* [name]
  +--rw name                address-set-name
  +--rw desc?               string
  +--rw vpn-instance?       string
  +--rw elements* [elem-id]
    +--rw elem-id           uint16
    +--rw (object-items)
      +--:(ipv4)
        | +--rw address-ipv4  inet:ipv4-prefix
      +--:(ipv6)
        | +--rw address-ipv6  inet:ipv6-prefix
      +--:(mac)
        | +--rw mac-address    yang:mac-address
        | +--rw mac-address-mask yang:mac-address
      +--:(ipv4-range)
        | +--rw start-ipv4     inet:ipv4-address
        | +--rw end-ipv4       inet:ipv4-address
      +--:(ipv6-range)
        | +--rw start-ipv6     inet:ipv6-address
        | +--rw end-ipv6       inet:ipv6-address

```

The tree diagram of address group is:

grouping address-groups:

```

+--rw address-group* [name]
  +--rw name                address-set-name
  +--rw desc?               string
  +--rw vpn-instance        string
  +--rw elements* [elem-id]
    +--rw elem-id           uint16
    +--rw addr-object-name   address-set-name

```

5.2. Service Object and Service Group

A service object is a kind of service based on IP, or ICMP, or UDP, or TCP, or SCTP. Several related objects consist a service group. To identify different kinds of services, different kinds of attributes should be specified.

- o UDP, TCP, or SCTP based service is recognized by port number. The source port number and destination port number are used to identify the sending and receiving service respectively.
- o ICMP or ICMPv6 based service is recognized by two header fields in the ICMP/ICMPv6 packets: type field and code field.

- o IP based service is recognized by the value of the protocol field in IP packet header.

Besides, a set of well-known services should be predefined by NSFs as service objects to support direct usage.

The tree diagram of service object is:

grouping service-objects:

```

+--ro pre-defined-service* [name]
|   +--ro name                string
|   +--ro session-aging-time  uint16
+--rw service-object* [name]
    +--rw name                 service-set-name
    +--rw session-aging-time   uint16
    +--rw desc?                string
    +--rw items* [id]
        +--rw id               uint16
        +--rw (item)
            +--:(tcp-item)
            |   +--rw tcp
            |   +---u port-items
            +--:(udp-item)
            |   +--rw udp
            |   +---u port-items
            +--:(sctp-item)
            |   +--rw sctp
            |   +---u port-items
            +--:(icmp-item)
            |   +--rw (icmp-type)
            |   |   +--:(name-type)
            |   |   |   +--rw icmp-name                icmp-name-type
            |   |   +--:(type-code)
            |   |   |   +--rw icmp-type-code
            |   |   |   |   +--rw icmp-type-number      uint8
            |   |   |   |   +--rw icmp-code-number      string
            |   +--:(icmp6-item)
            |   |   +--rw (icmp6)
            |   |   |   +--:(name-type)
            |   |   |   |   +--rw icmp6-name            icmp6-name-type
            |   |   |   +--:(type-code)
            |   |   |   |   +--rw icmp6-type-code
            |   |   |   |   |   +--rw icmp-type-number    uint8
            |   |   |   |   |   +--rw icmp-code-number    string
            |   +--:(protocol-id)
            |   |   +--rw proto-id                proto-id-range

```

The "port-items" grouping reuses "port-range-or-operator" grouping defined in [I-D.ietf-netmod-acl-model].

```
grouping port-items:
  +--rw source-port
  |   +---u pf:port-range-or-operator
  +--rw dest-port
      +---u pf:port-range-or-operator
```

The tree diagram of service group is:

```
grouping service-groups:
  +--rw service-group* [name]
      +--rw name                service-set-name
      +--rw desc?               string
      +--rw items* [id]
          +--rw id              uint16
          +--rw service-set-name service-set-name
```

5.3. Application Object and Application Group

Due to the diversity and large amount of applications, it is not able to identify a certain application based on protocol type and port number. For example, there are many web applications with different risk levels run on ports 80 and 443 using HTTP and HTTPS, such as web gaming application and web chat application. Protocol type and port number could not distinguish applications using the same application protocol. In this document, category, subcategory, data transmission model, and risk level are used to describe an application. A set of well-known application objects should be predefined in NSFs to support direct reference.

The tree diagram of application object is:


```

grouping application-objects:
  +--rw user-defined-application {user-defined-application}?
  |   +--rw application* [name]
  |   |   +--rw name                string
  |   |   +--rw label*              string
  |   |   +--rw data-model?         string
  |   |   +--rw category?           string
  |   |   +--rw subcategory?        string
  |   |   +--ro risk-value?         uint32
  |   |   +--rw desc?              string
  |   |   +--rw rule* [name]
  |   |   |   +--rw name                string
  |   |   |   +--rw protocol?          protocol
  |   |   |   +--rw signature
  |   |   |   |   +--rw mode?          mode
  |   |   |   |   +--rw direction?    direction
  |   |   |   |   +--rw pattern-type? pattern-type
  |   |   |   |   +--rw pattern?      string
  |   |   |   |   +--rw field?        identityref
  |   |   +--rw ip-address*         inet:ip-prefix
  |   |   +--rw port*               inet:port-number
  |   |   +--rw desc?              string
  |   +--ro predefined-application
  |   |   +--ro application* [name]
  |   |   |   +--ro name                string
  |   |   |   +--ro protocol*           string
  |   |   |   +--ro risk-value?         uint32
  |   |   |   +--ro label*              string
  |   |   |   +--ro abandon?            boolean
  |   |   |   +--ro multichannel?       boolean
  |   |   |   +--ro data-model?         string
  |   |   |   +--ro category?           string
  |   |   |   +--ro subcategory?        string
  |   |   |   +--ro desc?              string

```

The tree diagram of application group is:

```

grouping application-groups:
  +--rw application-group* [name]
  |   +--rw name                string
  |   +--rw desc?              string
  |   +--rw items* [id]
  |   |   +--rw id              uint16
  |   |   +--rw application-object-name string

```

5.4. User Object, User Group and Security Group

A user object identifies a person who may access network resources. It is the basis of implementing user-based policy control. The user objects may be created locally on the NSFs, or be imported from third parties, such as authentication servers. User objects that require the same policy enforcement are grouped as user group objects or security group objects. The user group objects are organized as a hierarchical structure. A security group object consists of user objects from different user group objects that require the same policy enforcement.

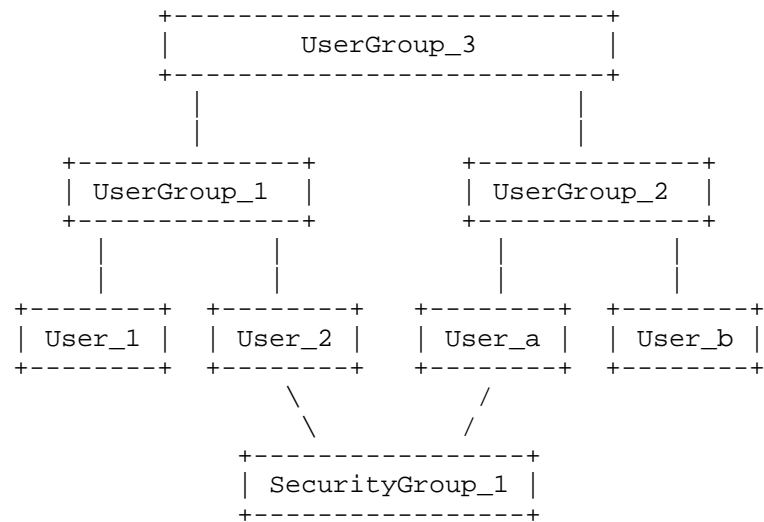


Figure 2: Example of User, User Group and Security Group Structure

The tree diagram of user object is:

```

grouping user-objects:
  +--rw user-object* [name aaa-domain]
    +--rw name          user-name
    +--rw aaa-domain    string
    +--rw desc?         string
    +--rw password?     ianach:crypt-hash
    +--rw parent-user-group      user-group-name
    +--rw parent-security-group  user-security-group-name
    +--rw expiration-time
      +--:(expiration-type)
      |   +--rw (never-expire)
      |   |   +--rw never-expire
      |   +--rw (expire-after-this-time)
      |   |   +--rw expiration-time  yang:date-and-time
    +--rw ip-mac-binding
      +--:(bind-state)
      |   +--rw (no-binding)
      |   |   +--rw no-binding
      |   +--rw (binding)
      |   |   +--rw bind-mode          ip-mac-binding-type
      |   |   +--rw ip-binding*       inet:ipv4-address
      |   |   +--rw mac-binding*      yang:mac-address
      |   |   +--rw ip-mac-bindings [ip-binding]
      |   |   |   +--rw ip-binding    inet:ipv4-address
      |   |   |   +--rw mac-binding   yang:mac-address;

```

The tree diagram of user group is:

```

grouping user-groups:
  +--rw user-group* [name]
    +--rw name          user-group-name
    +--rw desc?         string
    +--rw parent-user-group  user-group-name

```

The tree diagram of security group is:

```

grouping security-groups:
  +--rw security-group* [name]
    +--rw name          user-security-group-name
    +--rw desc?         string
    +--rw parent-security-group*? user-security-group-name
    +--rw filter-action
      +--:(filter-type)
      |   +--rw (static)
      |   |   +--rw static
      |   +--rw (dynamic)
      |   |   +--rw dynamic
      |   +--rw filter-rule*  string

```

5.5. Time Range Object

There are two kinds of time ranges: periodic time range and absolute time range. A periodic time range occurs every week. An absolute time range occurs only once.

The tree diagram of time range object is:

```
grouping time-range-objects:
  +--rw time-range-object* [name]
    +--rw name                time-range-name
    +--rw period-time* [start end]
      |   +--rw start          hour-minute-second
      |   +--rw end            hour-minute-second
      |   +--rw weekday        weekday
    +--rw absolute-time* [start end]
      +--rw start              yang:date-and-time
      +--rw end                yang:date-and-time
```

5.6. Region Object and Region Group

A region object is a set of public IP addresses that are assigned to a certain geographic location. A region group consists of a set of region objects.

The tree diagram of region object is:

```

grouping region-objects:
  +--ro pre-defined-region* [name]
  |   +--ro name                region-name
  |   +--ro desc?               string
  |   +--ro region-ipv4-address
  |   |   +--ro address-ipv4*   inet:ipv4-prefix
  |   |   +--ro address-ipv4-range* [start-ipv4 end-ipv4]
  |   |   |   +--ro start-ipv4   inet:ipv4-address
  |   |   |   +--ro end-ipv4     inet:ipv4-address
  |   +--ro region-ipv6-address {support-ipv6-address}?
  |   |   +--ro address-ipv6*   inet:ipv6-prefix
  |   |   +--ro address-ipv6-range* [start-ipv6 end-ipv6]
  |   |   |   +--ro start-ipv6   inet:ipv6-address
  |   |   |   +--ro end-ipv6     inet:ipv6-address
  +--rw user-defined-region* [name]
  |   +--rw name                region-name
  |   +--rw desc?               string
  |   +--rw coordinate
  |   |   +--rw longitude        region-longitude
  |   |   +--rw latitude         region-latitude
  +--rw region-ipv4-address
  |   +--rw address-ipv4*       inet:ipv4-prefix
  |   +--rw address-ipv4-range* [start-ipv4 end-ipv4]
  |   |   +--rw start-ipv4       inet:ipv4-address
  |   |   +--rw end-ipv4         inet:ipv4-address
  +--rw region-ipv6-address {support-ipv6-address}?
  |   +--rw address-ipv6*       inet:ipv6-prefix
  |   +--rw address-ipv6-range* [start-ipv6 end-ipv6]
  |   |   +--rw start-ipv6       inet:ipv6-address
  |   |   +--rw end-ipv6         inet:ipv6-address

```

The tree diagram of region group is:

```

grouping region-groups:
  +--rw region-group* [name]
  |   +--rw name                region-name
  |   +--rw desc?               string
  |   +--rw region-name*        region-name
  +--rw region-group-name*      region-name

```

5.7. Domain Object

The tree diagram of domain object is:

```
grouping domain-objects:
  +--rw domain-object*  [name]
    +--rw name           domain-name
    +--rw desc?          string
    +--rw domain*        string
```

6. I2NSF Security Policy Object YANG Module

```
<CODE BEGINS> file "ietf-policy-object@2018-10-12.yang"
module ietf-policy-object {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-policy-object";
  prefix policy-object;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC7317 - A YANG Data Model for System Management.";
  }

  import ietf-packet-fields {
    prefix pf;
    reference
      "draft-ietf-netmod-acl-model - Network Access Control List (ACL) YANG Data
Model.";
  }

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

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

    Editor: Liang Xia
           frank.xialiang@huawei.com
    Editor: Qiushi Lin
```

```
        linqiushi@huawei.com";

description
    "This YANG module defines groupings that are used by ietf-policy-object YANG
    module. Their usage is not limited to ietf-policy-object and can be used anywhe
    re as applicable.";

revision 2018-10-12 {
    description "Initial version.";
    reference
        "draft-xia-i2nsf-sec-object-dm-01";
}

/*
 * Typedefs for address object and address group
 */
typedef address-set-name {
    type string {
        length "1..63";
    }
    description
        "This type represents an address object or an address group name.";
}

/*
 * Typedefs for service object and service group
 */
typedef service-set-name {
    type string {
        length "1..63";
    }
    description
        "This type represents a service object or a service group name.";
}

typedef port-range {
    type uint16;
    description
        "This type represents a port number, which may be a start port of a port r
        ange or an end port of a port range.";
}

typedef proto-id-range {
    type uint8 {
        range "0..255";
    }
    description
        "This type represents the range of protocol id.";
}

typedef icmp-name-type {
```

```
type enumeration {
  enum echo {
    description
    "ICMP type number 8, ICMP code number 0";
  }
  enum echo-reply {
    description
    "ICMP type number 0, ICMP code number 0";
  }
  enum fragmentneed-DFset {
    description
    "ICMP type number 3, ICMP code number 4";
  }
  enum host-redirect {
    description
    "ICMP type number 5, ICMP code number 1";
  }
  enum host-tos-redirect {
    description
    "ICMP type number 5, ICMP code number 3";
  }
  enum host-unreachable {
    description
    "ICMP type number 3, ICMP code number 1";
  }
  enum information-reply {
    description
    "ICMP type number 16, ICMP code number 0";
  }
  enum information-request {
    description
    "ICMP type number 15, ICMP code number 0";
  }
  enum net-redirect {
    description
    "ICMP type number 5, ICMP code number 0";
  }
  enum net-tos-redirect {
    description
    "ICMP type number 5, ICMP code number 2";
  }
  enum net-unreachable {
    description
    "ICMP type number 3, ICMP code number 0";
  }
  enum parameter-problem {
    description
    "ICMP type number 12, ICMP code number 0";
  }
}
```



```
    }
    enum port-unreachable {
        description
        "ICMP type number 3, ICMP code number 3";
    }
    enum protocol-unreachable {
        description
        "ICMP type number 3, ICMP code number 2";
    }
    enum reassembly-timeout {
        description
        "ICMP type number 11, ICMP code number 1";
    }
    enum source-quench {
        description
        "ICMP type number 4, ICMP code number 0";
    }
    enum source-soute-failed {
        description
        "ICMP type number 3, ICMP code number 5";
    }
    enum timestamp-reply {
        description
        "ICMP type number 14, ICMP code number 0";
    }
    enum timestamp-request {
        description
        "ICMP type number 13, ICMP code number 0";
    }
    enum ttl-exceeded {
        description
        "ICMP type number 11, ICMP code number 0";
    }
}
description
    "This type is an enumeration of ICMP type names.";
}

typedef icmp6-name-type {
    type enumeration {
        enum redirect {
            description
            "ICMPv6 type number 137, ICMPv6 code number 0";
        }
        enum echo {
            description
            "ICMPv6 type number 128, ICMPv6 code number 0";
        }
    }
}
```

```
enum echo-reply {
    description
    "ICMPv6 type number 129, ICMPv6 code number 0";
}
enum err-Header-field {
    description
    "ICMPv6 type number 4, ICMPv6 code number 0";
}
enum frag-time-exceeded {
    description
    "ICMPv6 type number 3, ICMPv6 code number 1";
}
enum hop-limit-exceeded {
    description
    "ICMPv6 type number 3, ICMPv6 code number 0";
}
enum host-admin-prohib {
    description
    "ICMPv6 type number 1, ICMPv6 code number 1";
}
enum host-unreachable {
    description
    "ICMPv6 type number 1, ICMPv6 code number 3";
}
enum neighbor-advertisement {
    description
    "ICMPv6 type number 136, ICMPv6 code number 0";
}
enum neighbor-solicitation {
    description
    "ICMPv6 type number 135, ICMPv6 code number 0";
}
enum network-unreachable {
    description
    "ICMPv6 type number 1, ICMPv6 code number 0";
}
enum packet-too-big {
    description
    "ICMPv6 type number 2, ICMPv6 code number 0";
}
enum port-unreachable {
    description
    "ICMPv6 type number 1, ICMPv6 code number 4";
}
enum router-advertisement {
    description
    "ICMPv6 type number 134, ICMPv6 code number 0";
}
```

```
enum router-solicitation {
    description
    "ICMPv6 type number 133, ICMPv6 code number 0";
}
enum unknown-ipv6-opt {
    description
    "ICMPv6 type number 4, ICMPv6 code number 2";
}
enum unknown-next-hdr {
    description
    "ICMPv6 type number 4, ICMPv6 code number 1";
}
}
description
    "This type is an enumeration of ICMPv6 type names.";
}

/*
 * Typedefs for application object and application group
 */
typedef protocol {
    type enumeration {
        enum tcp {
            description
            "tcp protocol";
        }
        enum udp {
            description
            "udp protocol";
        }
        enum any {
            description
            "any protocol";
        }
    }
    description
    "The protocol of user-defined application rule:tcp/udp/any.";
}

typedef mode {
    type enumeration {
        enum flow {
            description
            "Keyword exists in multiple packets";
        }
        enum packet{
            description
            "Keyword exists in one packet";
        }
    }
}
```

```

    }
  }
  description
    "The mode of keyword identification to identify user-defined applications.
    If the keyword exists in one packet, the mode is Packet. If the keyword exists
    in multiple packets, the mode is Flow.";
}

typedef direction {
  type enumeration {
    enum request {
      description
        "Request indicates that data to the server is monitored to detect appl
        ications.";
    }
    enum response {
      description
        "Response indicates that data from the server is monitored to detect a
        pplications.";
    }
    enum both {
      description
        "Both indicates that data from and to the server is monitored to detec
        t applications.";
    }
  }
  description
    "The data flow direction that is monitored to identify user-defined applic
    ations:request/response/both. Request indicates that data to the server is monit
    ored to detect applications, Response indicates that data from the server is mon
    itored to detect applications, and Both indicates that data from and to the serv
    er is monitored to detect applications.";
}

typedef pattern-type {
  type enumeration {
    enum regular {
      description
        "Regular indicates that the keyword of the match pattern is not a fixe
        d string, it is represented by regular expression.";
    }
    enum plain {
      description
        "Plain indicates that the keyword of the match pattern is a fixed stri
        ng.";
    }
  }
  description
    "The match pattern of the user-defined application rule. If the keyword is
    a fixed string, the pattern type is Plain. If the keyword is not a fixed string
    , the pattern type is Regular Expression.";
}

/*
 * Typedefs for user object, user group, and security group
 */

typedef user-name {
  type string {
    length "1..63";

```



```
    }
    description
        "This type represents a user name.";
}

typedef user-group-name {
    type string {
        length "1..63";
    }
    description
        "This type represents a user group name.";
}

typedef user-security-group-name {
    type string {
        length "1..63";
    }
    description
        "This type represents a security group name.";
}

typedef ip-mac-binding-type {
    type enumeration {
        enum bidirectional {
            description
                "Bidirectional binding indicates that a user must use the specified IP
                and MAC addresses to log in. The same IP and MAC addresses cannot be used by ot
                her bidirectional binding users.";
        }
        enum unidirectional {
            description
                "Unidirectional binding indicates that a user must use the specified I
                P and MAC addresses to log in. The same IP and MAC addresses can also be used by
                other users.";
        }
    }
    description
        "The user and IP/MAC address binding mode: bidirectional, or unidirectiona
        1. In unidirectional binding, a user must use the specified IP and MAC addresses
        to log in. The same IP and MAC addresses can also be used by other users. In b
        idirectional binding, a user must use the specified IP and MAC addresses to log
        in. The same IP and MAC addresses cannot be used by other bidirectional binding
        users.";
}

/*
 * Typedefs for time range object
 */
typedef time-range-name {
    type string {
        length "1..32";
    }
    description
        "This type represents a time-range name.";
}
```

```
typedef hour-minute-second {
    type string {
        pattern '\d{1,2}:\d{1,2}:\d{1,2}';
    }
    description
        "The representation of Hour, Minute, Sencond - hh:mm:ss";
}

typedef weekday {
    type enumeration {
        enum sunday {
            description
                "Sunday of the week";
        }
        enum monday {
            description
                "Monday of the week";
        }
        enum tuesday {
            description
                "Tuesday of the week";
        }
        enum wednesday {
            description
                "Wednesday of the week";
        }
        enum thursday {
            description
                "Thursday of the week";
        }
        enum friday {
            description
                "Friday of the week";
        }
        enum saturday {
            description
                "Saturday of the week";
        }
    }
    description
        "A type modeling the weekdays in the Greco-Roman tradition.";
}

/*
 * Typedefs for region object and region group
 */
typedef region-name {
```

```
    type string;
    description
        "This type represents a location or location set name.";
}

typedef region-longitude {
    type string;
    description
        "This type represents a region longitude number(-180.00 - 180.00).";
}

typedef region-latitude {
    type string;
    description
        "This type represents a region latitude number(-90.00 - 90.00).";
}

typedef domain-name {
    type string {
        length "1..63";
    }
    description
        "This type represents a domain object name.";
}

/*
 * Identities for application object and application group
 */
identity protocol-field {
    description
        "Base type of protocol field.";
}

identity general-payload {
    base protocol-field;
    description
        "The field of signature is general-payload.";
}

identity http-method {
    base protocol-field;
    description
        "The field of signature is http.method.";
}

identity http-uri {
    base protocol-field;
```



```
    description
      "The field of signature is http.uri.";
  }

  identity http-user-agent {
    base protocol-field;
    description
      "The field of signature is http.user-agent.";
  }

  identity http-host {
    base protocol-field;
    description
      "The field of signature is http.host.";
  }

  identity http-content-type {
    base protocol-field;
    description
      "The field of signature is http.content-type.";
  }

  identity http-cookie {
    base protocol-field;
    description
      "The field of signature is http.cookie.";
  }

  identity http-body {
    base protocol-field;
    description
      "The field of signature is http.body.";
  }

  /*
  * Features for application object
  */
  feature user-defined-application {
    description
      "This feature means the NSF supports user-defined application function tha
t can be used to define application rule.";
  }

  /*
  * Features for region object
  */
  feature support-ipv6-address {
    description
```

```

    "This feature means the NSF support configuring IPv6 addresses for Region
    Object.";
  }

  /*
  * Groupings for address object and address group
  */
  grouping address-objects {
    list address-object {
      key "name";
      leaf name {
        type address-set-name;
        description
          "The name of the address object.";
      }
      leaf desc {
        type string{
          length "1..127";
        }
        description
          "The description of the address object.";
      }
      leaf vpn-instance {
        type string;
        description
          "The name of the vpn-instrance.";
      }
      list elements {
        key "elem-id";
        leaf elem-id {
          type uint16;
          description
            "The id of the element in address object.";
        }
        choice object-items {
          case ipv4 {
            leaf address-ipv4 {
              type inet:ipv4-prefix;
              description
                "A set of IPv4 addresses that are represented by an IPv4 address
                prefix.";
            }
          }
          case ipv6 {
            leaf address-ipv6 {
              type inet:ipv6-prefix;
              description
                "A set of IPv6 addresses that are represented by an IPv6 address
                prefix.";
            }
          }
        }
      }
    }
  }

```

```

        case mac {
            leaf mac-address {
                type yang:mac-address;
                description
                    "MAC address. This leaf is combined with the mac-address-mask le
af to represent a single MAC address or a set of MAC addresses. If the mac-addre
ss-mask leaf is not presented, this leaf represents a single MAC address. If the
mac-address-mask leaf is setted, this leaf represents a range of contiguous MAC
addresses.";
            }
            leaf mac-address-mask {
                type yang:mac-address;
                description
                    "If this leaf is not presented, the mac-address leaf represents
a single MAC address. If this leaf is setted, the mac-address leaf represents a
range of contiguous MAC addresses.";
            }
        }
        case ipv4-range {
            leaf start-ipv4 {
                type inet:ipv4-address;
                description
                    "The start IPv4 address of an IPv4 address range.";
            }
            leaf end-ipv4 {
                type inet:ipv4-address;
                description
                    "The end IPv4 address of an IPv4 address range.";
            }
        }
        case ipv6-range {
            leaf start-ipv6 {
                type inet:ipv6-address;
                description
                    "The start IPv6 address of an IPv6 address range.";
            }
            leaf end-ipv6 {
                type inet:ipv6-address;
                description
                    "The end IPv6 address of an IPv6 address range.";
            }
        }
        description
            "Diffrent types of addresses: IPv4, IPv6, MAC.";
    }
    description
        "A list of addresses that belong to a specific address object.";
}
description
    "A list of address objects.";
}
description
    "This grouping represents a list of address objects. An address object is
identified by a unique name and contains a set of IPv4/IPv6 addresses or MAC add
resses. This grouping reuse the predefined address-object-item grouping.";
}

```

```

grouping address-groups {
  list address-group {
    key "name";
    leaf name {
      type address-set-name;
      description
        "The name of the address group.";
    }
    leaf desc {
      type string{
        length "1..127";
      }
      description
        "The description of the address group.";
    }
    leaf vpn-instance {
      type string;
      description
        "The name of the vpn-instrance.";
    }
    list elements {
      key "elem-id";
      leaf elem-id {
        type uint16;
        description
          "The id of the element in address group.";
      }
      leaf addr-object-name {
        type address-set-name;
        mandatory true;
        description
          "The name of the address object that consists the address group.";
      }
      description
        "A list of address objects that consists the address group object.";
    }
    description
      "A list of address group objects.";
  }
  description
    "An address group object is comprised of several address objects that require the same policy enforcement. This grouping represents a list of address groups.";
}

/*
 * Groupings for service object and service group
 */
grouping port-items {

```

```
    container source-port {
      uses pf:port-range-or-operator;
      description
        "Source port definition from range or operator.";
    }
    container dest-port {
      uses pf:port-range-or-operator;
      description
        "Destination port definition from range or operator.";
    }
    description
      "This grouping consists of the source port numbers and destination port numbers that represent UDP, TCP or SCTP based services.";
  }

  grouping service-objects {
    list pre-defined-service {
      key "name";
      config false;
      leaf name {
        type service-set-name;
        config false;
        description
          "The name of the predefined service object.";
      }
      leaf session-aging-time {
        type uint16;
        units second;
        config false;
        description
          "The aging time of the predefined service object.";
      }
    }
    description
      "A list of the predefined service objects.";
  }
  list service-object {
    key "name";
    leaf name {
      type service-set-name;
      description
        "The name of the service object.";
    }
    leaf session-aging-time {
      type uint16;
      units second;
      description
        "The aging time of the service object.";
    }
  }
```

```

    leaf desc {
      type string{
        length "1..127";
      }
      description
        "The description of the service object.";
    }
    list items {
      key "id";
      leaf id {
        type uint16;
        description
          "The id of the element in service object.";
      }
      choice item {
        case tcp-item {
          container tcp {
            uses port-items;
            description
              "TCP based service is recognized by source port number and destination port number. This container reuse the port-items grouping.";
          }
        }
        case udp-item {
          container udp {
            uses port-items;
            description
              "UDP based service is recognized by source port number and destination port number. This container reuse the port-items grouping.";
          }
        }
        case sctp-item {
          container sctp {
            uses port-items;
            description
              "SCTP based service is recognized by source port number and destination port number. This container reuse the port-items grouping.";
          }
        }
        case icmp-item {
          choice icmp-type {
            case name-type {
              leaf icmp-name {
                type icmp-name-type;
                mandatory true;
                description
                  "The ICMP based service is identified by the predefined ICMP name type.";
              }
            }
            case type-code {
              container icmp-type-code {

```

```

        leaf icmp-type-number {
            type uint8;
            mandatory true;
            description
                "The ICMP type number.";
        }
        leaf icmp-code-number {
            type string;
            mandatory true;
            description
                "The ICMP code number.";
        }
        description
            "The ICMP based service is recognized by two header fields i
n the ICMP packets: type field and code field.";
    }
    description
        "The ICMP based service object and its attributes.";
}
case icmp6-item {
    choice icmp6-type {
        case name-type {
            leaf icmp6-name {
                type icmp6-name-type;
                mandatory true;
                description
                    "The ICMPv6 based service is identified by the predefined IC
MPv6 name type.";
            }
        }
        case type-code {
            container icmp6-type-code {
                leaf icmp6-type-number {
                    type uint8;
                    mandatory true;
                    description
                        "The ICMPv6 type number.";
                }
                leaf icmp6-code-number {
                    type string;
                    mandatory true;
                    description
                        "The ICMP code number.";
                }
            }
            description
                "The ICMPv6 based service is recognized by two header fields
in the ICMPv6 packets: type field and code field.";
        }
    }
}

```

```

        description
        "The ICMPv6 based service object and its attributes.";
    }
    description
    "The ICMPv6 based service object and its attributes.";
}
case protocol-id {
  leaf proto-id {
    type proto-id-range;
    mandatory true;
    description
    "IP based service is identified by the value of the protocol fie
ld in IP packet header.";
  }
}
  description
  "Diffrent types of protocols for service definition.";
}
  description
  "A list of service items that consistan service object.";
}
description
  "A list of user defined service objects.";
}
description
  "A list of the predefined service objects and user defined service objects
.";
}

grouping service-groups {
  list service-group {
    key "name";
    leaf name {
      type service-set-name;
      description
      "The name of the service group.";
    }
    leaf desc {
      type string{
        length "1..127";
      }
      description
      "The description of the service group.";
    }
  }
  list items {
    key "id";
    leaf id {
      type uint16;
      description
      "The id of the element in service group.";
    }
  }
}

```



```

    }
    leaf service-object-name {
        type service-set-name;
        mandatory true;
        description
            "The name of the service object that consists the service group.";
    }
    description
        "A list of service objects that consists the service group object.";
}
description
    "A list of service group objects.";
}
description
    "A service group object is comprised of several service objects that require the same policy enforcement. This grouping represents a list of service groups.";
}

/*
 * Groupings for application object and application group
 */
grouping application-objects {
    container user-defined-application {
        if-feature user-defined-application;
        container applications {
            list application {
                key "name";
                leaf name {
                    type string;
                    description
                        "The name of user-defined application object.";
                }
                leaf-list label {
                    type string;
                    description
                        "A list of labels for user-defined application.";
                }
                leaf data-model {
                    type string;
                    description
                        "The data transmission model of user-defined application. Examples are client/server, peer-to-peer. Data transmission models are predefined in the NSF.";
                }
                leaf category {
                    type string;
                    description
                        "The category of user-defined application. The value of this leaf is selected from a predefined set of categories, e.g., general category, network category.";
                }
                leaf subcategory {

```

```

        type string;
        description
            "The subcategory of user-defined application. ";
    }
    leaf risk-value {
        type uint32;
        config false;
        description
            "The risk value of predefined application.";
    }
    leaf desc {
        type string;
        description
            "The description information of user-defined application.";
    }
    list rule {
        key "name";
        leaf name {
            type string;
            description
                "The name of the user-defined application rule.";
        }
        leaf protocol {
            type protocol;
            description
                "The protocol that user-defined application is based on.";
        }
        container signature {
            leaf mode {
                type string;
                description
                    "The mode of keyword identification. If the keyword exists in
one packet, the mode is Packet. If the keyword exists in multiple packets, the m
ode is Flow.";
            }
            leaf direction {
                type direction;
                description
                    "The traffic direction for application identification. Request
indicates that data to the server is detected, Response indicates that data fro
m the server is detected, and Both indicates that data from and to the server is
detected.";
            }
            leaf pattern-type{
                type pattern-type;
                description
                    "The match pattern of the user-defined application rule. If th
e keyword is a fixed string, the pattern type is Plain. If the keyword is not a
fixed string, the pattern type is Regular Expression.";
            }
            leaf pattern {
                type string;
                description
                    "The keyword of user-defined application rule.";
            }
        }
    }

```

```

        leaf field {
            type identityref {
                base protocol-field;
            }
            default general-payload;
            description
                "The protocol field to search for a signature. The default pro
tolocol field is General-payload.";
        }
        description
            "The signature/characteristics of user-defined application.";
    }
    description
        "The rule used to identify the user-defined application.";
    }
    leaf-list ip-address {
        type inet:ip-prefix;
        description
            "The destination IPv4/IPv6 address of user-defined application.";
    }
    leaf-list port {
        type inet:port-number;
        description
            "The destination port number of user-defined application.";
    }
    description
        "A list of user-defined application objects.";
    }
    description
        "When the NSF supports user-defined application function, these are a
list of user-defined application objects.";
    }
    description
        "When the NSF supports user-defined application function, this container
is used to configure application objects.";
    }
    container predefined-application {
        config false;
        list application {
            key "name";
            leaf name {
                type string;
                config false;
                description
                    "The name of the predefined application.";
            }
            leaf-list protocol {
                type string;
                config false;
                description
                    "The protocol information of application.";
            }
        }
    }

```

```
    }
    leaf risk-value {
        type uint32;
        config false;
        description
            "The risk value of predefined application.";
    }
    leaf-list label {
        type string;
        config false;
        description
            "The label of predefined application,an application may have multiple labels.";
    }
    leaf abandon {
        type boolean;
        config false;
        description
            "The abandon flag of predefined application.";
    }
    leaf multichannel {
        type boolean;
        config false;
        description
            "The multi channel flag of predefined application.";
    }
    leaf data-model {
        type string;
        description
            "The data transmission model of user-defined application. Examples are client/server, peer-to-peer. Data transmission models are predefined in the NISF.";
    }
    leaf category {
        type string;
        config false;
        description
            "The category of user-defined application. The value of this leaf is selected from a predefined set of categories, e.g., general category, network category.";
    }
    leaf subcategory {
        type string;
        config false;
        description
            "The name of application subcategory.";
    }
    leaf desc {
        type string;
        config false;
        description
            "The description information of application.";
    }
}
```

```

        description
            "The attributes of a predefined application.";
    }
    description
        "The information of all predefined applications.";
}
    description
        "A list of predefined application objects.";
}

grouping application-groups {
    list application-group {
        key "name";
        leaf name {
            type string;
            description
                "The name of the application group.";
        }
        leaf desc {
            type string{
                length "1..127";
            }
            description
                "The description of the application group.";
        }
        list items {
            key "id";
            leaf id {
                type uint16;
                description
                    "The id of the element in application group.";
            }
            leaf application-object-name {
                type string;
                mandatory true;
                description
                    "The name of the application object that consists the application gr
oup.";
            }
        }
        description
            "A list of application objects that consist an application group objec
t.";
    }
    description
        "A list of application group objects.";
}
description
    "An application group object is comprised of several application objects t
hat require the same policy enforcement. This grouping represents a list of appl
ication groups.";
}

```

```

/*
 * Groupings for user object, user group and security group
 */
grouping user-objects {
  list user-object {
    key "name aaa-domain";
    leaf name {
      type user-name;
      description
        "The name of the user.";
    }
    leaf aaa-domain {
      type string {
        length "1..64";
      }
      description
        "The name of the domain to which the user belong.";
    }
    leaf desc {
      type string {
        length "1..127";
      }
      description
        "The description of the user.";
    }
    leaf password {
      type ianach:crypt-hash;
      description
        "If user is authenticated locally on the NSF, this attribute is mandatory. It defines the password corresponding to the user name.";
    }
    leaf parent-user-group {
      type user-group-name;
      description
        "The name of the parent group. User objects and user groups are in a hierarchical structure. A user object can only belong to one user group.";
    }
    leaf-list parent-security-group {
      type user-security-group-name;
      max-elements 40;
      description
        "The name of the parent security group. A user object can belong to several security groups.";
    }
    container expiration-time {
      choice expiration-type {
        case never-expire {
          leaf never-expire {
            type empty;
            description
              "This case indicates that the user never expire.";
          }
        }
      }
    }
  }
}

```

```

    }
  }
  case expire-after-this-time {
    leaf expiration-time {
      type yang:date-and-time;
      description
        "User expired time.";
    }
  }
  description
    "Two types of user expiration configurations.";
}
description
  "User expiration time.";
}
container ip-mac-binding {
  choice bind-state {
    case no-binding {
      leaf no-binding{
        type empty;
        mandatory true;
        description
          "No binding: Indicates that a user is not bound to any IP or MAC
address.";
      }
    }
    case binding {
      leaf bind-mode{
        type ip-mac-binding-type;
        description
          "The user and IP/MAC address binding mode: bidirectional, or uni
directional. In unidirectional binding, a user must use the specified IP and MAC
addresses to log in. The same IP and MAC addresses can also be used by other us
ers. In bidirectional binding, a user must use the specified IP and MAC address
es to log in. The same IP and MAC addresses cannot be used by other bidirectiona
l binding users.";
      }
      leaf-list ip-binding {
        type inet:ipv4-address;
        description
          "The IP address bound to the user.";
      }
      leaf-list mac-binding {
        type yang:mac-address;
        description
          "The MAC address bound to the user.";
      }
    }
  }
  list ip-mac-bindings {
    key "ip-binding";
    unique "mac-binding";
    leaf ip-binding {
      type inet:ipv4-address;
      description
        "The bound IPv4 address";
    }
  }
}

```

```

    }
    leaf mac-binding {
        type yang:mac-address;
        description
            "The bound mac address";
    }
    description
        "Configure the IP address and MAC address pairs bound to the use
r.";
    }
    }
    description
        "The binding state: no-binding, binding.";
    }
    description
        "Whether there are IP/MAC addresses bound to the user.";
    }
    description
        "User Object and its attributes.";
    }
    description
        "A list of user objects.";
    }

    grouping security-groups {
        list security-group {
            key "name";
            leaf name {
                type user-security-group-name;
                description
                    "The name of the security-group.";
            }
            leaf desc {
                type string {
                    length "1..127";
                }
                description
                    "The description of the security-group.";
            }
            leaf-list parent-security-group {
                type user-security-group-name;
                max-elements 40;
                description
                    "Configure the name of the parent-security-group.";
            }
            container filter-action {
                choice filter-type {
                    case static {
                        leaf static {

```



```

        type empty;
        mandatory true;
        description
            "Empty leaf indicates that this is a static security group.";
    }
}
case dynamic {
    leaf dynamic {
        type empty;
        mandatory true;
        description
            "Empty leaf indicates that this is a dynamic security group.";
    }
    leaf-list filter-rule {
        type string {
            length "1..256";
        }
        max-elements 5;
        description
            "Filter rules for dynamic security group.";
    }
}
description
    "The filter type: static, dynamic.";
}
description
    "The filter type of the security group, static and dynamic. For dynamic
c security group, an filter rule needs to be configured.";
}
description
    "Security group and its attributes.";
}
description
    "A list of security groups.";
}

grouping user-groups {
    list user-group {
        key "name";
        leaf name {
            type user-group-name;
            description
                "The name of the user group.";
        }
        leaf desc {
            type string {
                length "1..63";
            }
        }
        description

```

```

        "The description of the user group.";
    }
    leaf parent-user-group {
        type user-group-name;
        description
            "The name of the user group. A user group can only belong to one paren
t user group.";
    }
    description
        "User group and its attributes.";
    }
    description
        "A list of user groups";
    }

/*
 * Groupings for time range object
 */
grouping time-range-objects {
    list time-range-object {
        key "name";
        leaf name {
            type time-range-name;
            description
                "The name of the time range object.";
        }
    }
    list period-time {
        key "start end";
        leaf start {
            type hour-minute-second;
            mandatory true;
            description
                "Start time of the periodic time range.";
        }
        leaf end {
            type hour-minute-second;
            mandatory true;
            description
                "End time of the periodic time range.";
        }
    }
    leaf-list weekday {
        type weekday;
        min-elements 1;
        max-elements 7;
        description
            "The weekday to which the periodic time range belongs.";
    }
    description

```

```
        "Periodic time that the associated function starts going into effect."
;
    }
    list absolute-time {
        key "start end";
        leaf start {
            type yang:date-and-time;
            description
                "Absolute start time and date";
        }
        leaf end {
            type yang:date-and-time;
            description
                "Absolute end time and date";
        }
        description
            "Absolute time and date that the associated function starts going into
effect.";
    }
    description
        "The time range object and its attributes.";
}
description
    "A list of time range objects";
}

/*
 * Groupings for region object and region group
 */
grouping region-objects {
    list pre-defined-region {
        key "name";
        config false;
        leaf name {
            type region-name;
            config false;
            description
                "The name of the predefined region.";
        }
        leaf desc {
            type string;
            config false;
            description
                "The description of the predefined region.";
        }
    }
    container region-ipv4-address {
        leaf-list address-ipv4 {
            type inet:ipv4-prefix;
            config false;
        }
    }
}
```

```
        description
            "IPv4 address.";
    }
    list address-ipv4-range {
        key "start-ipv4 end-ipv4";
        leaf start-ipv4 {
            type inet:ipv4-address;
            config false;
            description
                "Start ipv4 address.";
        }
        leaf end-ipv4 {
            type inet:ipv4-address;
            config false;
            description
                "End ipv4 address.";
        }
        description
            "A list of ipv4 address ranges";
    }
    description
        "The IPv4 addresses of the predefined region.";
}
container region-ipv6-address {
    if-feature support-ipv6-address;
    leaf-list address-ipv6 {
        type inet:ipv6-prefix;
        config false;
        description
            "IPv6 address.";
    }
    list address-ipv6-range {
        key "start-ipv6 end-ipv6";
        leaf start-ipv6 {
            type inet:ipv6-address;
            config false;
            description
                "Start ipv6 address.";
        }
        leaf end-ipv6 {
            type inet:ipv6-address;
            config false;
            description
                "End ipv6 address.";
        }
        description
            "A list of ipv6 address ranges";
    }
}
```

```
        description
            "The IPv6 addresses of the predefined region.";
    }
    description
        "A list of predefined region objects.";
}
list user-defined-region {
    key "name";
    leaf name {
        type region-name;
        description
            "The name of the user-defined region.";
    }
    leaf desc {
        type string;
        description
            "The description of the user-defined region.";
    }
    container coordinate {
        leaf longitude {
            type region-longitude;
            description
                "The latitude of the user-defined region.";
        }
        leaf latitude {
            type region-latitude;
            description
                "The longitude of the user-defined region.";
        }
    }
    description
        "The latitude and longitude of the user-defined region.";
}
container region-ipv4-address {
    leaf-list address-ipv4 {
        type inet:ipv4-prefix;
        description
            "IPv4 address.";
    }
    list address-ipv4-range {
        key "start-ipv4 end-ipv4";
        leaf start-ipv4 {
            type inet:ipv4-address;
            description
                "Start ipv4 address.";
        }
        leaf end-ipv4 {
            type inet:ipv4-address;
            description
```

```

        "End ipv4 address.";
    }
    description
        "A list of ipv4 address ranges";
}
description
    "The IPv4 addresses of the predefined region.";
}
container region-ipv6-address {
    if-feature support-ipv6-address;
    leaf-list address-ipv6 {
        type inet:ipv6-prefix;
        description
            "IPv6 address.";
    }
    list address-ipv6-range {
        key "start-ipv6 end-ipv6";
        leaf start-ipv6 {
            type inet:ipv6-address;
            description
                "Start ipv6 address.";
        }
        leaf end-ipv6 {
            type inet:ipv6-address;
            description
                "End ipv6 address.";
        }
    }
    description
        "A list of ipv6 address ranges";
}
description
    "The IPv6 addresses of the user-defined region.";
}
description
    "A list of user-defined region objects.";
}
description
    "A list of predefined region objects and a list of user-defined region objects.";
}

grouping region-groups {
    list region-group {
        key "name";
        leaf name {
            type region-name;
            description
                "The name of the region group.";
        }
    }
}

```

```
    leaf desc {
      type string;
      description
        "The description of the region group.";
    }
    leaf-list region-name {
      type region-name;
      description
        "A list of region objects.";
    }
    leaf-list region-group-name {
      type region-name;
      description
        "A list of region groups.";
    }
    description
      "Region group consists of a set of region objects or region groups.";
  }
  description
    "A list of region group objects.";
}

/*
 * Groupings for domain object
 */
grouping domain-objects {
  list domain-object {
    key "name";
    leaf name {
      type domain-name;
      description
        "The name of the domain object.";
    }
    leaf desc {
      type string;
      description
        "The description of the domain object.";
    }
    leaf-list domain {
      type string;
      description
        "A list of domains that consists the domain objects.";
    }
    description
      "Domain object and its attributes.";
  }
  description
    "A list of domain objects.";
```

```
}  
}
```

7. Acknowledgements

8. IANA Considerations

This document requires no IANA actions.

9. Security Considerations

Secure transport should be used to retrieve the current status of management plane security baseline.

10. References

10.1. Normative References

[I-D.ietf-i2nsf-capability]

Xia, L., Strassner, J., Basile, C., and D. Lopez,
"Information Model of NSFs Capabilities", draft-ietf-
i2nsf-capability-02 (work in progress), July 2018.

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H.
Birkholz, "Interface to Network Security Functions (I2NSF)
Terminology", draft-ietf-i2nsf-terminology-06 (work in
progress), July 2018.

[I-D.ietf-netmod-acl-model]

Jethanandani, M., Agarwal, S., Huang, L., and D. Blair,
"Network Access Control List (ACL) YANG Data Model",
draft-ietf-netmod-acl-model-20 (work in progress), October
2018.

[RFC8329]

Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R.
Kumar, "Framework for Interface to Network Security
Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018,
<<https://www.rfc-editor.org/info/rfc8329>>.

10.2. Informative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Authors' Addresses

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

Qiusi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

Email: linqiushi@huawei.com

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

J. Yang
J. Jeong
J. Kim
Sungkyunkwan University
July 2, 2018

Security Policy Translation in Interface to Network Security Functions
draft-yang-i2nsf-security-policy-translation-00

Abstract

This document proposes a scheme of security policy translation (i.e., Security Policy Translator) in Interface to Network Security Functions (I2NSF) Framework. When I2NSF User delivers a high-level security policy for a security service, Security Policy Translator in Security Controller translates it into a low-level security policy for Network Security Functions (NSFs).

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Necessity for Policy Translator	3
4. Design of Policy Translator	4
4.1. Overall Structure of Policy Translator	4
4.2. DFA-based Data Extractor	6
4.3. Data Converter	6
4.4. Context-free Grammar-based Policy Generator	7
5. Features of Policy Translator Design	7
6. Acknowledgments	8
7. Informative References	8

1. Introduction

This document defines a scheme of a security policy translation in Interface to Network Security Functions (I2NSF) Framework [RFC8329]. First of all, this document explains the necessity of a security policy translator (shortly called policy translator) in the I2NSF framework.

The policy translator resides in Security Controller in the I2NSF framework and translates a high-level security policy to a low-level security policy for Network Security Functions (NSFs). A high-level policy is specified by I2NSF User in the I2NSF framework and is delivered to Security Controller via Consumer-Facing Interface [consumer-facing-inf-dm]. A low-level policy is translated by Policy Translator in Security Controller and is delivered to NSFs to execute the rules corresponding to the low-level policy via NSF-Facing Interface [nsf-facing-inf-dm].

2. Terminology

This document uses the terminology specified in [i2nsf-terminology] [RFC8329].

3. Necessity for Policy Translator

Security Controller acts as a coordinator between I2NSF User and NSFs. Also, Security Controller has capability information of NSFs that are registered via Registration Interface [registration-inf-dm] by Developer's Management System [RFC8329].

As a coordinator, Security Controller needs to generate a low-level policy in the form of security rules intended by the high-level policy, which can be understood by the corresponding NSFs.

A high-level security policy is specified by RESTCONF/YANG [RFC8040][RFC6020], and a low-level security policy is specified by NETCONF/YANG [RFC6241][RFC6020]. The translation from a high-level security policy to the corresponding low-level security policy will be able to rapidly elevate I2NSF in real-world deployment. A rule in a high-level policy can include a broad target object, such as employees in a company for a security service (e.g., firewall and web filter). Such employees are from human resource (HR) department, software engineering department, and advertisement department. A keyword of employee needs to be mapped to these employees from various departments. This mapping needs to be handled by a policy translator in a flexible way while understanding the intention of a policy specification.

This document proposes an approach using Automata theory for the policy translation, such as deterministic finite automaton (DFA) and context-free grammar. Note that Automata theory is the foundation of programming languages and compilers. Thus, with this approach, I2NSF User can easily specify a high-level security policy that will be enforced into the corresponding NSFs with a compatible low-level security policy with the help of Policy Translator. Also, for easy management of Policy Translator, a modularized translator structure is proposed.

4. Design of Policy Translator

Commonly used security policies are created as xml files. A popular way to change the format of an xml file is to use an xslt document. However, the use of xslt makes it difficult to manage the policy translator and to handle the registration of new capabilities of NSFs. With the necessity for policy translator, this document proposes a policy translator based on Automata.

4.1. Overall Structure of Policy Translator

Figure 1 shows the overall design for Policy Translator in Security Controller. There are three main parts for Policy Translator: Extractor, Capability Converter, and Policy Generator.

Extractor is a DFA-based tool for extracting data from a high-level policy which I2NSF User delivered via Consumer-Facing Interface. Data Converter converts the extracted data to the capabilities of target NSFs for a low-level policy. Policy Generator generates a low-level policy which will execute the NSF capabilities from Converter.

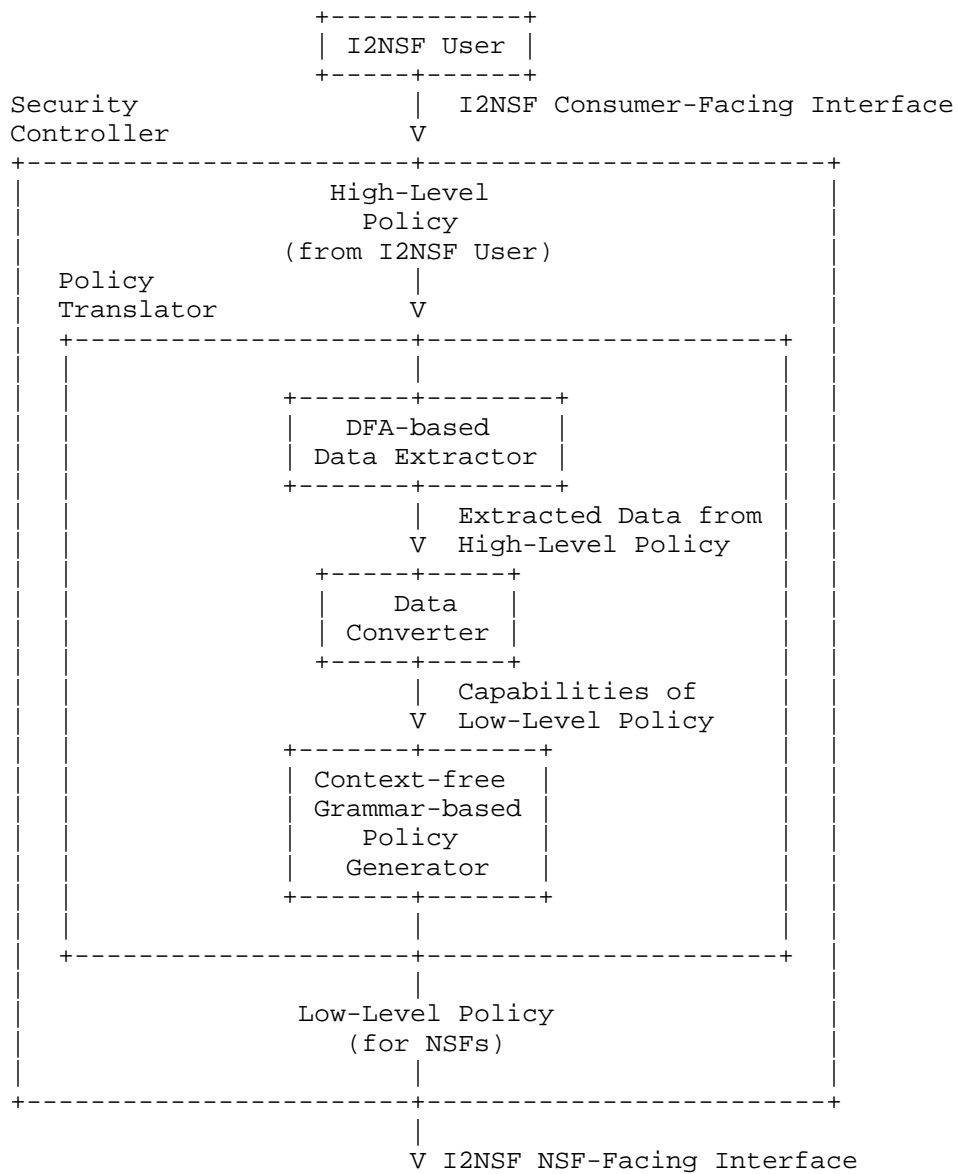


Figure 1: The Overall Design of Policy Translator

4.2. DFA-based Data Extractor

Figure 2 shows a design for Data Extractor in the policy translator. If the high-level policy contains data along the hierarchical structure of the standard YANG data model, data can be easily extracted using the state transition machine, DFA.

The extracted data is processed and used by an NSF to understand it. Extractor can be constructed by designing a DFA with the same hierarchical structure as a YANG data model.

Since the translator is modularized into a DFA structure, a visual understanding is feasible. Also, the following structure of Extractor is easy to manage. If I2NSF User wants to modify the data model of a high-level policy, it only needs to change the connection of the relevant DFA node.

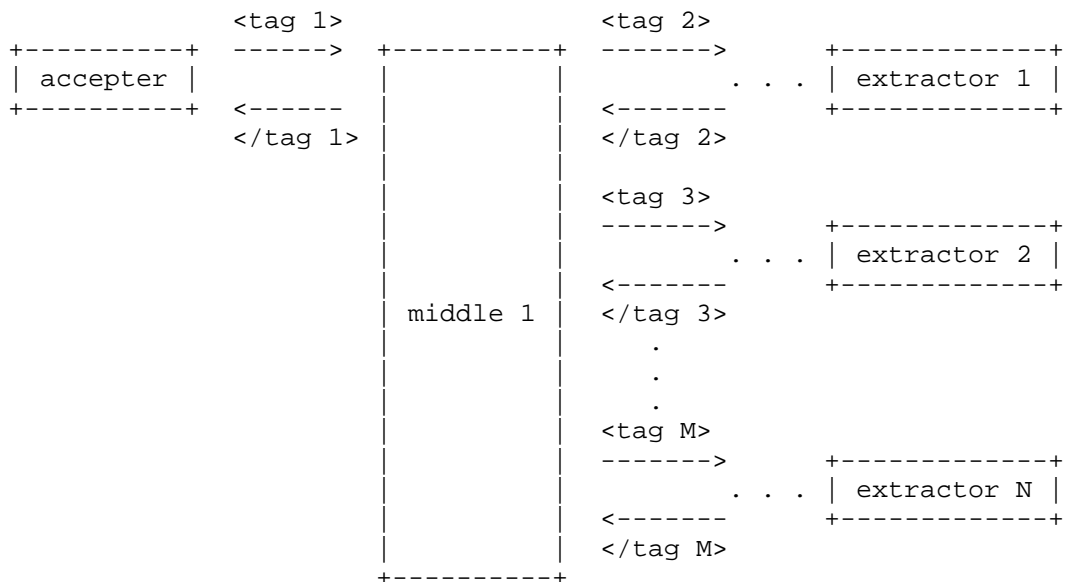


Figure 2: The Design of Data Extractor

4.3. Data Converter

Every NSF has its own unique capability. NSF registration is performed through Registration Interface, and the registration process is dedicated to Security Controller. Therefore, Security Controller already has all information about the capabilities of the NSFs. This means that Security Controller can find target NSFs with only the data of the high-level policy. Data Converter selects

target NSFs and converts the extracted data into the capabilities of selected NSFs. This eliminates the need for I2NSF User to set the target NSFs one by one.

4.4. Context-free Grammar-based Policy Generator

The grammar that makes up the low-level security policy is categorized into two types, Structure Grammar and Content Grammar. Structure Grammar is for grouping other tags into a hierarchy. A security administrator called manager constructs Structure Grammar in the form of an expression as the following equation:

- o `[structure_grammar] -> <structure_tag>[policy:1][policy:2]...[policy:n]</structure_tag>`

Content Grammar is for injecting data into low-level policies to be generated. The manager can construct Content Grammar in the form of an expression as following equation:

- o `[content_grammar] -> <content_tag>[content_data]</content_tag>`
- o `[content_data] -> data:1 | data:2 | ... | data:n`
- o `[content_grammar] -> [content_grammar][content_grammar]` (When duplication is allowed)

5. Features of Policy Translator Design

First, by showing the visualized translator structure, the manager can handle various policy changes. Translator can be shown by visualizing DFA and Context-free Grammar so that the manager can easily understand the structure of Policy Translator.

Second, if I2NSF User only keeps the hierarchy of the data model, I2NSF User can freely create high-level policies. In the case of DFA, data extraction can be performed in the same way even if the order of input is changed. The design of the policy translator is more flexible than the existing method that works by keeping the tag's position and order exactly.

Third, the structure of Policy Translator can be updated even while Policy Translator is operating. Because Policy Translator is modularized, the translator can adapt to changes in the NSF capability while the I2NSF framework is running. The function of changing the translator's structure can be provided through Registration Interface.

6. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea MSIT (Ministry of Science and ICT) (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This work was supported in part by the MSIT under the ITRC (Information Technology Research Center) support program (IITP-2018-2017-0-01633) supervised by the IITP.

7. Informative References

- [i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.
- [consumer-facing-inf-dm] Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", draft-ietf-i2nsf-consumer-facing-interface-dm-01 (work in progress), July 2018.
- [nsf-facing-inf-dm] Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-data-model-01 (work in progress), July 2018.
- [registration-inf-dm] Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-hyun-i2nsf-registration-dm-04 (work in progress), July 2018.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Jinhyuk Yang
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 8520 8039
EMail: jin.hyuk@skku.edu

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

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

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

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

