Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2017

S. Hares
Huawei
R. Moskowitz
HTT Consulting
L. Xia
Huawei
J. Jeong
J. Kim
Sungkyunkwan University

March 12, 2017

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

Abstract

This document defines a YANG data model for capabilities that enables an I2NSF user to control various network security functions in network security devices via an I2NSF security controller.

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/1id-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 September 13, 2017.

Copyright Notice

Table of Contents

1. Introduction

[i2nsf-problem-statement] proposes two different types of interfaces:

- Interface between I2NSF user and I2NSF security controller called I2NSF consumer-facing interface
- Interface between I2NSF security controller and network security functions (NSFs) called I2NSF NSF-facing interface

This document provides a YANG model that defines the capabilities for security devices that can be utilized by I2NSF NSF-facing interface between the I2NSF security controller and the NSF devices to express the capabilities of NSF devices. This YANG model can also be used by the IN2SF user (or I2NSF client) to provide security controller with a complete list of the I2NSF capabilities that can be controlled by security controller. This document defines a YANG [RFC6020] data model based on the [i2nsf-cap-im]. Terms used in document are defined in [i2nsf-terminology]. [i2nsf-cap-im] defines the following type of functionality in NSFs.

- Network Security Control
- Content Security Control
- Attack Mitigation Control

This document contains high-level YANG for each type of control.

2. Requirements Language

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

3. Terminology

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

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

independent of data repository, data definition language, query language, implementation language, and protocol.

3.1. Tree Diagrams

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

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

4. High-Level YANG

This section provides an overview of the high-level YANG.

4.1. Capabilities per NSF

The high-level YANG capabilities per NSF devices, controller, or application is the following:

```
module : ietf-i2nsf-capability
  +--rw sec-ctl-capabilities
  +--rw nsf-capabilities
    +--rw nsf* [nsf-name]
      +--rw nsf-name  string
      +--rw nsf-address
      |  +--rw (nsf-address-type)?
      |    +--: (ipv4-address)
      |    |  +--rw ipv4-address  inet:ipv4-address
      |    +--: (ipv6-address)
      |       +--rw ipv6-address  inet:ipv6-address
      +--rw net-sec-control-capabilities
      |  uses i2nsf-net-sec-control-caps
      +--rw con-sec-control-capabilities
      |  uses i2nsf-con-sec-control-caps
      +--rw attack-mitigation-capabilities
      |  uses i2nsf-attack-mitigation-control-caps
      +--rw it-resource
      |  uses i2nsf-it-resources
```

Figure 1: High-Level YANG of I2NSF Capability Interface

Each of these section mirror sections in: [i2nsf-cap-im]. The high-level YANG for net-sec-control-capabilities, con-sec-control-capabilities, and attack-mitigation-capabilities. This draft is also utilizes the concepts originated in Basile, Lioy, Pitscheider, and Zhao[2015] concerning conflict resoulution, use of external data, and IT-Resources. The authors are grateful to Cataldo for pointing out this excellent work.

4.2. Network Security Control

This section expands the

```
+--rw net-sec-control-capabilities
| uses i2nsf-net-sec-control-caps
```

Network Security Control

```
  +--rw i2nsf-net-sec-control-caps
    +--rw network-security-control
      +--rw nsc-support?  boolean
      +--rw nsc-fcn*  [nsc-fcn-name]
        +--rw nsc-fcn-name  string  //std or vendor name
```

Figure 2: High-Level YANG of Network Security Control

4.3. Content Security Control

This section expands the

```
            +--rw net-sec-control-capabilities
            | uses i2nsf-con-sec-control-caps
```

Content Security Control

```
  +--rw i2nsf-con-sec-control-caps
    +--rw content-security-control
      +--rw antivirus
      |  +--rw antivirus-support?  boolean
      |  +--rw antivirus-fcn*  [antivirus-fcn-name]
      |    +--rw antivirus-fcn-name  string  //std or vendor name
      +--rw ips
      |  +--rw ips-support?  boolean
      |  +--rw ips-fcn*  [ips-fcn-name]
      |    +--rw ips-fcn-name  string  //std or vendor name
      +--rw ids
      |  +--rw ids-support?  boolean
      |  +--rw ids-fcn*  [ids-fcn-name]
      |    +--rw ids-fcn-name  string  //std or vendor name
      +--rw url-filter
      |  +--rw url-filter-support?  boolean
      |  +--rw url-filter-fcn*  [url-filter-fcn-name]
      |    +--rw url-filter-fcn-name  string  //std or vendor name
      +--rw data-filter
      |  +--rw data-filter-support?  boolean
      |  +--rw data-filter-fcn*  [data-filter-fcn-name]
      |    +--rw data-filter-fcn-name  string  //std or vendor name
      +--rw mail-filter
      |  +--rw mail-filter-support?  boolean
      |  +--rw mail-filter-fcn*  [mail-filter-fcn-name]
      |    +--rw mail-filter-fcn-name  string  //std or vendor name
      +--rw dns-filter
      |  +--rw dns-filter-support?  boolean
      |  +--rw dns-filter-fcn*  [dns-filter-name]
      |    +--rw dns-filter-fcn-name  string  //std or vendor name
      +--rw ftp-filter
      |  +--rw ftp-filter-support?  boolean
      |  +--rw ftp-filter-fcn*  [ftp-filter-fcn-name]
      |    +--rw ftp-filter-fcn-name  string  //std or vendor name
      +--rw games-filter
      |  +--rw games-filter-support?  boolean
      |  +--rw games-filter-fcn*  [games-filter-fcn-name]
      |    +--rw games-filter-fcn-name  string  //std or vendor name
      +--rw p2p-filter
```

```
|  +--rw p2p-filter-support?  boolean
|  +--rw p2p-filter-fcn*  [p2p-filter-fcn-name]
|    +--rw p2p-filter-fcn-name  string  //std or vendor name
+--rw rpc-filter
|  +--rw rpc-filter-support?  boolean
|  +--rw rpc-filter-fcn*  [rpc-filter-fcn-name]
|    +--rw rpc-filter-fcn-name  string  //std or vendor name
+--rw sql-filter
|  +--rw sql-filter-support?  boolean
|  +--rw sql-filter-fcn*  [sql-filter-fcn-name]
|    +--rw sql-filter-fcn-name  string  //std or vendor name
+--rw telnet-filter
|  +--rw telnet-filter-support?  boolean
|  +--rw telnet-filter-fcn*  [telnet-filter-fcn-name]
|    +--rw telnet-filter-fcn-name  string  //std or vendor name
+--rw tftp-filter
|  +--rw tftp-filter-support?  boolean
|  +--rw tftp-filter-fcn*  [tftp-filter-fcn-name]
|    +--rw tftp-filter-fcn-name  string  //std or vendor name
+--rw file-blocking
|  +--rw file-blocking-support?  boolean
|  +--rw file-blocking-fcn*  [file-blocking-fcn-name]
|    +--rw file-blocking-fcn-name  string  //std or vendor name
+--rw pkt-capture
|  +--rw pkt-capture-support?  boolean
|  +--rw pkt-capture-fcn*  [pkt-capture-fcn-name]
|    +--rw pkt-capture-fcn-name  string  //std or vendor name
+--rw app-control
|  +--rw app-control-support?  boolean
|  +--rw app-control-fcn*  [app-control-fcn-name]
|    +--rw app-control-fcn-name  string  //std or vendor name
+--rw voip-volte
   +--rw voip-volte-support?  boolean
   +--rw voip-volte-fcn*  [voip-volte-fcn-name]
     +--rw voip-volte-fcn-name  string  //std or vendor name
```

Figure 3: High-Level YANG of Content Security Control

4.4. Attack Mitigation Control

This high-level YANG below expands the following section of the top-level model:

```
+--rw attack-mitigation-control-capabilities
| uses i2nsf-attack-mitigation-control-caps
```

Attack Mitigation Control

```
   +--rw i2nsf-attack-mitigation-control-caps
     +--rw attack-mitigation-control
       +--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
         |    |    |  +--rw syn-flood-attack-support?  boolean
         |    |    |  +--rw syn-flood-fcn*  [syn-flood-fcn-name]
         |    |    |    +--rw syn-flood-fcn-name  string
         |    |    +--rw udp-flood-attack
         |    |    |  +--rw udp-flood-attack-support?  boolean
         |    |    |  +--rw udp-flood-fcn*  [udp-flood-fcn-name]
         |    |    |    +--rw udp-flood-fcn-name  string
         |    |    +--rw icmp-flood-attack
         |    |    |  +--rw icmp-flood-attack-support?  boolean
         |    |    |  +--rw icmp-flood-fcn*  [icmp-flood-fcn-name]
         |    |    |    +--rw icmp-flood-fcn-name  string
         |    |    +--rw ip-fragment-flood-attack
         |    |    |  +--rw ip-fragment-flood-attack-support?  boolean
         |    |    |  +--rw ip-frag-flood-fcn*  [ip-frag-flood-fcn-name]
         |    |    |    +--rw ip-frag-flood-fcn-name  string
         |    |    +--rw ipv6-related-attack
         |    |       +--rw ipv6-related-attack-support?  boolean
         |    |       +--rw ipv6-related-fcn*  [ipv6-related-fcn-name]
         |    |         +--rw ipv6-related-fcn-name  string
         |    +--: (app-layer-ddos-attack)
         |       +--rw app-layer-ddos-attack-types
         |         +--rw http-flood-attack
         |         |  +--rw http-flood-attack-support?  boolean
         |         |  +--rw http-flood-fcn*  [http-flood-fcn-name]
         |         |    +--rw http-flood-fcn-name  string
         |         +--rw https-flood-attack
         |         |  +--rw https-flood-attack-support?  boolean
         |         |  +--rw https-flood-fcn*  [https-flood-fcn-name]
         |         |    +--rw https-flood-fcn-name  string
         |         +--rw dns-flood-attack
         |         |  +--rw dns-flood-attack-support?  boolean
         |         |  +--rw dns-flood-fcn*  [dns-flood-fcn-name]
         |         |    +--rw dns-flood-fcn-name  string
         |         +--rw dns-amp-flood-attack
         |         |  +--rw dns-amp-flood-attack-support?  boolean
         |         |  +--rw dns-amp-flood-fcn*  [dns-amp-flood-fcn-name]
         |         |    +--rw dns-amp-flood-fcn-name  string
         |         +--rw ssl-ddos-attack
         |            +--rw ssl-ddos-attack-support?  boolean
         |            +--rw ssl-ddos-fcn*  [ssl-ddos-fcn-name]
```

```
|              +--rw ssl-ddos-fcn-name  string
+--: (single-packet-attack)
   +--rw (single-packet-attack-type)?
     +--: (scan-and-sniff-attack)
     |  +--rw ip-sweep-attack
     |  |  +--rw ip-sweep-attack-support?  boolean
     |  |  +--rw ip-sweep-fcn*  [ip-sweep-fcn-name]
     |  |    +--rw ip-sweep-fcn-name  string
     |  +--rw port-scanning-attack
     |     +--rw port-scanning-attack-support?  boolean
     |     +--rw port-scanning-fcn*  [port-scanning-fcn-name]
     |       +--rw port-scanning-fcn-name  string
     +--: (malformed-packet-attack)
     |  +--rw ping-of-death-attack
     |  |  +--rw ping-of-death-attack-support?  boolean
     |  |  +--rw ping-of-death-fcn*  [ping-of-death-fcn-name]
     |  |    +--rw ping-of-death-fcn-name  string
     |  +--rw teardrop-attack
     |     +--rw teardrop-attack-support?  boolean
     |     +--rw tear-drop-fcn*  [tear-drop-fcn-name]
     |       +--rw tear-drop-fcn-name  string
     +--: (special-packet-attack)
        +--rw oversized-icmp-attack
        |  +--rw oversized-icmp-attack-support?  boolean
        |  +--rw oversized-icmp-fcn*  [oversized-icmp-fcn-name]
        |    +--rw oversized-icmp-fcn-name  string
        +--rw tracert-attack
           +--rw tracert-attack-support?  boolean
           +--rw tracert-fcn*  [tracert-fcn-name]
             +--rw tracert-fcn-name  string
```

Figure 4: High-Level YANG of Attack Mitigation Control

4.5. IT Resources linked to Capabilities

This section provides a link between capabilities and IT resources. This section has a list of IT resources by name. Additional input is needed.

```
               +--rw it-resource
               | uses i2nsf-it-resources
```

It Resource

```
               +--rw i2nsf-it-resources
                 +--rw it-resources* [it-resource-id]
                    +--rw it-resource-id  uint64
                    +--rw it-resource-name  string
```

Figure 5: High-Level YANG of IT Resources

4.6. Actions

Notifications indicate when rules are added or deleted. These notifications will be defined later.

5. YANG Modules

This section introduces a YANG module for the information model of I2NSF capability interface, as defined in the [i2nsf-cap-im].

<CODE BEGINS> file "ietf-i2nsf-capability@2017-03-12.yang"

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

  import ietf-inet-types{
    prefix inet;
  }

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

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

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

     WG Chair: Linda Dunbar
```

```
        <mailto:Linda.duhbar@huawei.com>

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

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

        Editor: Jinyong Tim Kim
        <mailto:wlsdyd0930@nate.com>";

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

    revision "2017-03-12"{
      description "The fourth revision";
      reference
        "draft-xibassnez-i2nsf-capability-00
         draft-hares-i2nsf-capability-data-model-01";
    }

    container sec-ctl-capabilities {
      description
        "sec-ctl-capabilities";
    }



    grouping i2nsf-net-sec-control-caps {
      description
        "i2nsf-net-sec-control-caps";
      container network-security-control {
        description
          "i2nsf-net-sec-control-caps";
        leaf nsc-support {
          type boolean;
          mandatory true;
          description
            "nsc-support";
        }
        list nsc-fcn {
          key "nsc-fcn-name";
          description
            "nsc-fcn";
          leaf nsc-fcn-name {
            type string;
            mandatory true;
```

```
            description
              "nsc-fcn-name";
          }
        }
      }
    }

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

      container content-security-control {
        description
          "content-security-control";

        container antivirus {
          description
            "antivirus";

          leaf antivirus-support {
            type boolean;
            mandatory true;
            description
              "antivirus-support";
          }
          list antivirus-fcn-name {
            key "antivirus-fcn-name";
            description
              "antivirus-fcn-name";

            leaf antivirus-fcn-name {
              type string;
              mandatory true;
              description
                "antivirus-fcn-name";
            }
          }
        }

        container ips {
          description
            "ips";

          leaf ips-support {
            type boolean;
            mandatory true;
            description
              "ips-support";
```

```
          }
          list ips-fcn {
            key "ips-fcn-name";
            description
              "ips-fcn";

            leaf ips-fcn-name {
              type string;
              mandatory true;
              description
                "ips-fcn-name";
            }
          }
        }

        container ids {
          description
            "ids";

          leaf ids-support {
            type boolean;
            mandatory true;
            description
              "ids-support";
          }
          list ids-fcn {
            key "ids-fcn-name";
            description
              "ids-fcn";

            leaf ids-fcn-name {
              type string;
              mandatory true;
              description
                "ids-fcn-name";
            }
          }
        }

        container url-filter {
          description
            "url-filter";

          leaf url-filter-support {
            type boolean;
            mandatory true;
            description
              "url-filter-support";
```

```
      }
      list url-filter-fcn {
        key "url-filter-fcn-name";
        description
          "url-filter-fcn";

        leaf url-filter-fcn-name {
          type string;
          mandatory true;
          description
            "url-filter-fcn-name";
        }
      }
    }

    container data-filter {
      description
        "data-filter";

      leaf data-filter-support {
        type boolean;
        mandatory true;
        description
          "data-filter-support";
      }
      list data-filter-fcn {
        key "data-filter-fcn-name";
        description
          "data-filter-fcn";

        leaf data-filter-fcn-name {
          type string;
          mandatory true;
          description
            "data-filter-fcn-name";
        }
      }
    }

    container mail-filter {
      description
        "mail-filter";

      leaf mail-filter-support {
        type boolean;
        mandatory true;
        description
          "mail-filter-support";
```

```
          }
          list mail-filter-fcn {
            key "mail-filter-fcn-name";
            description
              "mail-filter-fcn";

            leaf mail-filter-fcn-name {
              type string;
              mandatory true;
              description
                "mail-filter-fcn-name";
            }
          }
        }

        container dns-filter {
          description
            "dns-filter";

          leaf dns-filter-support {
            type boolean;
            mandatory true;
            description
              "dns-filter-support";
          }
          list dns-filter-fcn {
            key "dns-filter-fcn-name";
            description
              "dns-filter-fcn";

            leaf dns-filter-fcn-name {
              type string;
              mandatory true;
              description
                "dns-filter-fcn-name";
            }
          }
        }

        container ftp-filter {
          description
            "ftp-filter";

          leaf ftp-filter-support {
            type boolean;
            mandatory true;
            description
              "ftp-filter-support";
```

```
          }
          list ftp-filter-fcn {
            key "ftp-filter-fcn-name";
            description
              "ftp-filter-fcn";

            leaf ftp-filter-fcn-name {
              type string;
              mandatory true;
              description
                "ftp-filter-fcn-name";
            }
          }
        }

        container games-filter {
          description
            "games-filter";

          leaf games-filter-support {
            type boolean;
            mandatory true;
            description
              "games-filter-support";
          }
          list games-filter-fcn {
            key "games-filter-fcn-name";
            description
              "games-filter-fcn";

            leaf games-filter-fcn-name {
              type string;
              mandatory true;
              description
                "games-filter-fcn-name";
            }
          }
        }

        container p2p-filter {
          description
            "p2p-filter";

          leaf p2p-filter-support {
            type boolean;
            mandatory true;
            description
              "p2p-filter-support";
```

```
      }
      list p2p-filter-fcn {
        key "p2p-filter-fcn-name";
        description
          "p2p-filter-fcn";

        leaf p2p-filter-fcn-name {
          type string;
          mandatory true;
          description
            "p2p-filter-fcn-name";
        }
      }
    }

    container rpc-filter {
      description
        "rpc-filter";

      leaf rpc-filter-support {
        type boolean;
        mandatory true;
        description
          "rpc-filter-support";
      }
      list rpc-filter-fcn {
        key "rpc-filter-fcn-name";
        description
          "rpc-filter-fcn";

        leaf rpc-filter-fcn-name {
          type string;
          mandatory true;
          description
            "rpc-filter-fcn-name";
        }
      }
    }

    container sql-filter {
      description
        "sql-filter";

      leaf sql-filter-support {
        type boolean;
        mandatory true;
        description
          "sql-filter-support";
```

```
          }
          list sql-filter-fcn {
            key "sql-filter-fcn-name";
            description
              "sql-filter-fcn";

            leaf sql-filter-fcn-name {
              type string;
              mandatory true;
              description
                "sql-filter-fcn-name";
            }
          }
        }

        container telent-filter {
          description
            "telent-filter";

          leaf telent-filter-support {
            type boolean;
            mandatory true;
            description
              "telent-filter-support";
          }
          list telent-filter-fcn {
            key "telent-filter-fcn-name";
            description
              "telent-filter-fcn";

            leaf telent-filter-fcn-name {
              type string;
              mandatory true;
              description
                "telent-filter-fcn-name";
            }
          }
        }

        container tftp-filter {
          description
            "tftp-filter";

          leaf tftp-filter-support {
            type boolean;
            mandatory true;
            description
              "tftp-filter-support";
```

```
          }
          list tftp-filter-fcn {
            key "tftp-filter-fcn-name";
            description
              "tftp-filter-fcn";

            leaf tftp-filter-fcn-name {
              type string;
              mandatory true;
              description
                "tftp-filter-fcn-name";
            }
          }
        }

        container file-blocking {
          description
            "file-blocking";

          leaf file-blocking-support {
            type boolean;
            mandatory true;
            description
              "file-blocking-support";
          }
          list file-blocking-fcn {
            key "file-blocking-fcn-name";
            description
              "file-blocking-fcn";

            leaf file-blocking-fcn-name {
              type string;
              mandatory true;
              description
                "file-blocking-fcn-name";
            }
          }
        }

        container file-isolate {
          description
            "file-isolate";

          leaf file-isolate-support {
            type boolean;
            mandatory true;
            description
              "file-isolate-support";
```

```
          }
          list file-isolate-fcn {
            key "file-isolate-fcn-name";
            description
              "file-isolate-fcn";

            leaf file-isolate-fcn-name {
              type string;
              mandatory true;
              description
                "file-isolate-fcn-name";
            }
          }
        }

        container pkt-capture {
          description
            "pkt-capture";

          leaf pkt-capture-support {
            type boolean;
            mandatory true;
            description
              "pkt-capture-support";
          }
          list pkt-capture-fcn {
            key "pkt-capture-fcn-name";
            description
              "pkt-capture-fcn";

            leaf pkt-capture-fcn-name {
              type string;
              mandatory true;
              description
                "pkt-capture-fcn-name";
            }
          }
        }

        container app-control {
          description
            "app-control";

          leaf app-control-support {
            type boolean;
            mandatory true;
            description
              "app-control-support";
```

```
          }
          list app-control-fcn {
            key "app-control-fcn-name";
            description
              "app-control-fcn";

            leaf app-control-fcn-name {
              type string;
              mandatory true;
              description
                "app-control-fcn-name";
            }
          }
        }

        container voip-volte {
          description
            "voip-volte";

          leaf voip-volte-support {
            type boolean;
            mandatory true;
            description
              "voip-volte-support";
          }
          list voip-volte-fcn {
            key "voip-volte-fcn-name";
            description
              "voip-volte-fcn";

            leaf voip-volte-fcn-name {
              type string;
              mandatory true;
              description
                "voip-volte-fcn-name";
            }
          }
        }
      }
    }

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

      container attack-mitigation-control {
        description
          "attack-mitigation-control";
```

```
          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";
                    container syn-flood-attack {
                      description
                        "syn-flood-attack";
                      leaf syn-flood-attack-support {
                        type boolean;
                        mandatory true;
                        description
                          "syn-flood-attack-support";
                      }
                      list syn-flood-fcn {
                        key "syn-flood-fcn-name";
                        description
                          "syn-flood-fcn";
                        leaf syn-flood-fcn-name {
                          type string;
                          mandatory true;
                          description
                            "syn-flood-fcn-name";
                        }
                      }
                    }
                    container udp-flood-attack {
                      description
                        "udp-flood-attack";
                      leaf udp-flood-attack-support {
                        type boolean;
                        mandatory true;
                        description
                          "udp-flood-attack-support";
                      }
                      list udp-flood-fcn {
                        key "udp-flood-fcn-name";
                        description
```

```
            "udp-flood-fcn";
          leaf udp-flood-fcn-name {
            type string;
            mandatory true;
            description
              "udp-flood-fcn-name";
          }
        }
      }
      container icmp-flood-attack {
        description
          "icmp-flood-attack";
        leaf icmp-flood-attack-support {
          type boolean;
          mandatory true;
          description
            "icmp-flood-attack-support";
        }
        list icmp-flood-fcn {
          key "icmp-flood-fcn-name";
          description
            "icmp-flood-fcn";
          leaf icmp-flood-fcn-name {
            type string;
            mandatory true;
            description
              "icmp-flood-fcn-name";
          }
        }
      }
      container ip-fragment-flood-attack {
        description
          "ip-fragment-flood-attack";
        leaf ip-fragment-flood-attack-support {
          type boolean;
          mandatory true;
          description
            "ip-fragment-flood-attack-support";
        }
        list frag-flood-fcn {
          key "ip-frag-flood-fcn-name";
          description
            "frag-flood-fcn";
          leaf ip-frag-flood-fcn-name {
            type string;
            mandatory true;
            description
              "ip-frag-flood-fcn-name";
```

```
                    }
                  }
                }
                container ipv6-related-attack {
                  description
                    "ipv6-related-attack";
                  leaf ipv6-related-attack-support {
                    type boolean;
                    mandatory true;
                    description
                      "ipv6-related-attack-support";
                  }
                  list ipv6-related-fcn {
                    key "ipv6-related-fcn-name";
                    description
                      "ipv6-related-fcn";
                    leaf ipv6-related-fcn-name {
                      type string;
                      mandatory true;
                      description
                        "ipv6-related-fcn-name";
                    }
                  }
                }
              }
            }
            case app-layer-ddos-attack {
              description
                "app-layer-ddos-attack";
              container app-layer-ddos-attack-types {
                description
                  "app-layer-ddos-attack-types";
                container http-flood-attack {
                  description
                    "http-flood-attack";
                  leaf http-flood-attack-support {
                    type boolean;
                    mandatory true;
                    description
                      "http-flood-attack-support";
                  }
                  list http-flood-fcn {
                    key "http-flood-fcn-name";
                    description
                      "http-flood-fcn";
                    leaf http-flood-fcn-name {
                      type string;
                      mandatory true;
```

```
          description
            "http-flood-fcn-name";
        }
      }
    }
    container https-flood-attack {
      description
        "https-flood-attack";
      leaf https-flood-attack-support {
        type boolean;
        mandatory true;
        description
          "https-flood-attack-support";
      }
      list https-flood-fcn {
        key "https-flood-fcn-name";
        description
          "https-flood-fcn";
        leaf https-flood-fcn-name {
          type string;
          mandatory true;
          description
            "https-flood-fcn-name";
        }
      }
    }
    container dns-flood-attack {
      description
        "dns-flood-attack";
      leaf dns-flood-attack-support {
        type boolean;
        mandatory true;
        description
          "dns-flood-attack-support";
      }
      list dns-flood-fcn {
        key "dns-flood-fcn-name";
        description
          "dns-flood-fcn";
        leaf dns-flood-fcn-name {
          type string;
          mandatory true;
          description
            "dns-flood-fcn-name";
        }
      }
    }
    container dns-amp-flood-attack {
```

```
                    description
                      "dns-amp-flood-attack";
                    leaf dns-flood-attack-support {
                      type boolean;
                      mandatory true;
                      description
                        "dns-flood-attack-support";
                    }
                    list dns-amp-flood-fcn {
                      key "dns-amp-flood-fcn-name";
                      description
                        "dns-amp-flood-fcn";
                      leaf dns-amp-flood-fcn-name {
                        type string;
                        mandatory true;
                        description
                          "dns-amp-flood-fcn-name";
                      }
                    }
                  }
                  container ssl-ddos-attack {
                    description
                      "ssl-ddos-attack";
                    leaf ssl-ddos-attack-support {
                      type boolean;
                      mandatory true;
                      description
                        "ssl-ddos-attack-support";
                    }
                    list ssl-ddos-fcn {
                      key "ssl-ddos-fcn-name";
                      description
                        "ssl-ddos-fcn";
                      leaf ssl-ddos-fcn-name {
                        type string;
                        mandatory true;
                        description
                          "ssl-ddos-fcn-name";
                      }
                    }
                  }
                }
              }
            }
          }

          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";
              container ip-sweep-attack {
                description
                  "ip-sweep-attack";
                leaf ip-sweep-attack-suppor {
                  type boolean;
                  mandatory true;
                  description
                    "ip-sweep-attack-suppor";
                }
                list ip-sweep-fcn {
                  key "ip-sweep-fcn-name";
                  description
                    "ip-sweep-fcn";
                  leaf ip-sweep-fcn-name {
                    type string;
                    mandatory true;
                    description
                      "ip-sweep-fcn-name";
                  }
                }
              }
              container port-scanning-attack {
                description
                  "port-scanning-attack";
                leaf port-scanning-attack-support {
                  type boolean;
                  mandatory true;
                  description
                    "port-scanning-attack-support";
                  }
                list port-scanning-fcn {
                  key "port-scanning-fcn-name";
                  description
                    "port-scanning-fcn";
                  leaf port-scanning-fcn-name {
                    type string;
                    mandatory true;
                    description
                      "port-scanning-fcn-name";
                  }
                }
```

```
        }
      }
      case malformed-packet-attack {
        description
          "malformed-packet-attack";
        container ping-of-death-attack {
          description
            "ping-of-death-attack";
          leaf ping-of-death-attack-support {
            type boolean;
            mandatory true;
            description
              "ping-of-death-attack-support";
          }
          list ping-of-death-fcn {
            key "ping-of-death-fcn-name";
            description
              "ping-of-death-fcn";
            leaf ping-of-death-fcn-name {
              type string;
              mandatory true;
              description
                "ping-of-death-fcn-name";
            }
          }
        }
        container teardrop-attack {
          description
            "teardrop-attack";
          leaf teardrop-attack-support {
            type boolean;
            mandatory true;
            description
              "teardrop-attack-support";
          }
          list tear-drop-fcn {
            key "tear-drop-fcn-name";
            description
              "tear-drop-fcn";
            leaf tear-drop-fcn-name {
              type string;
              mandatory true;
              description
                "tear-drop-fcn-name";
            }
          }
        }
      }
```

```
            case special-packet-attack {
              description
                "special-packet-attack";
              container oversized-icmp-attack {
                description
                  "oversized-icmp-attack";
                leaf oversized-icmp-attack-support {
                  type boolean;
                  mandatory true;
                  description
                    "oversized-icmp-attack-support";
                }
                list oversized-icmp-fcn {
                  key "oversized-icmp-fcn-name";
                  description
                    "oversized-icmp-fcn";
                  leaf oversized-icmp-fcn-name {
                    type string;
                    mandatory true;
                    description
                      "oversized-icmp-fcn-name";
                  }
                }
              }
              container tracert-attack {
                description
                  "tracert-attack";
                leaf tracert-attack-support {
                  type boolean;
                  mandatory true;
                  description
                    "tracert-attack-support";
                }
                list tracert-fcn {
                  key "tracert-fcn-name";
                  description
                    "tracert-fcn";
                  leaf tracert-fcn-name {
                    type string;
                    mandatory true;
                    description
                      "tracert-fcn-name";
                  }
                }
              }
            }
          }
        }
```

```
        }
      }
    }

    grouping i2nsf-it-resources {
      description
        "i2nsf-it-resource";
      list it-resources {
        key "it-resource-id";
        description
          "it-resource";
        leaf it-resource-id {
          type uint64;
          mandatory true;
          description
            "it-resource-id";
        }
        leaf it-resource-name {
          type string;
          mandatory true;
          description
            "it-resource-name";
        }
      }
    }

    container nsf-capabilities {
      description
        "nsf-capabilities";

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

        container net-sec-control-capabilities {
          uses i2nsf-net-sec-control-caps;
          description
            "net-sec-control-capabilities";
        }
        container con-sec-control-capabilities {
          uses i2nsf-con-sec-control-caps;
          description
            "con-sec-control-capabilities";
        }
        container attack-mitigation-capabilities {
          uses i2nsf-attack-mitigation-control-caps;
          description
            "attack-mitigation-capabilities";
        }
        container it-resource {
          uses i2nsf-it-resources;
          description
            "it-resource";
        }
      }
    }
  }
```

<CODE ENDS>

Figure 6: Data Model of I2NSF Capability Interface

6. IANA Considerations

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

7. Security Considerations

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

8. Acknowledgements

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

This document has greatly benefited from inputs by Daeyoung Hyun, Hyoungshick Kim, Jung-Soo Park, Tae-Jin Ahn, and Se-Hui Lee.

9. References

9.1. 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.

9.2. Informative References

[i2nsf-cap-im] Xia, L., Strassner, J., Zhang, D., Li, K., Basile, C., Lioy, A., Lopez, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-00 (work in progress), Novemver 2016.

[i2nsf-problem-statement] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases",

draft-ietf-i2nsf-problem-and-use-cases-11 (work in progress), March 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-03 (work in progress), March 2017.

[i2rs-rib-data-model] Wang, L., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.

[supa-policy-info-model] Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-02 (work in progress), January 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-04 (work in progress), October 2016.

Appendix A. Changes from draft-hares-i2nsf-capability-data-model-00

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

- IPv6 is supported for the addresses of NSF devices.
- Content Security Control is supported for various content-based security services, such as dns-filter, ftp-filter, games-filter, p2p-filter, rpc-filter, sql-filter, telnet-filter, and tftp-filter.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

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

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

Phone:
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

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: wlsdyd0930@nate.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

S. Hyun
J. Jeong
S. Woo
Y. Yeo
Sungkyunkwan University
J. Park
ETRI

March 13, 2017

NSF-triggered Traffic Steering Framework
draft-hyun-i2nsf-nsf-triggered-steering-02

Abstract

This document describes an architecture of the Interface to Network Security Functions (I2NSF) framework which enables traffic steering between Network Security Functions (NSFs) for security policy enforcement. Such traffic steering enables the composite inspection of network traffic by steering the traffic through multiple types of security functions according to the information model for the NSF-facing interface in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve NSF-triggered traffic steering. It also describes representative use cases to address major benefits from the proposed architecture.

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various security functions cooperatively analyze network traffic [sfc-ns-use-cases] [RFC7498] [i2nsf-problem] [capability-im]. In addition, depending on the characteristics of network traffic and their suspiciousness level, the different types of network traffic need to be analyzed through the different sets of security functions. [capability-im] proposes an information model for the interface between a Security Controller and Network Security Functions (NSFs) (called NSF-facing interface) in the Interface to Network Security Functions (I2NSF) framework [i2nsf-framework]. This NSF-facing interface enables an NSF to trigger further inspection by calling another NSF based on its own analysis results. However, the current design of the I2NSF framework does not consider network traffic steering fully in order to enable such consecutive inspections through multiple security functions.

In this document, we propose an architecture that integrates additional components for traffic steering over NSFs into the I2NSF framework. We extend the security controller’s functionalities such that it can interpret a high-level policy of NSF-triggered traffic steering into a low-level policy and manage them. It also keeps track of the available network security function instances and their information (e.g., network information and workload), and makes a decision on which NSF instances to use for a given network security function. Based on the forwarding information provided by the security controller, the security function forwarder performs network traffic steering through required security functions. The security function forwarder is also responsible for interpreting inspection result from a network security function to enforce more advanced inspection. We define an additional packet header format to specify security inspection results and advanced inspection requests.

2. Objective

- Policy configuration for consecutive inspections: NSF-triggered traffic steering architecture allows policy configuration and management of network security function triggering. Based on the triggering policy, relevant network traffic can be analyzed through various security functions in a composite, cooperative manner.

- Network traffic steering for consecutive inspection: NSF-triggered traffic steering architecture allows network traffic to be steered through multiple required network security functions based on the triggering policy. Moreover, the I2NSF information model for NSF-facing interface [capability-im] requires a security function to

call another security function for further inspection based on its own inspection result. To meet this requirement, NSF-triggered traffic steering architecture also enables traffic forwarding from one security function to another security function.

- Load balancing over network security function instances: NSF-triggered traffic steering architecture provides load balancing of incoming traffic over available network security function instances by leveraging the flexible traffic steering mechanism. For this objective, it also performs dynamic instantiation of a security function when there are an excessive amount of requests for that network security function.

3. Terminology

This document uses the terminology described in [RFC7665], [RFC7665] [sfc-ns-use-cases] [i2nsf-terminology][ONF-SFC-Architecture].

- 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) [RFC7665]. 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.

- 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.

- Network Security Function Profile (NSF Profile): NSF Profile represents NSF’s inspection capabilities. Each NSF has its own NSF Profile to specify the type of security service it provides and its resource capacity etc.

- Network Security Function Operation Manager (NSF Operation Manager): NSF Operation Manager consistently manages information and state of NSF instances and provides NSF network access information to support advanced inspection request. For example, the information includes the supported transport protocols, IP addresses, and locations for the NSF instances. Also, NSF Operation Manager takes charge of dynamic management of a pool of NSF instances by consulting with Developer’s Management System and load balancing over NSF instances.

- Packet Forwarding Header/Encapsulation: Packet Forwarding Header is used to forward a packet from one NSF to another for further inspection. The former NSF constructs a Packet Forwarding Header with the NSF profile of the latter NSF and transmits it to a NSFF. The required fields are the action code, the number of the metadata, and the metadata. In this context, the metadata is a part of NSF profile.

- Network Security Function Forwarder (NSFF): A security function forwarder is responsible for forwarding traffic to one or more connected network security functions according to the information carried in the packet forwarding encapsulation when the traffic comes back from an NSF. Additionally, an NSFF is responsible for transporting traffic to another NSFF (in the same or the different type of overlay), and terminating overlay inspection [RFC7665].

4. Architecture

This section describes an NSF-triggered traffic steering architecture and the basic operations of traffic steering. It also includes details about each component of the architecture.

Figure 1 describes the components of NSF-triggered traffic steering architecture. Our architecture enables support a composite inspection of packets in transit. According to the inspection result of each NSF, which is stored in the Packet Forwarding Header, the traffic packets could be steered to another NSF for futher detailed analysis. It is also possible to reflect a high-level advanced inspection policy and a configuration from I2NSF User which is a component of the original I2NSF framwork. Moreover, the proposed architecture provides load balancing, auto supplementary NSF instance generation, and the elimination of unused NSF instances. In order to achieve these design purposes, we integrate several components to the original I2NSF framwork. In the following sections, we explain the details of each component.

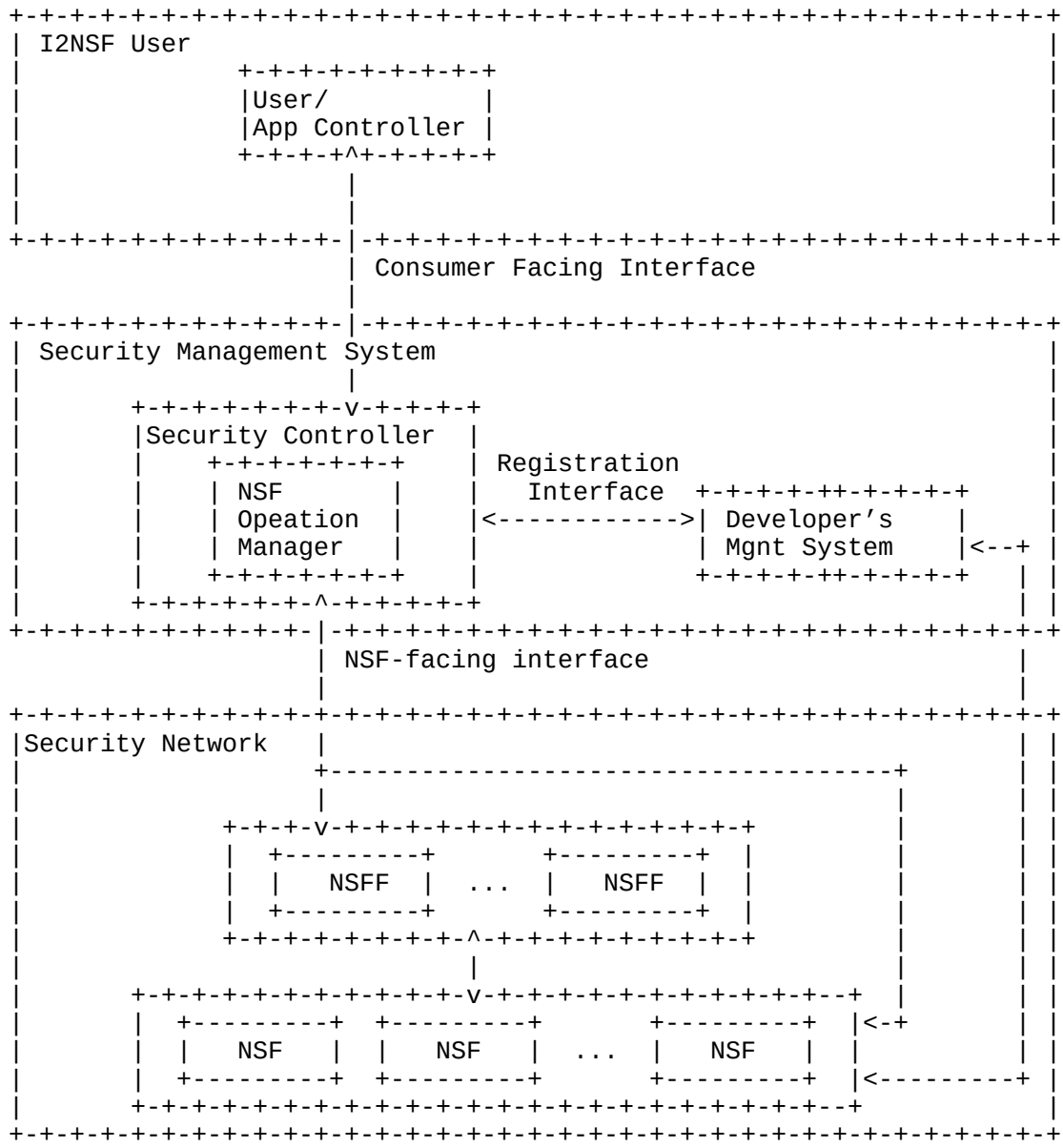


Figure 1: NSF-triggered Traffic Steering Architecture

4.1. NSF Operation Manager

NSF Operation Manager is a core component in our system. It is responsible for the following three things: (1) Maintaining the information of every available NSF instance such as IP address, supported transport protocol, NSF profile, and load status. (2) Responding the queries of available NSF instances from NSFF so as to help to conduct advanced inspection relevant to a given NSF profile. (3) Requesting Developer's Management System for the dynamic instantiation of supplementary NSF instances to avoid service congestion or the elimination of an existing NSF instance to avoid resource waste. As Figure 1 describes, NSF Operation Manager is a sub-module of Security Controller.

Whenever a new NSF instance is registered, Developer's Management System passes the information of the registered NSF instance to NSF Operation Manager, so NSF Operation Manager maintains a list of the information of every available NSF instance. NSF Operation Manger will receive the request packet containing NSF profile for advanced inspection from NSFF. Once receiving a query of a certain NSF profile from NSFF, NSF Operation Manager searches for all the available NSF instances applicable for that NSF profile and then finds the best instance with selection criteria like location and load status. After finding the best instance, it returns the search result to NSFF.

In our system, each NSF instance periodically reports its load status to NSF Operation Manager. Based on such reports, NSF Operation Manager updates the information of the NSF instances and manages the pool of NSF instances by requesting Developer's Management System for the additional instantiation or elimination of the NSF instances. Consequently, NSF Operation Manager enables efficient resource utilization by avoiding congestion and resource waste.

4.2. Developer's Management System

We extend Developer's Management System for additional functionalities as follows. As mentioned above, NSF Operation Manager requests Developer's Management System to create additional NSF instances when the existing instances of that security function are congested. On the other hand, when there are an excessive number of instances for a certain security function, NSF Operation Manager requests Developer's Management System to eliminate some of the NSF instances. As a response to such requests, Developer's Management System creates and/or removes NSF instances. Once it creates a new NSF instance or removes an existing NSF instance, the changes must be notified to NSF Operation Manager.

4.3. Network Security Function Forwarder (NSFF)

It is responsible for the following two functionalities: (1) Initially forwarding the incoming traffic/packets to Network Security Sub-Module, as described in the I2NSF information model for NSF-facing interface [capability-im]. (2) Forwarding the traffic/packets to the matched NSF with the NSF profile which is specified in a Packet Forwarding Header.

An NSFF takes a gateway functionality, so it receives incoming traffic/packets first and attaches outer encapsulation in order to forward the traffic/packets to Network Sub-Module [capability-im]. The example of Network Sub-Moudle is a firewall which performs packet header inspection. This Network Security Sub-Module attaches a Packet Forwarding Header between the outer encapsulation and the original packet and specifies NSF Profile in that header so that it can be forwarded to Content Security Sub-Module or Mitigate Sub-Module for advanced inspection.

When receiving a packet attached with a packet forwarding header of a specific NSF profile, an NSFF searches for an available NSF instance which provides the network security service corresponding to (matching with) the NSF profile and forward the packet to the NSF instance. If an NSF decides that the packet requires further inspection via another type of network security function, it constructs a packet forwarding header specified with (including) the NSF profile of the advanced network security function, attaches the header to the packet, and then sends the resulting packet to the NSFF. Once receiving the packet, the NSFF checks the NSF profile specified in the packet forwarding header. Then it searches for an NSF instance matching with the NSF profile by consulting with NSF Operation Manager, and finally forwards the packet to the NSF instance.

5. Information for Traffic Steering

This section describes the details of Packet Forwarding Header and NSF Forwarding Information for traffic forwarding between NSFs in the I2NSF system

5.1. Packet Forwarding Header

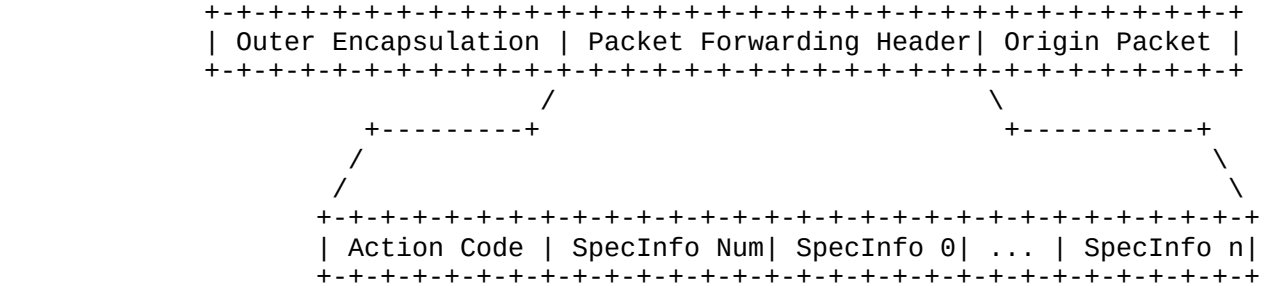


Figure 2: Packet Forwarding Header Format

An NSF uses Packet Forwarding Header to inform an NSFF(NSF Forwarder) of its inspection results and/or an advanced security inspection which is further required. As shown in Figure 2, Packet Forwarding Header consists of a single Action Code and a variable number of SpecInfo fields. The Action Code field has a value out of "allow", "deny", "advanced", and "mirror". SpecInfo Num field represents how many SpecInfos are included in this Packet Forwarding Header, and each SpecInfo includes a part of NSF Profile which describes the capabilities of an NSF required for an advanced security inspection. For instance, the value of SepcInfo can be "syn-flood-mitigate", "udp-flood-mitigate" or "content-matching-tcp" etc., which describes the service profile of an NSF.

5.2. NSF Forwarding Information

The NSF-facing interface takes charge of core functions for steering packets in the I2NSF system.

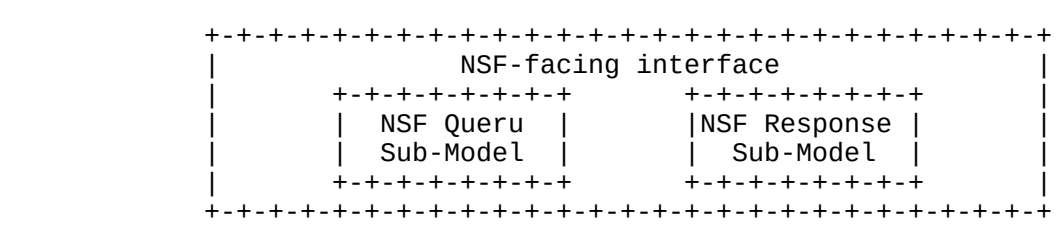


Figure 3: NSF-facing interface Model Design

5.2.1. Query of NSF forwarding information

An NSF can request an NSFF to forward packets to another NSF for more advanced security inspection of the packets. In this case, if the NSFF fails to find an NSF that can provide the security capabilities required for the advanced inspection in its forwarding information table, it sends a query to NSF Operation Manager through NSF-facing interface. The query contains an NSF profile which describes the security required for the advanced inspection capabilities. We share the definition of NSF profile in Section 4.4 of [i2nsf-reg-inf-im].

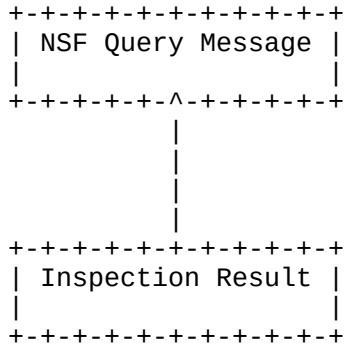


Figure 4: NSF Query Message

5.2.2. Response of NSF forwarding information

NSF Operation Manager maintains an information table of every NSF running in the system. If it receives the query from an NSFF, NSF Operation Manager searches the table for an NSF matching with the NSF profile included in the query. If there are multiple candidate NSFs, NSF Operation Manager could further consider the current workload levels of those NSFs. After choosing an NSF, it notifies the NSFF of the network forwarding information of the chosen NSF. The network forwarding information consists of IPv4 address, IPv6 address, supported transport protocols, and location information.

- IP address : As unique indetifier of an NSFs, IP address is the basic network information that allows forwarding packets to the NSF.

- Supported Transport Protocol : In order to forward packets to an NSF, it is essential to figure out which transport protocol(s) the NSF supports. Examples of the transport protocols are as follows: Virtual Extensible LAN (VXLAN) [RFC7348], Generic Protocol Extension for VXLAN (VXLAN-GPE) [nvo3-vxlan-gpe], Generic Route

Encapsulation (GRE), Ethernet). An NSFF performs encapsulation of the packets to forward as defined in the transport protocol.

- Location Information : NSFs in the system can be distributed over a wide physical region. Unlike IP address, location information specifies the physical location of an NSF. Thus, NSF Operation Manager can consider physical proximity as an additional factor to select an NSF.

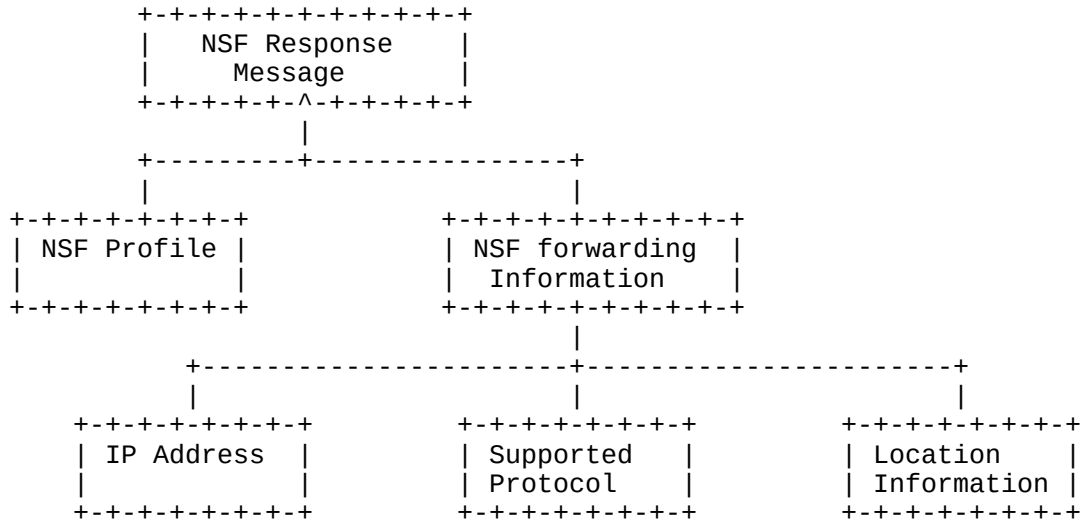


Figure 5: NSF Response Message

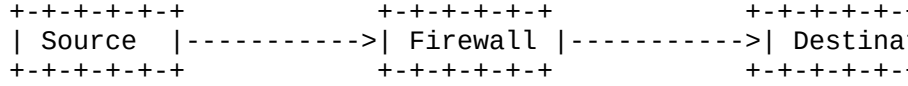
6. Use Cases

This section introduces two use cases for the NSF-triggered Traffic Steering Framework: (1) Enforcing Different NSFs Depending on a Packet Source’s Trust Level, (2) Effective Load Balancing with Dynamic NSF Instantiation.

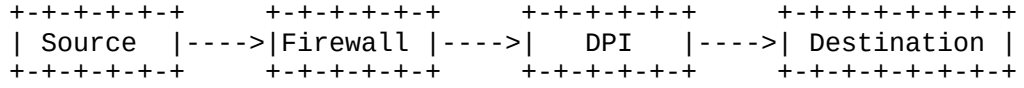
6.1. Enforcing Different NSFs Depending on a Packet Source’s Trust Level

In the proposed architecture, all incoming packets initially arrive at the NSFF. We assume that the current security policy forces all incoming packets to be by default inspected by a firewall in this scenario. Thus the NSFF forwards the received packets to a firewall instance. Then the firewall identifies the source of the traffic and evaluates the trust level of the source. If the traffic comes from a

trusted source, it is likely to be benign. In this case, the traffic is just forwarded to the destination without further detailed inspection via different types of security functions as illustrated in Figure 6-(a). Otherwise if the traffic comes from an untrusted source, the firewall attaches a packet forwarding header including the NSF profile corresponding to DPI to the packet and returns the resulting packet to the NSFF. Once receiving the packet, the NSFF forwards the packet to the DPI instance which will perform detailed inspection for the packet payload. Figure 6-(b) illustrates this case.



(a) Traffic flow of trusted source



(b) Traffic flow of untrusted source

Figure 6: Different path allocation depending on source of traffic

6.2. Effective Load Balancing with Dynamic NSF Instantiation

In a large-scale network domain, there typically exist a large number of NSF instances that provide various security services. It is possible that a specific NSF instance experiences an excessive amount of traffic beyond its capacity. In this case, it is required to allocate some of the traffic to another available instance of the same security function. If there are no additional instances of the same security function available, we need to create a new NSF instance and then direct the subsequent traffic to the new instance. In this way, we can avoid service congestion and achieve more efficient resource utilization.

This process is commonly called load balancing. In our proposed architecture, NSF Operation Manager performs periodic monitoring of the load status of available NSF instances. In addition, it is possible to dynamically generate a new NSF instance through Developer’s Management System. With these functionalities along with the flexible traffic steering mechanism, we can eventually provide load balancing service.

The following describes the detailed process of load balancing when congestion occurs at the firewall instance:

1. NSF Operation Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.

2. NSF Operation Manager requests Developer's Management System to create a new firewall instance.

3. Developer's Management System creates a new firewall instance and then registers the information of the new firewall instance to NSF Operation Manager.

4. NSF Operation Manager updates the SFC Information Table to reflect the new firewall instance, and notifies NSF and NSFF of this update.

5. According to the new forwarding information, the NSFF forwards the subsequent traffic to the new firewall instance. As a result, we can effectively alleviate the burden of the existing firewall instance.

7. Security Considerations

To enable security function chaining in the I2NSF framework, we adopt the additional components in the SFC architecture. Thus, this document shares the security considerations of the SFC architecture that are specified in [RFC7665] for the purpose of achieving secure communication among components in the proposed architecture.

8. Acknowledgements

This work was supported by Institute for Information and 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. References

9.1. Normative References

[RFC7665] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7665, March 2014.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

[sfc-ns-use-cases] Wang, E., Leung, K., Felix, J., and J. Iyer, "Service Function Chaining Use Cases for Network Security", draft-wang-sfc-ns-use-cases-02 , October 2016.

9.2. Informative References

[RFC7498] Quinn, P. and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, April 2015.

[capability-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01 (work in progress), March 2017.

[i2nsf-reg-inf-im] Hyun, S., Woo, S., Yeo, Y., Jeong, J., and J. Park, "Registration Interface Information Model", draft-hyun-i2nsf-registration-interface-im-01 (work in progress), March 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-04 (work in progress), October 2016.

[i2nsf-problem] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-11 (work in progress), March 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-03 (work in progress), March 2017.

[ONF-SFC-Architecture] ONF, "L4-L7 Service Function Chaining Solution Architecture", June 2015.

[nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-03 (work in progress), October 2016.

Appendix A. Changes from draft-hyun-i2nsf-nsf-triggered-steering-01

The following changes are made from draft-hyun-i2nsf-nsf-triggered-steering-01:

- Section 5 is added to explain more details of Packet Forwarding Header and NSF Forwarding Information than the previous version.
- In Section 5.1, the explanation of Packet Forwarding Header is clarified more clearly.
- In Section 5.2, we specify the details of NSF Forwarding Information.

Authors' Addresses

Sangwon Hyun
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: swhyun77@skku.edu
URI: http://imtl.skku.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

SangUk Woo
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: suwoo@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

YunSuk Yeo
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: yunsuk@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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: September 14, 2017

S. Hyun
J. Jeong
S. Woo
Y. Yeo
Sungkyunkwan University
J. Park
ETRI

March 13, 2017

Registration Interface Information Model for Interface to Network Security Functions
draft-hyun-i2nsf-registration-interface-im-01

Abstract

This document describes an information model for Interface to Network Security Functions (I2NSF) Registration Interface between Security Controller and Developer's Management System. The information model is required for Network Security Function (NSF) instance registration and the dynamic life cycle management of NSF instances. 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 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/1id-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 September 14, 2017.

Table of Contents

1. Introduction

A number of virtual network security function instances typically exist in Interface to Network Security Functions (I2NSF) framework [i2nsf-framework]. In this environment, it is important to dynamically manage a Network Security Function (NSF) instance pool for efficient resource utilization. For instance, if a certain NSF instance is receiving an excessive amount of traffic beyond its capacity, an additional instance for the same security function should be created. If an NSF instance is idle for a period of time, it would be better to destroy it to avoid resource waste. In addition, the existing information model for NSF-facing interface requires an NSF to trigger another type of NSF for further inspection [capability-im]. In this case, if there is no available instance for the latter NSF, a new NSF should be instantiated. Similarly, in order to enforce a security policy from the client, all the required NSF instances should be created.

This document describes the procedures which should be performed on the registration interface between security controller and developer's management system to dynamically manage a pool of NSF instances. It further describes the detailed information which should be exchanged between security controller and developer's management system.

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][i2nsf-framework] [nsf-triggered-steering].

- 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].

- 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].

- 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 resource capacity. [nsf-triggered-steering].

4. Objectives

- Efficient network resource utilization through dynamic instantiation of NSFs and load balancing: In I2NSF framework, it is sometimes possible that a specific NSF experiences heavy traffic loads. For example, under DDoS attacks, a huge volume of traffic would be delivered to DoS attack mitigator function to cope with the attacks. In this case, we should allocate a large portion of resources to that DoS attack mitigator function by creating a sufficient number of DoS mitigator instances. In addition, after the attack is terminated, we should eliminate some of the instances no longer used. In this way, we can achieve efficient resource utilization. For this purpose, it is essential to define an information model of registration interface for dynamic instantiation/elimination of NSF instances.

- 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.

- 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.

- 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.

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 dynamic NSF life cycle management 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.

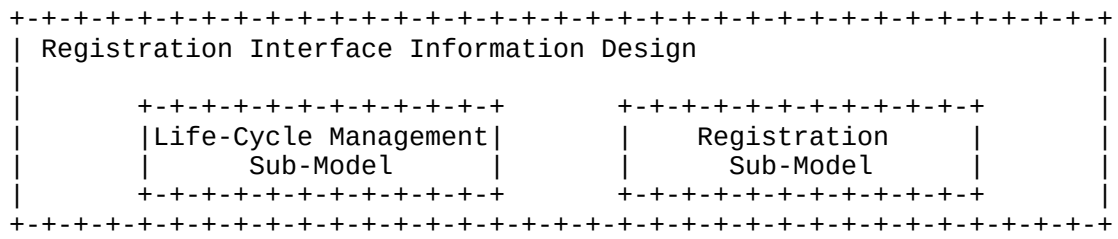


Figure 1: The Registration Interface Information Model Design

As illustrated in Figure 1, the information model for Registration Interface consists of two sub-models: life-cycle management, registration sub-models. The life-cycle 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. Life-Cycle Managment Mechanism

For the life-cycle management of NSFs, Security Controller in I2NSF framework requires two types of requests: Instance Creation and Elimination Request Messages. 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. There are several cases requiring creation of a new NSF instance which provides specific security inspection functionalities and elimination of an existing NSF which is unused for a period of time. For example,

1) When an NSF triggers an advanced inspection of the suspicious traffic via another type of NSF whose instance is currently unavailable in the system.

2) When an NSF instance undergoes an excessive amount of traffic

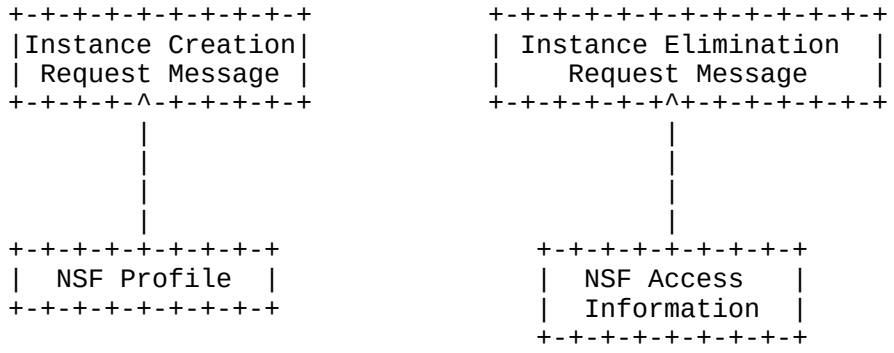


Figure 2: Life-Cycle Management Sub-Model Overview

5.2. 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.

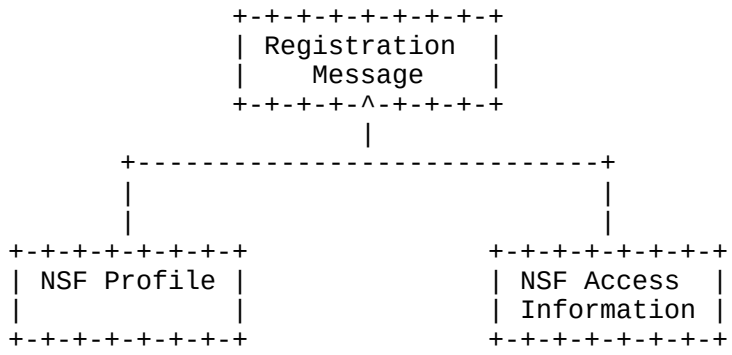


Figure 3: Registration Mechanism Sub-Model Overview

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) [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 Profile (Capabilities of an NSF instance)

NSF Profile basically refers the inspection capabilities of an NSF instance. As illustrated in Figure 4, it can be split into five capabilities (Content-Matching, Context-Matching, Attack-Mitigation, Action, Performance Capabilities). We share the security capabilities which are defined in Section 3 (Overall Analysis of Security Capability) in [capability-im] for the first five capabilities and append one additional capability.

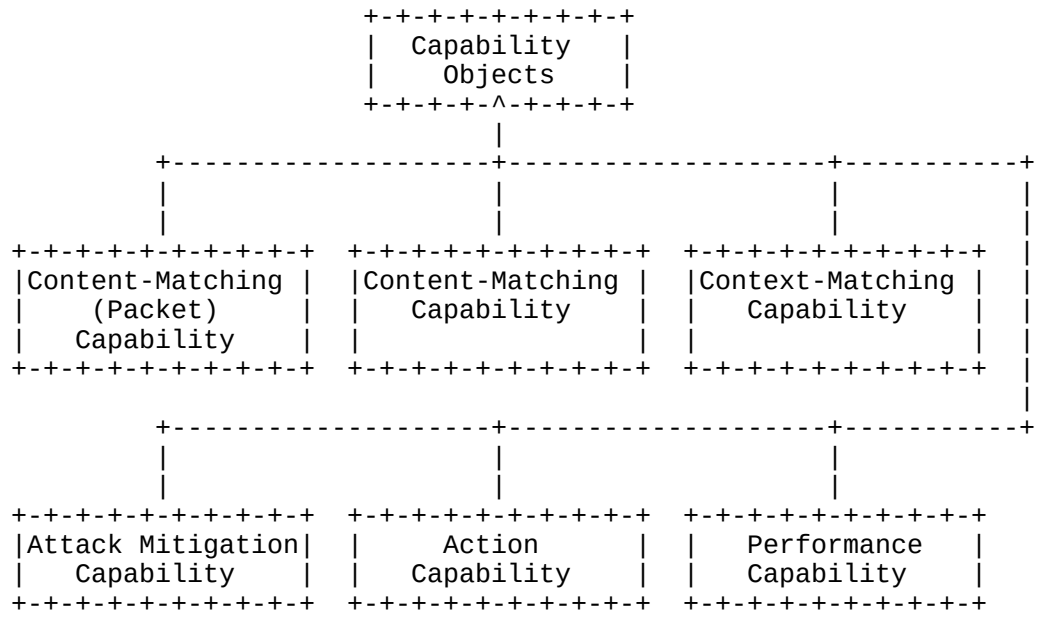


Figure 4: NSF Profile Overview

5.4.1. Packet Content-Matching Capability

Refer to the kind of information or attributes acquired directly from the packet headers or payloads that can be used in the security policy. It can be any fields or attributes in the packet L2/L3/L4 header, or special segment of bytes in the packet payload. [capability-im]

5.4.2. Content-Matching Capability

Content security 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 functions, such as defending against intrusion, inspecting virus, filtering malicious URL or junk email, blocking illegal web access or malicious data retrieval. [capability-im]

5.4.3. Context-Matching Capability

This capability refers to the content information for the received packets. It can be User, Schedule, Region, Target, State and Direction information. [capability-im]

5.4.4. Attack-Mitigation Capability

This category of security capabilities is used to detect and mitigate various types of network attacks. Today's common network attacks can be classified into the following sets, and each set further consists of a number of specific attacks: [capability-im]

- DDoS attacks:
  - Network layer DDoS attacks: Examples include SYN flood, UDP flood, ICMP flood, IP fragment flood, IPv6 Routing header attack, and IPv6 duplicate address detection attack;
  - Application layer DDoS attacks: Examples include http flood, https flood, cache-bypass http floods, WordPress XML RPC floods, ssl DDoS.
- Single-packet attack:
  - Scanning and sniffing attacks: IP sweep, port scanning, etc
  - Malformed packet attacks: Ping of Death, Teardrop, etc
  - Special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc

5.4.5. Action Capability

NSFs provide security functions by executing various Actions, which at least includes: [capability-im]

- Ingress actions, such as pass, drop, mirroring, etc;
- Egress actions, such as invoke signaling, tunnel encapsulation, packet forwarding and/or transformation;
- Applying a specific Functional Profile or signature (NSF Profile) - The functional profile or signature file defines the security capabilities for content security control and/or attack mitigation control. One goal of I2NSF is to standardize the form and functional interface of those security capabilities while supporting vendor-specific implementations of each.

5.4.6. Performance Capability

This information basically represents the processing capability of an NSF, i.e., the amount of traffic it can process for a unit time period. 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 and memory which are available on the NSF.

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 framwork that are specified in [i2nsf-framework] for the purpose of achieving secure communication between components in the proposed architecture.

7. Acknowledgements

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

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-xibassnez-i2nsf-capability-01 (work in progress), March 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-04 (work in progress), October 2016.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.

[nsf-triggered-steering] Hyun, S., Jeong, J., Woo, S., Yeo, Y., and J. Park, "NSF-Triggered Traffic Steering", draft-hyun-i2nsf-nsf-triggered-steering-02 (work in progress), March 2017.

[nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-03 (work in progress), October 2016.

Appendix A. Changes from draft-hyun-i2nsf-registration-interface-im-00

The following changes are made from draft-hyun-i2nsf-registration-interface-im-00:

- Miscellaneous expressions in the whole descriptions are corrected.
- The description of NSF Access Information and Performance Capability is specified in more detail than the previous version.

Authors' Addresses

Sangwon Hyun
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: swhyun77@skku.edu
URI: http://imtl.skku.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

SangUk Woo
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: suwoo@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

YunSuk Yeo
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: yunsuk@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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
Internet-Draft
Intended status: Informational
Expires: February 25, 2018

D. Lopez
Telefonica I+D
E. Lopez
Curveball Networks
L. Dunbar
J. Strassner
Huawei
R. Kumar
Juniper Networks

August 25, 2017

Framework for Interface to Network Security Functions
draft-ietf-i2nsf-framework-07

Abstract

This document describes the framework for the Interface to Network Security Functions (I2NSF), and defines a reference model (including major functional components) for I2NSF. Network security functions (NSFs) are packet-processing engines that inspect and optionally modify packets traversing networks, either directly or in the context of sessions to which the packet is associated.

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 February 25, 2018.

Table of Contents

1. Introduction

This document describes the framework for the Interface to Network Security Functions (I2NSF), and defines a reference model (including major functional components) for I2NSF. This includes an analysis of the threats implied by the deployment of Network Security Functions (NSFs) that are externally provided. It also describes how I2NSF facilitates implementing security functions in a technology- and vendor-independent manner in Software-Defined Networking (SDN) and Network Function Virtualization (NFV) environments, while avoiding potential constraints that could limit the capabilities of NSFs.

The I2NSF use cases [RFC8192] call for standard interfaces for users of an I2NSF system (e.g., applications, overlay or cloud network management system, or enterprise network administrator or management system), to inform the I2NSF system which I2NSF functions should be applied to which traffic (or traffic patterns). The I2NSF system realizes this as a set of security rules for monitoring and controlling the behavior of different traffic. It also provides standard interfaces for users to monitor flow-based security functions hosted and managed by different administrative domains.

[RFC8192] also describes the motivation and the problem space for an Interface to Network Security Functions system.

2. Conventions used in this document

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.

Note: as this is an informational document, no RFC-2119 key words are used.

2.1. Acronyms

The following acronyms are used in this document:

DOTS	Distributed Denial-of-Service Open Threat Signaling
IDS	Intrusion Detection System
IoT	Internet of Things
IPS	Intrusion Protection System
NSF	Network Security Function

2.2. Definitions

The following terms, which are used in this document, are defined in the I2NSF terminology document [I-D.ietf-i2nsf-terminology]:

Capability

Controller

Firewall

I2NSF Consumer

I2NSF NSF-Facing Interface

I2NSF Policy Rule

I2NSF Producer

I2NSF Registration Interface

I2NSF Registry

Interface

Interface Group

Intrusion Detection System

Intrusion Protection System

Network Security Function

Role

3. I2NSF Reference Model

Figure 1 shows a reference model (including major functional components and interfaces) for an I2NSF system. This figure is drawn from the point-of-view of the Network Operator Management System; hence, this view does not assume any particular management architecture for either the NSFs or for how NSFs are managed (on the developer’s side). In particular, the Network Operator Management System does not participate in NSF data plane activities.

Note that the term "Controller" is defined in [I-D.ietf-i2nsf-terminology].

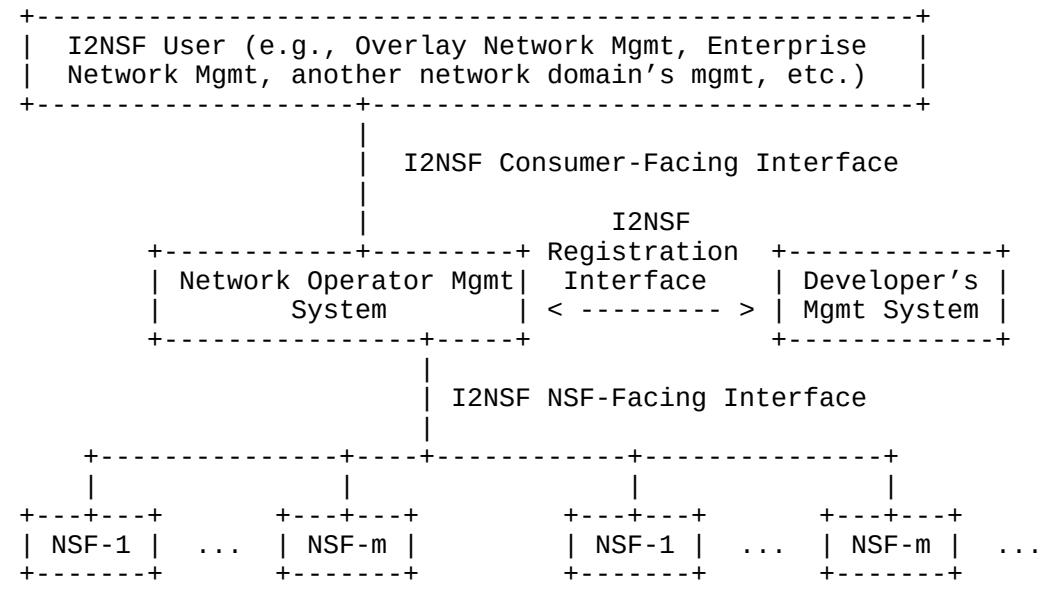


Figure 1: I2NSF Reference Model

When defining I2NSF interfaces, this framework adheres to the following principles:

- Agnostic of network topology and NSF location in the network
- Agnostic of provider of the NSF (i.e., independent of the way that the provider makes an NSF available, as well as how the provider allows the NSF to be managed)
- Agnostic of any vendor-specific operational, administrative, and management implementation, hosting environment, and form-factor (physical or virtual)
- Agnostic to NSF control plane implementation (e.g., signaling capabilities)
- Agnostic to NSF data plane implementation (e.g., encapsulation capabilities)

3.1. I2NSF Consumer-Facing Interface

The I2NSF Consumer-Facing Interface is used to enable different users of a given I2NSF system to define, manage, and monitor security policies for specific flows within an administrative domain. The location and implementation of I2NSF policies are irrelevant to the consumer of I2NSF policies.

Some examples of I2NSF Consumers include:

- A videoconference network manager that needs to dynamically inform the underlay network to allow, rate-limit, or deny flows (some of which are encrypted) based on specific fields in the packets for a certain time span.

- Enterprise network administrators and management systems that need to request their provider network to enforce specific I2NSF policies for particular flows.

- An IoT management system sending requests to the underlay network to block flows that match a set of specific conditions.

3.2. I2NSF NSF-Facing Interface

The I2NSF NSF-Facing Interface (NSF-Facing Interface for short) is used to specify and monitor flow-based security policies enforced by one or more NSFs. Note that the I2NSF Management System does not need to use all features of a given NSF, nor does it need to use all available NSFs. Hence, this abstraction enables NSF features to be treated as building blocks by an NSF system; thus, developers are free to use the security functions defined by NSFs independent of vendor and technology.

Flow-based NSFs [RFC8192] inspect packets in the order that they are received. Note that all Interface Groups require the NSF to be registered using the Registration Interface. The Interface to flow-based NSFs can be categorized as follows:

1. NSF Operational and Administrative Interface: an Interface Group used by the I2NSF Management System to program the operational state of the NSF; this also includes administrative control functions. I2NSF Policy Rules represent one way to change this Interface Group in a consistent manner. Since applications and I2NSF Components need to dynamically control the behavior of traffic that they send and receive, much of the I2NSF effort is focused on this Interface Group.

2. Monitoring Interface: an Interface Group used by the the I2NSF Management System to obtain monitoring information from one or more selected NSFs. Each interface in this Interface Group could be a query- or a report-based interface. The difference is that a query-based interface is used by the the I2NSF Management System to obatin information, whereas a report-based interface is used by the NSF to provide information. The functionality of this Interface Group may also be defined by other protocols, such as SYSLOG and DOTS (Distributed Denial-of-Service Open Threat Signaling). The I2NSF Management System may take one or more actions based on the receipt of information; this should be specified by an I2NSF Policy Rule. This Interface Group does NOT change the operational state of the NSF.

This document uses the flow-based paradigm to develop the NSF-Facing Interface. A common trait of flow-based NSFs is in the processing of packets based on the content (e.g., header/payload) and/or context (e.g., session state, authentication state) of the received packets. This feature is one of the requirements for defining the behavior of I2NSF.

3.3. I2NSF Registration Interface

NSFs provided by different vendors may have different capabilities. In order to automate the process of utilizing multiple types of security functions provided by different vendors, it is necessary to have a dedicated interface for vendors to define the capabilities of (i.e., register) their NSFs. This Interface is called the I2NSF Registration Interface.

An NSF’s capabilities can either be pre-configured or retrieved dynamically through the I2NSF Registration Interface. If a new function that is exposed to the consumer is added to an NSF, then the capabilities of that new function should be registered in the I2NSF Registry via the I2NSF Registration Interface, so that interested management and control entities may be made aware of them.

4. Threats Associated with Externally Provided NSFs

While associated with a much higher flexibility, and in many cases a necessary approach given the deployment conditions, the usage of externally provided NSFs implies several additional concerns in security. The most relevant threats associated with a security platform of this nature are:

- An unknown/unauthorized user can try to impersonate another user that can legitimately access external NSF services. This attack may lead to accessing the I2NSF Policy Rules and applications of the attacked user, and/or to generate network traffic outside the security functions with a falsified identity.

- An authorized user may misuse assigned privileges to alter the network traffic processing of other users in the NSF underlay or platform.

- A user may try to install malformed elements (e.g., I2NSF Policy Rules, or configuration files), trying to directly take the control of a NSF or the whole provider platform. For example, a user may exploit a vulnerability on one of the functions, or may try to intercept or modify the traffic of other users in the same provider platform.

- A malicious provider can modify the software (e.g., the operating system or the specific NSF implementation) to alter the behavior of one or more NSFs. This event has a high impact on all users accessing NSFs, since the provider has the highest level of privileges controlling the operation of the software.

- A user that has physical access to the provider platform can modify the behavior of the hardware/software components, or the components themselves. For example, the user can access a serial console (most devices offer this interface for maintenance reasons) to access the NSF software with the same level of privilege of the provider.

The above threats may be mitigated by requiring the use of an AAA framework for all users to access the I2NSF environment. This could be further enhanced by requiring attestation to be used to detect changes to the I2NSF environment by authorized parties.

5. Avoiding NSF Ossification

An important concept underlying this framework is the fact that attackers do not have standards as to how to attack networks, so it is equally important to not constrain NSF developers to offering a limited set of security functions. In other words, the introduction of I2NSF standards should not make it easier for attackers to compromise the network. Therefore, in constructing standards for I2NSF Interfaces as well as I2NSF Policy Rules, it is equally important to allow support for specific functions, as this enables the introduction of NSFs that evolve to meet new threats. Proposed standards for I2NSF Interfaces to communicate with NSFs, as well as I2NSF Policy Rules to control NSF functionality, should not:

- Narrowly define NSF categories, or their roles, when implemented within a network

- Attempt to impose functional requirements or constraints, either directly or indirectly, upon NSF developers

- o Be a limited lowest common denominator approach, where interfaces can only support a limited set of standardized functions, without allowing for developer-specific functions

- o Be seen as endorsing a best common practice for the implementation of NSFs

To prevent constraints on NSF developers' creativity and innovation, this document recommends the Flow-based NSF interfaces to be designed from the paradigm of processing packets in the network. Flow-based NSFs ultimately are packet-processing engines that inspect packets traversing networks, either directly or in the context of sessions in which the packet is associated. The goal is to create a workable interface to NSFs that aids in their integration within legacy, SDN, and/or NFV environments, while avoiding potential constraints which could limit their functional capabilities.

6. The Network Connecting I2NSF Components

6.1. Network Connecting I2NSF Users and the I2NSF Controller

As a general principle, in the I2NSF environment, users directly interact with the I2NSF Controller. Given the role of the I2NSF Controller, a mutual authentication of users and the I2NSF Controller may be required. I2NSF does not mandate a specific authentication scheme; it is up to the users to choose available authentication schemes based on their needs.

Upon successful authentication, a trusted connection between the user and the I2NSF Controller (or an endpoint designated by it) will be established. All traffic to and from the NSF environment will flow through this connection. The connection is intended not only to be secure, but trusted in the sense that it should be bound to the mutual authentication between the user and the I2NSF Controller, as described in [I-D.pastor-i2nsf-remote-attestation]. The only possible exception is when the required level of assurance is lower, (see Section 4.1 of [I-D.pastor-i2nsf-remote-attestation]), in which case the user must be made aware of this circumstance.

6.2. Network Connecting the I2NSF Controller and NSFs

Most likely the NSFs are not directly attached to the I2NSF Controller; for example, NSFs can be distributed across the network. The network that connects the I2NSF Controller with the NSFs can be the same network that carries the data traffic, or can be a dedicated network for management purposes only. In either case, packet loss could happen due to failure, congestion, or other reasons.

Therefore, the transport mechanism used to carry the control messages and monitoring information should provide reliable message delivery. Transport redundancy mechanisms such as Multipath TCP (MPTCP) and the Stream Control Transmission Protocol (SCTP) will need to be evaluated for applicability. Latency requirements for control message delivery must also be evaluated.

The network connection between the I2NSF Controller and NSFs can rely either on:

- Closed environments, where there is only one administrative domain. Less restrictive access control and simpler validation can be used inside the domain because of the protected nature of a closed environment.

- Open environments, where one or more NSFs can be hosted in one or more external administrative domains that are reached via secure external network connections. This requires more restrictive security control to be placed over the I2NSF interface. The information over the I2NSF interfaces shall be exchanged by using the trusted connection described in section 6.1.

When running in an open environment, I2NSF needs to rely on the use of standard I2NSF interfaces to properly verify peer identities (e.g., through an AAA framework). The implementations of identity management functions, as well as the AAA framework, are out of scope for I2NSF.

6.3. Interface to vNSFs

There are some unique characteristics in interfacing to virtual NSFs:

- There could be multiple instantiations of one single NSF that has been distributed across a network. When different instantiations are visible to the I2NSF Controller, different policies may be applied to different instantiations of an individual NSF (e.g., to reflect the different roles that each vNSF is designated for). Therefore, it is recommended that Roles, in addition to the use of robust identities, be used to distinguish between different instantiations of the same vNSF. Note that this also applies to physical NSFs.

- When multiple instantiations of one single NSF appear as one single entity to the I2NSF Controller, the I2NSF Controller may need to either get assistance from other entities in the I2NSF Management System, and/or delegate the provisioning of the multiple instantiations of the (single) NSF to other entities in the I2NSF Management System. This is shown in Figure 2 below.

- Policies enforced by one vNSF instance may need to be retrieved and moved to another vNSF of the same type when user flows are moved from one vNSF to another.

- Multiple vNSFs may share the same physical platform.

- There may be scenarios where multiple vNSFs collectively perform the security policies needed.

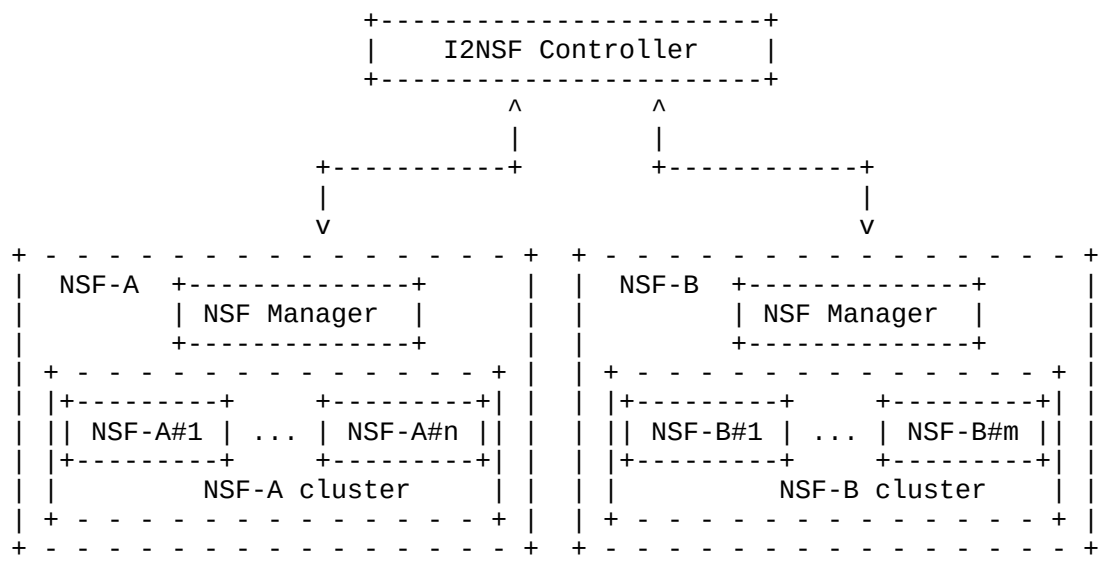


Figure 2: Cluster of NSF Instantiations Management

6.4. Consistency

There are three basic models of consistency:

- centralized, which uses a single manager to impose behavior
- decentralized, in which managers make decisions without being aware of each other (i.e., managers do not exchange information)
- distributed, in which managers make explicit use of information exchange to arrive at a decision

This document does NOT make a recommendation on which of the above three models to use. I2NSF Policy Rules, coupled with an appropriate management strategy, is applicable to the design and integration of any of the above three consistency models.

7. I2NSF Flow Security Policy Structure

Even though security functions come in a variety of form factors and have different features, provisioning to flow-based NSFs can be standardized by using policy rules.

In this version of I2NSF, policy rules are limited to imperative paradigms. I2NSF is using an Event - Condition - Action (ECA) policy, where:

- An Event clause is used to trigger the evaluation of the Condition clause of the Policy Rule.
- A Condition clause is used to determine whether or not the set of Actions in the I2NSF Policy Rule can be executed or not.
- An Action clause defines the type of operations that may be performed on this packet or flow.

Each of the above three clauses are defined to be Boolean clauses. This means that each is a logical statement that evaluates to either TRUE or FALSE.

The above concepts are described in detail in [I-D.draft-xibassnez-i2nsf-capability].

7.1. Customer-Facing Flow Security Policy Structure

This layer is for the user’s network management system to express and monitor the needed flow security policies for their specific flows.

Some customers may not have the requisite security skills to express security requirements or policies that are precise enough to implement in an NSF. These customers may instead express expectations (e.g., goals, or intent) of the functionality desired by their security policies. Customers may also express guidelines, such as which types of destinations are (or are not) allowed for certain users. As a result, there could be different levels of content and abstractions used in Service Layer policies. Here are some examples of more abstract security Policies that can be developed based on the I2NSF defined customer-facing interface:

Pass for Subscriber "xxx"

Enable basic parental control

Enable "school protection control"

Allow Internet traffic from 8:30 to 20:00

Scan email for malware detection protect traffic to corporate network with integrity and confidentiality

Remove tracking data from Facebook [website = *.facebook.com]

My son is allowed to access Facebook from 18:30 to 20:00

One flow policy over the Customer-Facing Interface may need multiple NSFs at various locations to achieve the desired enforcement. Some flow security policies from users may not be granted because of resource constraints. [I-D.xie-i2nsf-demo-outline-design] describes an implementation of translating a set of user policies to the flow policies to individual NSFs.

I2NSF will first focus on user policies that can be modeled as closely as possible to the flow security policies used by individual NSFs. An I2NSF user flow policy should be similar in structure to the structure of an I2NSF Policy Rule, but with more of a user-oriented expression for the packet content, context, and other parts of an ECA policy rule. This enables the user to construct an I2NSF Policy Rule without having to know the exact syntax of the desired content (e.g., actual tags or addresses) to match in the packets. For example, when used in the context of policy rules over the Client Facing Interface:

An Event can be "the client has passed the AAA process"

A Condition can be matching user identifier, or from specific ingress or egress points

An action can be establishing a IPsec tunnel

7.2. NSF-Facing Flow Security Policy Structure

The NSF-Facing Interface is to pass explicit rules to individual NSFs to treat packets, as well as methods to monitor the execution status of those functions.

Here are some examples of events over the NSF facing interface:

time == 08:00

a NSF state change from standby to active

Here are some examples of conditions over the NSF facing interface

- Packet content values are based on one or more packet headers, data from the packet payload, bits in the packet, or data that are derived from the packet.

- Context values are based on measured and inferred knowledge that define the state and environment in which a managed entity exists or has existed. In addition to state data, this includes data from sessions, direction of the traffic, time, and geo-location information. State refers to the behavior of a managed entity at a particular point in time. Hence, it may refer to situations in which multiple pieces of information that are not available at the same time must be analyzed. For example, tracking established TCP connections (connections that have gone through the initial three-way handshake).

Actions to individual flow-based NSFs include:

- Action ingress processing, such as pass, drop, rate limiting, and mirroring.

- Action egress processing, such as invoke signaling, tunnel encapsulation, packet forwarding and/or transformation.

- Applying a specific functional profile or signature - e.g., an IPS Profile, a signature file, an anti-virus file, or a URL filtering file. Many flow-based NSFs utilize profile and/or signature files to achieve more effective threat detection and prevention. It is not uncommon for a NSF to apply different profiles and/or signatures for different flows. Some profiles/signatures do not require any knowledge of past or future activities, while others are stateful, and may need to maintain state for a specific length of time.

The functional profile or signature file is one of the key properties that determine the effectiveness of the NSF, and is mostly NSF-specific today. The rulesets and software interfaces of I2NSF aim to specify the format to pass profile and signature files while supporting specific functionalities of each.

Policy consistency among multiple security function instances is very critical because security policies are no longer maintained by one central security device, but instead are enforced by multiple security functions instantiated at various locations.

7.3. Differences from ACL Data Models

Policy rules are very different from ACLs. An ACL is NOT a policy. Rather, policies are used to manage the construction and lifecycle of an ACL.

[I-D.ietf-netmod-acl-model] has defined rules for the Access Control List supported by most routers/switches that forward packets based on packets' L2, L3, or sometimes L4 headers. The actions for Access Control Lists include Pass, Drop, or Redirect.

The functional profiles (or signatures) for NSFs are not present in [I-D.ietf-netmod-acl-model] because the functional profiles are unique to specific NSFs. For example, most IPS/IDS implementations have their proprietary functions/profiles. One of the goals of I2NSF is to define a common envelop format for exchanging or sharing profiles among different organizations to achieve more effective protection against threats.

The "packet content matching" of the I2NSF policies should not only include the matching criteria specified by [I-D.ietf-netmod-acl-model], but also the L4-L7 fields depending on the NSFs selected.

Some Flow-based NSFs need matching criteria that include the context associated with the packets. This may also include metadata.

The I2NSF "actions" should extend the actions specified by [I-D.ietf-netmod-acl-model] to include applying statistics functions, threat profiles, or signature files that clients provide.

8. Capability Negotiation

It is very possible that the underlay network (or provider network) does not have the capability or resource to enforce the flow security policies requested by the overlay network (or enterprise network). Therefore, it is required that the I2NSF system support dynamic discovery capabilities, as well as a query mechanism, so that the I2NSF system can explose appropriate security services using I2NSF capabilities. This may also be used to support negotiation between a user and the I2NSF system. Such dynamic negotiation facilitates the delivery of the required security service(s). The outcome of the negotiation would feed the I2NSF Management System, which would then dynamically allocate appropriate NSFs (along with any resources needed by the allocated NSFs) and configure the set of security services that meet the requirements of the user.

When an NSF cannot perform the desired provisioning (e.g., due to resource constraints), it must inform the I2NSF Management System.

The protocol needed for this security function/capability negotiation may be somewhat correlated to the dynamic service parameter negotiation procedure described in [RFC7297]. The Connectivity Provisioning Profile (CPP) template, even though currently covering only Connectivity requirements, includes security clauses such as isolation requirements and non-via nodes. Hence, it could be extended as a basis for the negotiation procedure. Likewise, the companion Connectivity Provisioning Negotiation Protocol (CPNP) could be a candidate for the negotiation procedure.

"Security-as-a-Service" would be a typical example of the kind of (CPP-based) negotiation procedures that could take place between a corporate customer and a service provider. However, more security specific parameters have to be considered.

[I.D.-draft-xibassnez-i2nsf-capability] describes the concepts of capabilities in detail.

9. Registration Considerations

9.1. Flow-Based NSF Capability Characterization

There are many types of flow-based NSFs. Firewall, IPS, and IDS are the commonly deployed flow-based NSFs. However, the differences among them are definitely blurring, due to more powerful technology, integration of platforms, and new threats. Basic types of flow-based NSFs include:

- Firewall - A device or a function that analyzes packet headers and enforces policy based on protocol type, source address, destination address, source port, destination port, and/or other attributes of the packet header. Packets that do not match policy are rejected. Note that additional functions, such as logging and notification of a system administrator, could optionally be enforced as well.

- IDS (Intrusion Detection System) - A device or function that analyzes packets, both header and payload, looking for known events. When a known event is detected, a log message is generated detailing the event. Note that additional functions, such as notification of a system administrator, could optionally be enforced as well.

- IPS (Intrusion Prevention System) - A device or function that analyzes packets, both header and payload, looking for known events. When a known event is detected, the packet is rejected. Note that additional functions, such as logging and notification of a system administrator, could optionally be enforced as well.

Flow-based NSFs differ in the depth of packet header or payload they can inspect, the various session/context states they can maintain, and the specific profiles and the actions they can apply. An example of a session is "allowing outbound connection requests and only allowing return traffic from the external network".

9.2. Registration Categories

Developers can register their NSFs using Packet Content Match categories. The IDR (Inter-Domain Routing) Flow Specification [RFC5575] has specified 12 different packet header matching types. More packet content matching types have been proposed in the IDR WG. I2NSF should re-use the packet matching types being specified as much as possible. More matching types might be added for Flow-based NSFS. Tables 1-4 below list the applicable packet content categories that can be potentially used as packet matching types by Flow-based NSFs:

Packet Content Matching Capability Index	
Layer 2 Header	Layer 2 header fields: Source/Destination/s-VID/c-VID/EtherType/.
Layer 3	Layer header fields:
IPv4 Header	protocol dest port src port src address dest address dscp length flags ttl
IPv6 Header	addr protocol/nh src port dest port src address dest address length traffic class hop limit flow label dscp
TCP SCTP DCCP	Port syn ack fin rst ? psh ? urg ? window sockstress
	Note: bitmap could be used to represent all the fields

UDP		
	flood abuse	
	fragment abuse	
	Port	
HTTP layer		
	hash collision	
	http - get flood	
	http - post flood	
	http - random/invalid url	
	http - slowloris	
	http - slow read	
	http - r-u-dead-yet (rudy)	
	http - malformed request	
	http - xss	
	https - ssl session exhaustion	
IETF PCP	Configurable Ports	
IETF TRAM	profile	

Table 1: Packet Content Matching Capability Index

Notes: DCCP: Datagram Congestion Control Protocol
PCP: Port Control Protocol
TRAM: TURN Revised and Modernized, where TURN stands for Traversal Using Relays around NAT

Context Matching Capability Index	
Session	Session state, bidirectional state
Time	time span time occurrence
Events	Event URL, variables
Location	Text string, GPS coords, URL
Connection Type	Internet (unsecured), Internet (secured by VPN, etc.), Intranet, ...
Direction	Inbound, Outbound

State	Authentication State
	Authorization State
	Accounting State
	Session State

Table 2: Context Matching Capability Index

Action Capability Index	
Ingress port	SFC header termination, VxLAN header termination
Actions	Pass Deny Mirror Simple Statistics: Count (X min; Day;..) Client specified Functions: URL
Egress	Encap SFC, VxLAN, or other header

Table 3: Action Capability Index

Functional Profile Index	
Profile types Signature	Name, type, or Flexible Profile/signature URL Command for I2NSF Controller to enable/disable

Table 4: Function Profile Index

10. Manageability Considerations

Management of NSFs includes:

- Lifecycle management and resource management of NSFs
- Configuration of devices, such as address configuration, device internal attributes configuration, etc.
- Signaling
- Policy rules provisioning

Currently, I2NSF only focuses on the policy rule provisioning part.

11. Security Considerations

NSF control and monitoring demand trustworthy, robust, and fully secured access. Therefore, proper secure communication channels have to be carefully specified for carrying the controlling and monitoring information between the NSFs and their management entity or entities. This has been discussed in Section 4.

12. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

13. Acknowledgements

This document includes significant contributions from Christian Jacquenet (Orange), Seetharama Rao Durbha (Cablelabs), Mohamed Boucadair (Orange), Ramki Krishnan (Dell), Anil Lohiya (Juniper Networks), Joe Parrott (BT), Frank Xialing (Huawei), and XiaoJun Zhuang (China Mobile).

Some of the results leading to this work have received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 611458.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, July 2014, <http://www.rfc-editor.org/info/rfc7297>.

14.2. Informative References

[RFC8192]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", RFC 8192, July 2017 https://datatracker.ietf.org/doc/rfc8192/.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Sreenivasa, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-11 (work in progress), June, 2017.

[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-04 (work in progress), July 2017.

[I-D.draft-xibassnez-i2nsf-capability]
Xia, L., Strassner, J., Basile, C., and Lopez, D., "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-02.txt (work in progress), July, 2017.

[I-D.pastor-i2nsf-remote-attestation]
Pastor, A., Lopez, D., and A. Shaw, "Remote Attestation Procedures for Network Security Functions (NSFs) through the I2NSF Security Controller", draft-pastor-i2nsf-nsf-remote-attestation-01 (work in progress), March 2017.

[I-D.xie-i2nsf-demo-outline-design]
Xie, Y., Xia, L., and J. Wu, "Interface to Network Security Functions Demo Outline Design", draft-xie-i2nsf-demo-outline-design-00 (work in progress), April 2015.

[gs_NFV] "ETSI NFV Group Specification; Network Functions Virtualization (NFV) Use Cases. ETSI GS NFV 001v1.1.1", 2013.

Authors' Addresses

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

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

Edward Lopez
Curveball Networks
Chantilly, Virgina
USA

Phone: +1 703 220 0988
Email: elopez@fortinet.com

Linda Dunbar
Huawei

Email: Linda.Dunbar@huawei.com

John Strassner
Huawei

Email: John.sc.Strassner@huawei.com

Rakesh Kumar
Juniper Networks

Email: rkkumar@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

J. Jeong
D. Daghmehchi
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei

March 13, 2017

Consumer-Facing Interface YANG Data Model for Interface to Network Security Functions
draft-jeong-i2nsf-consumer-facing-interface-dm-01

Abstract

This document describes a YANG data model for security management that is based on Interface to Network Security Functions (I2NSF) by using Network Functions Virtualization (NFV). This document proposes a security management architecture based on I2NSF framework. Note that the I2NSF framework consists of I2NSF User, Security Management System (i.e., Security Controller and Developer's Management System), and the instances of Network Security Functions (NSFs) in the lowest layer of the framework. I2NSF User consists of Application Logic, Policy Updater, and Event Collector. Security Controller consists of Security Policy Manager and NSF Capability Manager. This document explains a data model to perform the missions for a security service (i.e., VoIP-VoLTE) in I2NSF security management system.

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/1id-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 September 14, 2017.

Table of Contents

1. Introduction

Basically, information model and data model are used to defining the managed objects in the network management. Despite of some overlapped details, they have different characters in the view of network management. Generally, the main purpose of information model is to model managed objects at a conceptual level, with no dependent of any specific implementations or protocols. To make a clear overall design, the information model should hide all protocol and implementation details defining relationships between managed objects. Based on this, the information models can be implemented in different ways and mapped on different protocols. They are neutral to protocols. In general, information models can be defined in an informal way, using natural languages such as English. Furthermore, it seems advisable to use object-oriented techniques to describe an information model.

Data models are defined at a lower level of abstraction and provide many details. They provide details about the implementation and protocols' 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 impressive role of the network functions virtualization (NFV) in the network management leads to a rapid advent of NFV in this industry. As practical applications, network security functions (NSFs), such as firewall, intrusion detection system (IDS) and intrusion protection system (IPS), can also be provided as virtual network functions (VNF). By virtual technology, these VNFs might be automatically provisioned and dynamically migrated based on real-time security requirements. This document presents an information model to implement security functions based on NFV.

This document proposes a data modeling in an architecture for security management [i2nsf-security-mgnt], which is based on I2NSF framework [i2nsf-framework], the requirements and information model for I2NSF consumer-facing interface (i.e., client-facing interface) to security controller [client-facing-inf-req] [client-facing-inf-im]. This I2NSF framework contains I2NSF User, Security Management System (i.e., Security Controller and Developer's Management System), and NSFs in the NSF instance layer. The security management architecture has more detailed structures for core components in the I2NSF framework. I2NSF User includes Application Logic, Policy Updater, and Event Collector. Security Controller contains Security Policy Manager and NSF Capability Manager.

Application Logic generates a high-level policy and Policy Updater

sends it to Security Policy Manager via Consumer-Facing Interface. Security Policy Manager maps the high-level policy into several low-level policies in Security Controller. After mapping, the low-level policies are distributed to NSF(s) via NSF-Facing Interface so that they can be enforced in them. When an event occurs for NSF to change a low-level policy, NSF sends the event to Security Controller via NSF-Facing Interface. Security Controller then forwards it to Event Collector via Consumer-Facing Interface. Next, Event Collector sends it to Application Logic. Application Logic then updates the current policies in accordance with the event.

This document proposes a data model for security services in the security management architecture in [i2nsf-security-mgnt] so that the security management architecture can support flexible and effective security policies.

2. Objectives

The two main objectives for security management architecture in this document are as follows:

- High-level security management: To propose the design of a generic security management architecture to support the enforcement of flexible and effective security policies in NSFs.

- Automatic update of security policies: To provide the reflection of the updated low-level security policies for new security attacks on the corresponding high-level security policies.

3. 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].

4. Terminology

This document uses the terminology described in [i2nsf-framework][i2nsf-security-mgnt]. In addition, the following terms are defined below:

- Application Logic: It is a component in the security management architecture which generates high-level security policies to block or mitigate security attacks.

- Policy Updater: It is a component which forwards a high-level security policy to Security Controller. The high-level policy is received from Application Logic.

- Security Policy Manager: It maps a high-level security policy received from Policy Updater into low-level security policies, and vice versa.

- NSF Capability Manager: It is a component which stores the NSF capability registered by Developer’s Management System via Registration Interface and shares it with Security Policy Manger to generate the corresponding low-level security policies.

- Event Collector: It is a component which receives an event from Security Controller, which should be reflected by updating (or generating) a high-level policy in Application Logic.

5. Architecture of Security Management

Generally, Data models are often represented in formal data definition languages that are specific to the management protocol being used. Based on NFV, the structure of the proposed model is based on VNFs to provide a flexible and effective security policies. Figure 1 illustrates the structure of the suggested model. The architecture is designed based on three layers: I2NSF user, security management system, and NSF instances. The high level security policies are defined and distributed in the I2NSF user layer. Translating the high level security policies relevant to NSF capability and delivering them to NSF interfaces are performed in the security management system.

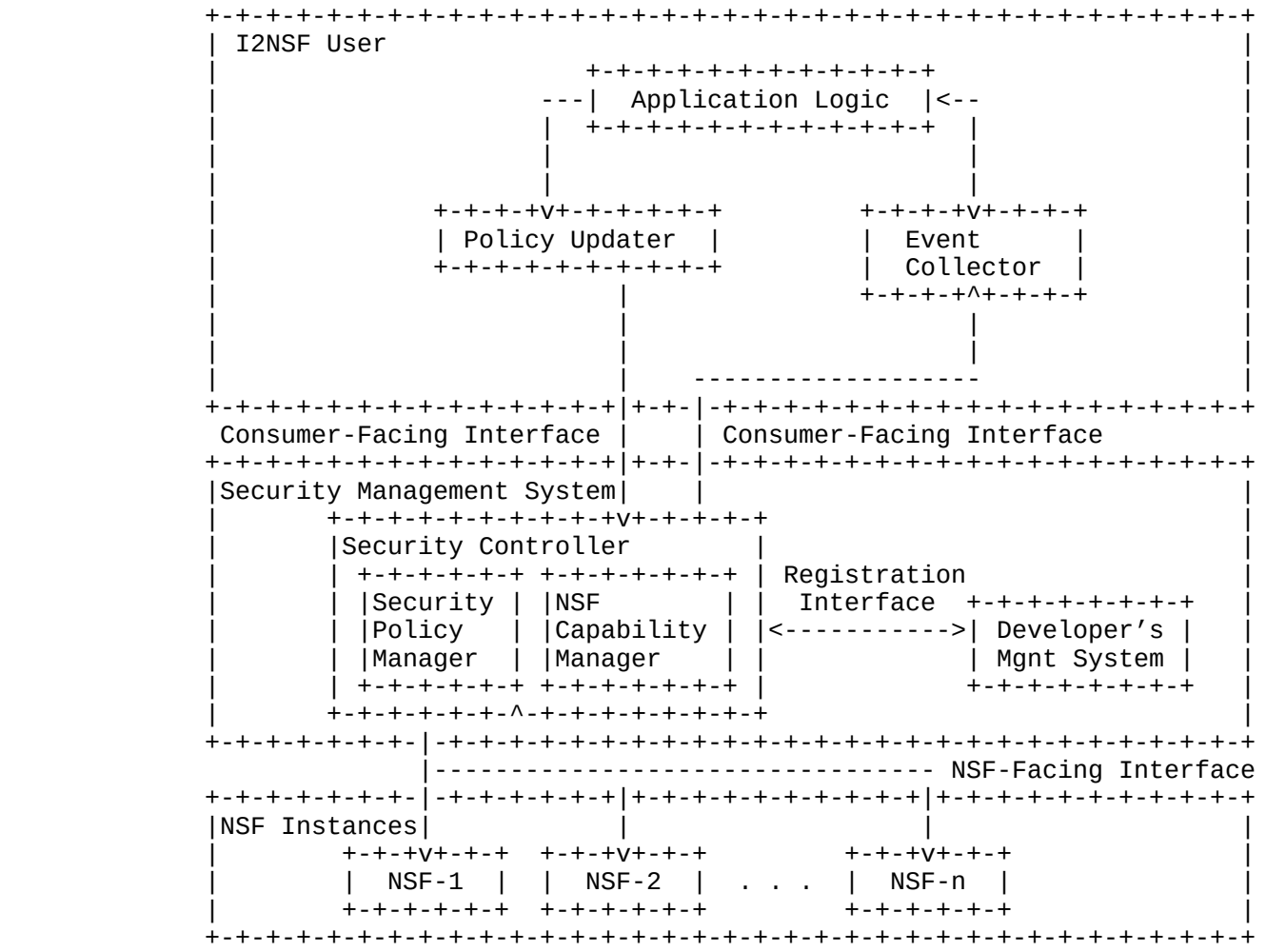


Figure 1: Security Management Architecture in I2NSF

5.1. I2NSF User

I2NSF User layer consists of three components; Application logic, Policy Updater, and Event Collector. The Application logic is a component which generates high level security policies. To this end, it receives the event for updating (or generating) a high level policy from Event collector and updates (or generates) a high level policy based on the collected events. After that, the high level policy is sent to Policy updater in order to distribute it to Security controller(s). In order to update (or generate) a high level policy, Event collector receives the events sent by Security

controller and sends them to Application logic. Based on these feedbacks, the Application logic can update (or generate) high level security policies.

5.2. Security Management System

In the security management system layer, the Security policy manager receives a high level policy from Policy updater via Consumer-Facing Interface and maps this policy into several low level policies. These low level policies are relevant to a given NSF capability that is registered into NSF capability manager. Moreover, Security policy manager delivers those policies to NSF(s) via NSF-Facing Interface.

To generate low level policies relevant to a given NSF capability, the NSF capability manager stores an NSF's capability registered by Developer's management system and shares it with Security policy manager. Whenever a new NSF is registered, NSF capability manager requests Developer's management system to register the NSF's capability into the management table of NSF capability manager via Registration Interface. Developer's management system is another part of security management system to registers a new NSF's capability into NSF capability manager.

5.3. NSF Instances

All NSFs are located at this layer. After mapping the high level policies to low level policies, the Security policy manager delivers those policies to NSF(s) through NSF-Facing Interface.

6. Use Case: VoIP-VoLTE Security Service

As a use case for implementation, VoIP-VoLTE security management is considered to develop a data model. Based on this, the VoIP-VoLTE security manager acts as Application logic for VoIP-VoLTE security services and defines the security conditions. Based on VoIP-VoLTE security management, the list of illegal devices information is stored in VoIP-VoLTE database and can be updated either manually or automatically by VoIP-VoLTE security manager. To define the policies, information of dangerous domain blacklists (e.g., IP addresses and source ports), time management (e.g., access time and expire time), user-agent (e.g., priority levels), and Session Initiation Protocol (SIP) URIs of an SIP device that are suspicious of illegal call or authentication is published by VoIP-VoLTE security manager. Accordingly, ahe list of illegal devices, which is automatically (or manually) updated, is stored in VoIP-VoLTE database. The VoIP-VoLTE security manager periodically loads this list to generate a new high level security policy (e.g., the blocking list of illegal devices using IP address, source ports, etc) to

prevent the delivery of packets from/to those newly added VoIP-VoLTE attackers.

When the NSF detects an anomalous message or call delivered from a domain, the information of the domain such as an IP address, user-agents and expire time values is sent by an NSF to Security controller via NSF-Facing Interface. Security controller delivers it to Event collector. Event collector forwards the detected domain information to VoIP-VoLTE security manager, and then VoIP-VoLTE security manager updates the VoIP-VoLTE database.

6.1. Security Management for VoIP-VoLTE Security Service

VoIP-VoLTE security management maintains and publishes the blacklists of IP addresses, source ports, expire time, user-agents, and Session Initiation Protocol (SIP) URIs of SIP device that are suspicious of illegal call and authentication. In our generic security management architecture, VoIP-VoLTE Security Manager is plays the role of Application Logic for VoIP-VoLTE security services in Figure 1.

Based on VoIP-VoLTE security management, the list of illegal devices information can be updated either manually or automatically by VoIP-VoLTE Security Manager as Application Logic. Also, VoIP-VoLTE Security Manager periodically generates a new high-level security policy to prevent the delivery of packets from/to those newly added VoIP-VoLTE attackers and enforce the low-level security policies in NSF. It sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

When the NSF detects an anomalous message or call delivered from a domain, the domain information such as an IP address, user-agents and expire time values is sent by an NSF to Security Controller via NSF Facing Interface. Security Controller delivers it to Event Collector. Event Collector forwards the detected domain information to VoIP-VoLTE Security Manager, and then VoIP-VoLTE Security Manager updates the VoIP-VoLTE database.

6.2. Data Modeling for VoIP-VoLTE Security Service

The data model for VoIP-VoLTE Security Service is derived from i2nsf consumer-facing interface information model. The main objective of this data model is to fully transform the information model into a YANG data model that can be used for delivering security policies to successfully orchestrate control or management messages between the components in the proposed security management architecture.

The sementics of the data model must align with those of client-facing interface to security controller information model. The

transformation of the information model was done by hand, and thus, certain changes were made to reflect the fact that this is a YANG data model.

This data model is a 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. Here, the volte/voip security service is used as an example case for policy rule generation.

To implement the model, three parameters have been considered to define the high level policies; blacklisting countries, time interval specification, and caller's priority levels. If the administrator sets a new high-level security policy, a data model parser in I2NSF User interprets the policy and generates an XML file in accordance with a YANG data model. In order to enable interaction between I2NSF User and Security management system, a communication channel based on RESTCONF is implemented. Basically, the data model is defined based on the security policy requirements to detect the suspicious calls in VoIP-VoLTE service. Figure 2 shows a generic data model of VoIP-VoLTE security service.

```
   +--: (ieft-i2nsf-policy)
      +--rw policy-lifecycle-list
      |  +--rw policy-lifecycle-container  *(policy-lifecycle-id)
      |   +--rw expiration-event
      |   |  +--rw enabled                  boolean
      |   |  +--rw event-id                 uint 16
      |   |  +--rw event-date               date-and-time
      |   +--rw expiration-time
      |      +--rw enabled                  boolean
      |         +--rw time                  date-and-time
      +--rw policy-rule-list
      |  +--rw policy-rule-container   *[policy-rule-id]
      |     +--rw policy-rule-id            uint 16
      |     +--rw policy-name               string
      |     +--rw policy-date               date-and-time
      |     +--rw service
      |     |  +--voip-handling             boolean
      |     |  +--volte-handling            boolean
      |     +--rw condition  *[condition-id]
      |        +--rw caller
      |        |  +--rw caller-id           uint 16
      |        |  +--rw caller-location
      |        |     +--rw country          string
      |        |     +--rw city             string
      |        +--rw callee
      |        |  +--rw callee-id           uint 16
```

```
|        |  +--rw callee-location
|        |     +--rw country          string
|        |     +--rw city             string
|        +--rw valid-time-interval
|           +--rw start-time          data-and-time
|           +--rw end-time            data-and-time
+--rw action-list
|  +--rw action-container
|     +--rw action-date                 date-and-time
|     +--rw action-name                 string
|        +--: (action-name-ingress)
|        |  +--rw permit?               boolean
|        |  +--rw mirror?               boolean
|        |  +--rw log?                  boolean
|        +--: (action-name-engress)
|        +--rw redirection?           boolean
|
+--rw update-list
   +--rw update-container                  *(update-id)
      +--rw update-event
      |  +--rw update-event-id             uint 16
      |  +--rw update-enabled              boolean
      |  +--rw update-event-date           date-and-time
      |  +--rw update-log             string
      +--rw update-time               date-and-time
```

Figure 2: Generic Data Model for VoIP-VoLTE Security Service

The data model consists of policy life cycle management, policy rule, and action. The policy life cycle field specifies an expiration time and/or a set of expiration events to determine the life-time of the policy itself. The policy rule field represents the specific information about a high-level policy such as service types, conditions and valid time interval. The action field specifies which actions should be taken. For example, call traffic from a blacklisted caller location at an unusual time of day (included in the valid-time-interval) could be blocked and sequentially forwarded to a pre-defined host for Deep Packet Inspection (DPI) when both permit and mirror are assigned true.

To translate a high level policy into a set of low level policies, the security management system is implemented. After translating the high-level security policy, Security management system generates low-level security policies to specify the actions network traffic from and/or to those IP addresses. The data model parser generates an XML _le for a low-level security policy and delivers it to proper NSF instances. Security management system also interprets security events generated by NSF into a high-level log message in a YANG data

model and delivers it to I2NSF Users in the opposite direction.

In this case, we select a firewall application as an NSF instance to determine whether a VoIP-VoLTE call is suspicious or not by checking the caller's and callee's locations and call time. When a call has suspicious behavior patterns, its network traffic could be effectively blocked by the firewall application according to the low-level security policy. The results for the firewall application would be delivered in a YANG data model to the Security management system through the RESTCONF protocol. Multiple NSF instances can be considered depending on specific situations. For example, additionally DPI can be used for analyzing the network traffic from suspicious callers.

6.3. YANG Data Model for VoIP-VoLTE Security Service

This section describes a YANG data model for VoIP-VoLTE security service, which is based on the information of consumer-facing interface to security controller [client-facing-inf-im].

<CODE BEGINS> file "ietf-i2nsf-consumer-facing-interface.yang"

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

  import ietf-yang-types {
    prefix yang;
  }

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

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

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

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

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

```
   description
     "This module defines a YANG data module for consumer-facing
      interface to security controller.";
   revision "2016-11-13"{
     description "Initial revision";
     reference
       "draft-kumar-i2nsf-client-facing-interface-im-01";
   }

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

     list policy-lifecycle {
       key "policy-lifecycle-id";
       description
         "The ID of the policy lifecycle for each policy.
         This must be unique.";

       leaf policy-lifecycle-id {
         type uint16;
         mandatory true;
         description
           "This is policy lifecycle-id.";
       }

       container expiration-event {
         description
           "The event which makes the policy expired.";

         leaf enabled {
           type boolean;
           mandatory true;
           description
             "This represents whether the policy is enabled or
             disabled.";
         }

         leaf event-id {
           type uint16;
           mandatory true;
           description
             "The ID of the event. This must be unique.";
```

```
        }
        leaf event-date {
          type yang:date-and-time;
          mandatory true;
          description
            "The date when the event is triggered.";
        }
      }

      container expiration-time {
        description
          "The time when the policy is expired.";

        leaf enabled {
          type boolean;
          mandatory true;
          description
            "This represents whether the policy is enabled or
            disabled.";
        }

        leaf time {
          type yang:date-and-time;
          mandatory true;
          description
            "The time when the policy is enabled.";
        }
      }
    }

    list policy-rule {
      key "policy-rule-id";
      description
        "The ID of the policy rule.
       This is key for policy-rule-list.
       This must be unique.";

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

      leaf policy-rule-id {
        type uint16;
        mandatory true;
```

```
        description
          "The ID of the policy rule. This must be unique.";
      }

      leaf policy-rule-date {
          type yang:date-and-time;
          mandatory true;
          description
            "The date when the date-and-time when the policy is
             created.";
        }

      container service {
        description
          "The services which NSFs could perform to manage the
          security attacks.
          This consists of voip-handling and volte-handling.
          This will be extended in later version.";

        leaf voip-handling {
          type boolean;
          mandatory true;
          description
            "This field represents whether the policy contains
            handling the voip packet flow.";
        }

        leaf volte-handling {
          type boolean;
          mandatory true;
          description
            "This field represents whether the policy contains
            handling the volte packet flow.";
        }
      }

      list condition {
        key "condition-id";
        description
          "The ID of the condition. This must be unique.";

        leaf condition-id {
        type uint16;
        mandatory true;
        description
          "This is condition id";
      }
```

```
        container caller {
          description
            "The caller of VoIP-VoLTE call";

          leaf caller-id {
            type uint16;
            mandatory true;
            description
              "The ID of the caller. This must be unique.";
          }

          container caller-location {
            description
              "The location of the caller.";

            leaf country {
              type string;
              mandatory true;
              description
                "The country of the caller.";
            }

            leaf city {
              type string;
              mandatory true;
              description
                "The city of the caller.";
            }
          }
        }

        container callee {
          description
            "The callee of VoIP-VoLTE call.";

          leaf callee-id {
            type uint16;
            mandatory true;
            description
              "The ID of the callee. This must be unique.";
          }

          container callee-location {
            description
              "The location of the callee.";

            leaf country {
              type string;
```

```
            mandatory true;
            description
              "The country of the callee.";
          }

          leaf city {
            type string;
            mandatory true;
            description
              "The city of the callee.";
          }
        }
      }

      container valid-time-interval {
        description
          "The time when the policy starts or ends to be valid.";

        leaf start-time {
          type yang:date-and-time;
          mandatory true;
          description
            "The time when the policy starts to be valid.";
        }

        leaf end-time {
          type yang:date-and-time;
          mandatory true;
          description
            "The time when the policy ends to be valid.";
        }
      }
    }
  }

  container action {
    description
      "TBD";

    choice action-type {
      description
        "The flow-based NSFs realize the network security
         functions by executing various Actions, which at least
         includes ingress-action, egress-action, and
         advanced-action.";

      case ingress-action {
        description
```

```
            "The ingress actions consist of permit, mirror and log.";

          leaf permit {
            type boolean;
            mandatory true;
            description
              "Packet flow is permitted or denyed.";
          }

          leaf mirror {
            type boolean;
            mandatory true;
            description
              "Packet flow is mirroried.";
          }

          leaf log {
            type boolean;
            mandatory true;
            description
              "Packet flow is logged.";
          }
        }

        case engress-type {
          description
            "The egress action consists of redirection. TBD";

          leaf redirection {
            type boolean;
            mandatory true;
            description
              "Packet flow is redireted.";
          }
        }

    container update {
        description
          "The event which makes the policy expired.";

        leaf update-id {
          type uint16;
            mandatory true;
            description
              "The policy update-id to distingush each update.
               This must be unique.";
          }
```

```
          leaf update-event-id {
            type uint16;
            mandatory true;
            description
              "The ID of the event. This must be unique.";
            }

            leaf update-enabled {
                type boolean;
                mandatory true;
                description
                  "The update is enabled or disabled.";
              }
            leaf update-event-date {
              type yang:date-and-time;
              mandatory true;
              description
                "The date when the update-event is triggered.";
            }
            leaf update-log {
              type boolean;
              mandatory true;
              description
                "To log update and its description";
            }
          }
        }
      }
    }
  }
```

<CODE ENDS>

Figure 3: YANG Data Model for VoIP-VoLTE Security Service

7. Security Considerations

The security management architecture is derived from the I2NSF framework [i2nsf-framework], so the security considerations of the I2NSF framework should be included in this document. Especially, proper secure communication channels should be used for the delivery of control or management messages amongst the components in the proposed architecture.

8. Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP)

(No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This document has greatly benefited from inputs by Hyoungshick Kim, Hoon Ko, Sanghak Oh, Eunsoo Kim, Jinyong Tim Kim, Daeyoung Hyun, and Se-Hui Lee.

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

[i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-04 (work in progress), October 2016.

[i2nsf-security-mgnt] Kim, H., Ko, H., Oh, S., Jeong, J., and S. Lee, "An Architecture for Security Management in I2NSF Framework", draft-kim-i2nsf-security-management-architecture-03 (work in progress), October 2016.

[client-facing-inf-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-00 (work in progress), October 2016.

[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-01 (work in progress), October 2016.

Appendix A. Changes from draft-jeong-i2nsf-consumer-facing-interface-dm-00

The following changes are made from draft-jeong-i2nsf-consumer-facing-interface-dm-00:

- Miscellaneous expressions in the whole descriptions are corrected.

- A YANG data model is clarified to efficiently support VoIP-VoLTE security services.

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

Mahdi Daghmehchi Firoozjaei
Department of Electical and Computer Enginering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82-31-299-4104
EMail: mdaghmechi@skku.edu

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

Phone:
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: September 13, 2017

J. Kim
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
L. Xia
Huawei

March 12, 2017

I2NSF Network Security Functions Facing Interface YANG Data Model
draft-kim-i2nsf-nsf-facing-interface-data-model-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 three security capabilities (i.e., network security functions), such as network security control, content security control, and attack mitigation control, as defined in the information model for the I2NSF NSF capabilities.

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/1id-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 September 13, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for 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 three security capabilities (i.e., network security functions), such as network security control, content security control, and attack mitigation control, as defined in [i2nsf-cap-interface-im]. With these data model, I2NSF controller can control the capabilities of NSFs.

2. Requirements Language

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

3. Terminology

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

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

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

3.1. Tree Diagrams

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

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

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

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

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

```
module : ietf-i2nsf-nsf-facing-interface
+--rw cfg-network-security-control
|  +--rw policy
|    +--rw policy-name string
|    +--rw policy-id string
|    +--rw rules* [rule-id]
|      +--rw rule-name string
|      +--rw rule-id uint 8
|      +--rw rule-msg string
|      +--rw rule-rev uint 8
|      +--rw rule-gid uint 8
|      +--rw rule-class-type string
|      +--rw rule-reference string
|      +--rw rule-priority uint 8
|      +--rw event
|      |  +--rw user-security-event* [usr-sec-event-id]
|      |  |  +--rw usr-sec-event-id uint 8
|      |  |  +--rw usr-sec-event-content string
|      |  |  +--rw usr-sec-event-format uint 8
|      |  |  +--rw usr-sec-event-type uint 8
|      |  +--rw device-security-event* [dev-sec-event-id]
|      |  |  +--rw dev-sec-event-id uint 8
|      |  |  +--rw dev-sec-event-content string
|      |  |  +--rw dev-sec-event-format uint 8
|      |  |  +--rw dev-sec-event-type uint 8
|      |  |  +--rw dev-sec-event-type-severity uint 8
|      |  +--rw system-security-event* [sys-sec-event-id]
|      |  |  +--rw sys-sec-event-id uint 8
|      |  |  +--rw sys-sec-event-content string
|      |  |  +--rw sys-sec-event-format uint 8
|      |  |  +--rw sys-sec-event-type uint 8
|      |  +--rw time-security-event* [time-sec-event-id]
|      |  |  +--rw time-sec-event-id uint 8
|      |  |  +--rw time-sec-event-period-begin yang:date-and-time
|      |  |  +--rw time-sec-event-period-end yang:date-and-time
|      |  |  +--rw time-sec-evnet-time-zone string
```

```
|      +--rw condition
|      |  +--rw packet-security-condition* [pkt-security-id]
|      |  |  +--rw pkt-security-id uint 8
|      |  |  +--rw packet-security-mac-condition
|      |  |  |  +--rw pkt-sec-cond-mac-dest* inet:port-number
|      |  |  |  +--rw pkt-sec-cond-mac-src* inet:port-number
|      |  |  |  +--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* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv4-tos* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv4-total-length* uint 16
|      |  |  |  +--rw pkt-sec-cond-ipv4-id* uint 16
|      |  |  |  +--rw pkt-sec-cond-ipv4-fragment* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv4-fragment-offset* uint 16
|      |  |  |  +--rw pkt-sec-cond-ipv4-ttl* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv4-protocol* uint 8
|      |  |  |  +--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* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv6-flow-label* uint 32
|      |  |  |  +--rw pkt-sec-cond-ipv6-payload-length* uint 16
|      |  |  |  +--rw pkt-sec-cond-ipv6-next-header* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv6-hop-limit* uint 8
|      |  |  |  +--rw pkt-sec-cond-ipv6-src* inet:ipv6-address
|      |  |  |  +--rw pkt-sec-cond-ipv6-dest* inet:ipv6-address
|      |  |  +--rw packet-security-tcp-condition
|      |  |  |  +--rw pkt-sec-cond-tcp-seq-num* uint 32
|      |  |  |  +--rw pkt-sec-cond-tcp-ack-num* uint 32
|      |  |  |  +--rw pkt-sec-cond-tcp-window-size* uint 16
|      |  |  |  +--rw pkt-sec-cond-tcp-falgs* uint 8
|      |  |  +--rw packet-security-udp-condition
|      |  |  |  +--rw pkt-sec-cond-udp-length* string
|      |  |  +--rw packet-security-icmp-condition
|      |  |     +--rw pkt-sec-cond-icmp-type* uint 8
|      |  |     +--rw pkt-sec-cond-icmp-code* uint 8
|      |  |     +--rw pkt-sec-cond-icmp-seq-num* uint 32
|      |  +--rw packet-payload-security-condition* [pkt-payload-id]
|      |  |  +--rw pkt-payload-id uint 8
|      |  |  +--rw pkt-payload-content string
|      |  |  +--rw pkt-payload-nocase boolean
```

```
|       |  |  +--rw pkt-payload-depth uint 32
|       |  |  +--rw pkt-payload-offset uint 32
|       |  |  +--rw pkt-payload-distance uint 32
|       |  |  +--rw pkt-payload-within uint 32
|       |  |  +--rw pkt-payload-isdataat uint 32
|       |  |  +--rw pkt-payload-dsize uint 32
|       |  |  +--rw pkt-payload-replace string
|       |  |  +--rw pkt-payload-pcre string
|       |  |  +--rw pkt-payload-rpc
|       |  |     +--rw pkt-payload-rpc-app-num uint 32
|       |  |     +--rw pkt-payload-rpc-version-num uint 32
|       |  |     +--rw pkt-payload-rpc-procedure-num uint 32
|       |  +--rw target-security-condition* [target-sec-cond-id]
|       |  |  +--rw target-sec-cond-id uint 8
|       |  |  +--rw service-sec-context-cond?
|       |  |  |  +--rw name string
|       |  |  |  +--rw protocol
|       |  |  |  |  +--rw TCP? boolean
|       |  |  |  |  +--rw UDP? boolean
|       |  |  |  |  +--rw ICMP? boolean
|       |  |  |  |  +--rw ICMPv6? boolean
|       |  |  |  |  +--rw IP? boolean
|       |  |  |  +--rw src-port? inet:port-number
|       |  |  |  +--rw dest-port? inet:port-number
|       |  |  +--rw application-sec-context-cond?
|       |  |  |  +--rw name string
|       |  |  |  +--rw category
|       |  |  |  |  +--rw business-system?  boolean
|       |  |  |  |  +--rw entertainment?  boolean
|       |  |  |  |  +--rw internet?  boolean
|       |  |  |  |  +--rw network?  boolean
|       |  |  |  |  +--rw general?  boolean
|       |  |  |  +--rw subcategory
|       |  |  |  |  +--rw finance?  boolean
|       |  |  |  |  +--rw email?  boolean
|       |  |  |  |  +--rw game?  boolean
|       |  |  |  |  +--rw media-sharing?  boolean
|       |  |  |  |  +--rw social-network?  boolean
|       |  |  |  |  +--rw web-posting?  boolean
|       |  |  |  +--rw data-transmission-model
|       |  |  |  |  +--rw client-server?  boolean
|       |  |  |  |  +--rw browser-based?  boolean
|       |  |  |  |  +--rw networking?  boolean
|       |  |  |  |  +--rw peer-to-peer?  boolean
|       |  |  |  |  +--rw unassigned?  boolean
|       |  |  |  +--rw risk-level
|       |  |  |     +--rw exploitable?  boolean
|       |  |  |     +--rw productivity-loss?  boolean
```

```
|      |  |  |     +--rw evasive?  boolean
|      |  |  |     +--rw data-loss?  boolean
|      |  |  |     +--rw malware-vehicle?  boolean
|      |  |  |     +--rw bandwidth-consuming?  boolean
|      |  |  |     +--rw tunneling?  boolean
|      |  |  +--rw device-sec-context-cond?
|      |  |     +--rw pc?  boolean
|      |  |     +--rw mobile-phone?  boolean
|      |  |     +--rw tablet?  boolean
|      |  |     +--rw voip-phone  boolean
|      |  +--rw user-security-cond* [usr-sec-cond-id]
|      |  |  +--rw usr-sec-cond-id uint 8
|      |  |     +--rw user
|      |  |     |  +--rw (user-name)?
|      |  |     |    +--: (tenant)
|      |  |     |    |  +--rw tenant uint 8
|      |  |     |    +--: (vn-id)
|      |  |     |       +--rw vn-id uint 8
|      |  |     +--rw group
|      |  |        +--rw (group-name)?
|      |  |          +--: (tenant)
|      |  |          |  +--rw tenant uint 8
|      |  |          +--: (vn-id)
|      |  |             +--rw vn-id uint 8
|      |  +--rw security-context-condition* [sec-context-cond-id]
|      |  |  +--rw sec-context-cond-id uint 8
|      |  |  +--rw (state)?
|      |  |  |  +--: (session-state)
|      |  |  |  |  +--rw tcp-session-state
|      |  |  |  |     +--rw new?  boolean
|      |  |  |  |     +--rw established?  boolean
|      |  |  |  |     +--rw related?  boolean
|      |  |  |  |     +--rw invalid?  boolean
|      |  |  |  |     +--rw untracked?  boolean
|      |  |  |  +--: (session-aaa-state)
|      |  |  |  |  +--rw session-sip-state
|      |  |  |  |     +--rw auth-state?  boolean
|      |  |  |  |     +--rw call-state?  boolean
|      |  |  |  +--: (access-mode)
|      |  |  |  |  +--rw access-mode  string
|      |  +--rw generic-context-condition* [gen-context-cond-id]
|      |     +--rw gen-context-cond-id uint 8
|      |     +--rw geographic-location
|      |        +--rw geographic-location-id* uint 8
|      +--rw action
|         +--rw (action-type)?
|            +--: (ingress-action)
|            |  +--rw (ingress-action-type)?
```

```
|            |    +--: (pass)
|            |    |  +--rw pass boolean
|            |    +--: (drop)
|            |    |  +--rw drop boolan
|            |    +--: (reject)
|            |    |  +--rw reject boolean
|            |    +--: (mirror)
|            |       +--rw mirror boolean
|            +--: (egress-action)
|            |  +--rw (egress-action-type)?
|            |    +--: (invoke-signaling)
|            |    |  +--rw invoke-signaling boolean
|            |    +--: (tunnel-encapsulation)
|            |    |  +--rw tunnel-encapsulation boolean
|            |    +--: (forwarding)
|            |       +--rw forwarding  boolean
|            +--: (apply-profile-action)
|               +--rw (apply-profile-action-type)?
|                  +--: (content-security-control)
|                  |  +--rw content-security-control-types
|                  |     +--rw antivirus
|                  |     | +--rw antivirus-insp?  boolean
|                  |     +--rw ips
|                  |     | +--rw ips-insp?  boolean
|                  |     +--rw ids
|                  |     |  +--rw ids-insp?  boolean
|                  |     +--rw url-filtering
|                  |     |  +--rw url-filtering-insp?  boolean
|                  |     +--rw data-filtering
|                  |     |  +--rw data-filtering-insp?  boolean
|                  |     +--rw mail-filtering
|                  |     |  +--rw mail-filtering-insp?  boolean
|                  |     +--rw file-blocking
|                  |     |  +--rw file-blocking-insp?  boolean
|                  |     +--rw file-isolate
|                  |     |  +--rw file-isolate-insp?  boolean
|                  |     +--rw pkt-capture
|                  |     |  +--rw pkt-capture-insp?  boolean
|                  |     +--rw application-control
|                  |     |  +--rw application-control-insp?  boolean
|                  |     +--rw voip-volte
|                  |        +--rw voip-volte-insp? boolean
|                  +--: (attack-mitigation-control)
|                     +--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
|                        |     |    |  +--rw syn-flood-insp    boolean
|                        |     |    +--rw udp-flood-attack
|                        |     |    |  +--rw udp-flood-insp    boolean
|                        |     |    +--rw icmp-flood-attack
|                        |     |    |  +--rw icmp-flood-insp    boolean
|                        |     |    +--rw ip-frag-flood-attack
|                        |     |    |  +--rw ip-frag-flood-insp boolean
|                        |     |    +--rw ipv6-related-attacks
|                        |     |       +--rw ipv6-related-insp    boolean
|                        |     +--: (app-layer-ddos-attack)
|                        |        +--rw app-layer-ddos-attack-types
|                        |           +--rw http-flood-attack
|                        |           |  +--rw http-flood-insp    boolean
|                        |           +--rw https-flood-attack
|                        |           |  +--rw https-flood-insp    boolean
|                        |           +--rw dns-flood-attack
|                        |           |  +--rw dns-flood-insp    boolean
|                        |           +--rw dns-amp-flood-attack
|                        |           |  +--rw dns-amp-flood-insp boolean
|                        |           +--rw ssl-ddos-attack
|                        |              +--rw ssl-ddos-insp    boolean
|                        +--: (single-packet-attack)
|                           +--rw (single-packet-attack-type)?
|                              +--: (scan-and-sniff-attack)
|                              |  +--rw scan-and-sniff-attack-types
|                              |     +--rw ip-sweep-attack
|                              |     |  +--rw ip-sweep-insp    boolean
|                              |     +--rw port-scanning-attack
|                              |        +--rw port-scanning-insp boolean
|                              +--: (malformed-packet-attack)
|                              |  +--rw malformed-packet-attack-types
|                              |     +--rw ping-of-death-attack
|                              |     |  +--rw ping-of-death-insp boolean
|                              |     +--rw teardrop-attack
|                              |        +--rw teardrop-insp    boolean
|                              +--: (special-packet-attack)
|                                 +--rw special-packet-attack-types
|                                    +--rw oversized-icmp-attack
|                                    |  +--rw oversized-icmp-insp boolean
|                                    +--rw tracert-attack
|                                       +--rw tracert-insp    boolean
+--rw cfg-content-security-control
| +--rw (cfg-content-security-control-type)?
|    +--: (cfg-antivirus)
|    |  +--rw antivirus-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-ips)
```

```
|    |  +--rw ips-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-ids)
|    |  +--rw ids-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-url-filter)
|    |  +--rw url-filter-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-data-filter)
|    |  +--rw data-filter-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-mail-filter)
|    |  +--rw mail-filter-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-file-blocking)
|    |  +--rw file-blocking-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-file-isolate)
|    |  +--rw file-isolate-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-pkt-capture)
|    |  +--rw pkt-capture-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-app-control)
|    |  +--rw app-control-rule* [rule-id]
|    |     +--rw rule-id  uint8
|    +--: (cfg-voip-volte)
|       +--rw voip-volte-rule*  [rule-id]
|          +--rw rule-id  uint 8
|          +--rw event
|          |  +--rw called-voip  boolean
|          |  +--rw called-volte  boolean
|          +--rw condition
|          |  +--rw sip-header*  [sip-header-uri]
|          |  |  +--rw sip-header-uri string
|          |  |  +--rw sip-header-method string
|          |  |  +--rw expire-time yang:date-and-time
|          |  |  +--rw sip-header-user-agent uint32
|          |  +--rw cell-region?* [cell-id-region]
|          |     +--rw cell-id-region uint 32
|          +--rw action
|             +--rw (action-type)?
|                +--: (ingress-action)
|                |  +--rw (ingress-action-type)?
|                |    +--: (pass)
|                |    |  +--rw pass boolean
|                |    +--: (drop)
|                |    |  +--rw drop boolean
```

```
|                |    +--: (reject)
|                |    |  +--rw reject boolean
|                |    +--: (alert)
|                |    |  +--rw alert boolean
|                |    +--: (mirror)
|                |       +--rw mirror boolean
|                +--: (egress-action)
|                   +--: (egress-action-type)?
|                     +--: (redirection)
|                        +--rw redirection? boolean
+--rw cfg-attack-mitigation-control
  +--rw (cfg-attack-mitigation-control-type)?
     +--: (cfg-ddos-attack)
     |  +--rw (cfg-ddos-attack-type)?
     |     +--: (cfg-network-layer-ddos-attack)
     |     |  +--rw (cfg-network-layer-ddos-attack-type)?
     |     |    +--: (cfg-syn-flood-attack)
     |     |    |  +--rw syn-flood-attack-rule* [rule-id]
     |     |    |     +--rw rule-id  uint8
     |     |    +--: (cfg-udp-flood-attack)
     |     |    |  +--rw udp-flood-attack-rule* [rule-id]
     |     |    |     +--rw rule-id  uint8
     |     |    +--: (cfg-icmp-flood-attack)
     |     |    |  +--rw icmp-flood-attack-rule* [rule-id]
     |     |    |     +--rw rule-id  uint8
     |     |    +--: (cfg-ip-frag-flood-attack)
     |     |    |  +--rw ip-frag-flood-attack-rule* [rule-id]
     |     |    |     +--rw rule-id  uint8
     |     |    +--: (cfg-ipv6-related-attacks)
     |     |       +--rw ipv6-related-attacks-rule* [rule-id]
     |     |          +--rw rule-id  uint8
     |     +--: (cfg-app-layer-ddos-attack)
     |        +--rw (cfg-app-layer-ddos-attack-type)?
     |          +--: (cfg-http-flood-attack)
     |          |  +--rw http-flood-attack-rule* [rule-id]
     |          |     +--rw rule-id  uint8
     |          +--: (cfg-https-flood-attack)
     |          |  +--rw https-flood-attack-rule* [rule-id]
     |          |     +--rw rule-id  uint8
     |          +--: (cfg-dns-flood-attack)
     |          |  +--rw dns-flood-attack-rule* [rule-id]
     |          |     +--rw rule-id  uint8
     |          +--: (cfg-dns-amp-flood-attack)
     |          |  +--rw dns-amp-flood-attack-rule* [rule-id]
     |          |     +--rw rule-id  uint8
     |          +--: (cfg-ssl-ddos-attack)
     |             +--rw ssl-ddos-attack-rule* [rule-id]
     |                +--rw rule-id  uint8
```

```
     +--: (cfg-single-packet-attack)
        +--rw (cfg-single-packet-attack-type)?
           +--: (cfg-scan-and-sniff-attack)
           |  +--rw (cfg-scan-and-sniff-attack-type)?
           |     +--: (cfg-ip-sweep-attack)
           |     |  +--rw ip-sweep-attack-rule* [rule-id]
           |     |     +--rw rule-id  uint8
           |     +--: (cfg-port-scanning-attack)
           |        +--rw prot-scanning-attack-rule* [rule-id]
           |           +--rw rule-id  uint8
           +--: (cfg-malformed-packet-attack)
           |  +--rw (cfg-malformed-packet-attack-type)?
           |     +--: (cfg-ping-of-death-attack)
           |     |  +--rw ping-of-death-attack-rule* [rule-id]
           |     |     +--rw rule-id  uint8
           |     +--: (cfg-teardrop-attack)
           |        +--rw teardrop-attack-rule* [rule-id]
           |           +--rw rule-id  uint8
           +--: (cfg-special-packet-attack)
              +--rw (cfg-special-packet-attack-type)?
                 +--: (cfg-oversized-icmp-attack)
                 |  +--rw oversized-icmp-attack-rule* [rule-id]
                 |     +--rw rule-id  uint8
                 +--: (cfg-tracert-attack)
                    +--rw tracert-attack-rule* [rule-id]
                       +--rw rule-id  uint8
```

Figure 1: Information Model of I2NSF NSF Facing Interface

5. YANG Model

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

<CODE BEGINS> file "ietf-i2nsf-nsf-facing-interface@2017-03-12.yang"

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

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

```
    prefix yang;
  }

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

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

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

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

     Editor: Jingyong Tim Kim
     <mailto:wlsdyd0930@nate.com>

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

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

  description
    "This module defines a YANG data module for network security
     functions.";
  revision "2017-03-12"{
    description "Initial revision";
    reference
      "draft-xibassnez-i2nsf-capability-00
       draft-kim-i2nsf-nsf-facing-interface-data-model-01";
  }

  //Groupings

 grouping cfg-network-security-conrol {
  description
    "Configuration for Network Security Control.";

  container policy {
    description
      "policy is a grouping
       including a set of security rules according to certain logic,
       i.e., their similarity or mutual relations, etc. The network
       security policy is able to apply over both the unidirectional
```

```
       and bidirectional traffic across the NSF.";

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

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

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

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

          leaf rule-msg {
            type string;
            mandatory true;
            description
              "The keyword msg gives more information about
               the signature and the possible alert.";
          }

          leaf rule-rev {
```

```
            type uint8;
            mandatory true;
            description
              "The sid keyword is almost every time
               accompanied by reb.";
          }

          leaf rule-gid {
            type uint8;
            mandatory true;
            description
              "The gid keyword can be used to give different
               groups of signatures another id value
               (like in sid)..";
          }

          leaf rule-class-type {
            type string;
            mandatory true;
            description
              "The classtype keyword gives information about
               the classification of rules and alerts.";
          }

          leaf rule-reference {
            type string;
            mandatory true;
            description
              "The reference keywords direct to places where
               information about the signature and about
               the problem the signature tries to address,
               can be found.";
          }

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

          container event {
            description
              " An Event is defined as any important occurrence in time
                of a change in the system being managed, and/or in the
                environment of the system being managed. When used in
                the context of policy rules for a flow-based NSF, it is
```

```
        used to determine whether the Condition clause of the
        Policy Rule can be evaluated or not. Examples of an
        I2NSF Event include time and user actions (e.g., logon,
        logoff, and actions that violate any ACL.).";
      list user-security-event {
        key usr-sec-event-id;
        description
          "The purpose of this class is to represent Events that
           are initiated by a user, such as logon and logoff
           Events. Information in this Event may be used as part
           of a test to determine if the Condition clause in
           this ECA Policy Rule should be evaluated or not.
           Examples include user identification data and the
           type of connection used by the user.";

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

        leaf usr-sec-event-content {
          type string;
          mandatory true;
          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).";
        }

        leaf usr-sec-event-format {
          type uint8;
          mandatory true;
          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 usr-sec-event-type {
                  type uint8;
                  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 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).";
                }
              }

              list device-security-event {
                key dev-sec-event-id;
                description
                 "The purpose of a DeviceSecurityEvent is to represent
                  Events that provide information from the Device that
                  are important to I2NSF Security. Information in this
                  Event may be used as part of a test to determine if
                  the Condition clause in this ECA Policy Rule should be
                  evaluated or not. Examples include alarms and various
                  device statistics (e.g., a type of threshold that was
                  exceeded), which may signal the need for further
                  action.";

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

                leaf dev-sec-event-content {
                  type string;
                  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).";
                }

                 leaf dev-sec-event-format {
                   type uint8;
                   mandatory true;
                   description
                    "This is a mandatory uint 8 enumerated integer, which
                     is used to specify the data type of the
                     devSecEventContent attribute.";
                 }

                 leaf dev-sec-event-type {
                   type uint8;
                   mandatory true;
                   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 dev-sec-event-type-severity {
                   type uint8;
                   mandatory true;
                   description
                    "This is a mandatory uint 8 enumerated integer, which
                     is used to specify the perceived severity of the
                     Event generated by this Device.";
                 }
               }

               list system-security-event {
                 key sys-sec-event-id;
                 description
                   "The purpose of a SystemSecurityEvent is to represent
                    Events that are detected by the management system,
                    instead of Events that are generated by a user or a
                    device. Information in this Event may be used as part
                    of a test to determine if the Condition clause in
                    this ECA Policy Rule should be evaluated or not.
                    Examples include an event issued by an analytics
```

```
                   system that warns against a particular pattern of
                   unknown user accesses, or an Event issued by a
                   management system that represents a set of correlated
                   and/or filtered Events.";

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

                leaf sys-sec-event-content {
                  type string;
                  mandatory true;
                  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).";
                }

                leaf sys-sec-event-format {
                  type uint8;
                  mandatory true;
                  description
                   "This is a mandatory uint 8 enumerated integer, which
                    is used to specify the data type of the
                    sysSecEventContent attribute.";
                }

                leaf sys-sec-event-type {
                  type uint8;
                  mandatory true;
                  description
                   "This is a mandatory uint 8 enumerated integer, which
                    is used to specify the type of Event that involves
                    this device.";
                }
              }
```

```
            list time-security-event {
              key time-sec-event-id;
              description
              "Purpose of a TimeSecurityEvent is to represent Events
               that are temporal in nature (e.g., the start or end of
               a period of time). Time events signify an individual
               occurrence, or a time period, in which a significant
               event happened. Information in the Event may be used as
               part of a test to determine if the Condition clause in
               this ECA Rule should be evaluated or not. Examples
               include issuing an Event at a specific time to indicate
               that a particular resource should not be accessed, or
               that different authentication and authorization
               mechanisms should now be used (e.g., because it is now
               past regular business hours).";

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

              leaf time-sec-event-period-begin {
                type yang:date-and-time;
                mandatory true;
                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-period-end {
                type yang:date-and-time;
                mandatory true;
                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 string;
              mandatory true;
              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
            "TBD";
          list packet-security-condition {
            key pkt-security-id;
            description
              "The purpose of this Class is to represent 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. This class is
               abstract, and serves as the superclass of more detailed
               conditions that involve different types of packet
               formats.";

            leaf pkt-security-id {
              type uint8;
              mandatory true;
              description
                "The ID of the packet-security-condition.";
            }

            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 inet:port-number;
                description
                  "The MAC destination address (6 octets long).";
              }

              leaf-list pkt-sec-cond-mac-src {
                type inet:port-number;
                description
```

```
            "The MAC source address (6 octets long).";
          }

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

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

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

        container packet-security-ipv4-condition {
          description
            "The purpose of this Class is to represent 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-list pkt-sec-cond-ipv4-header-length {
            type uint8;
            description
              "The IPv4 packet header consists of 14 fields,
               of which 13 are required.";
          }

          leaf-list pkt-sec-cond-ipv4-tos {
            type uint8;
            description
              "The ToS field could specify a datagram’s priority
```

```
                    and request a route for low-delay, high-throughput,
                    or highly-reliable service..";
                }

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

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

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

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

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

                leaf-list pkt-sec-cond-ipv4-protocol {
```

```
                type uint8;
                description
                  "Internet Protocol version 4(IPv4) is the fourth
                   version of the Internet Protocol (IP).";
              }

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

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

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

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

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

            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
                   now werves as a hint to routers and switches
                   with multiple outbound paths that these
                   packets should stay on the same path so that
                   they will not be reordered.";
              }

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

              leaf-list pkt-sec-cond-ipv6-next-header {
                type uint8;
```

```
        description
          "Specifies the type of the next header.
           This field usually specifies the transport
           layer protocol used by a packet's payload.";
      }

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

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

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

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

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

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

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

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

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

            leaf-list pkt-sec-cond-udp-length {
              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.";
                }
              }
            }
              list packet-payload-security-condition {
                key "pkt-payload-id";
                description
                  "The ID of the pkt-payload.
                   This is key for pkt-payload-list.
                   This must be unique.";
                leaf pkt-payload-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the packet payload.
                     This must be unique.";
                }

                leaf pkt-payload-content {
                  type string;
                  mandatory true;
                  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 pkt-payload-nocase {
                  type boolean;
                  mandatory true;
                  description
                    "If you do not want to make a distinction
                     between uppercase and lowercase characters,
                     you can use nocase.";
                }

                leaf pkt-payload-depth {
                  type uint32;
                  mandatory true;
                  description
                    "The depth keyword is a absolute content
                     modifier.";
```

```
                }

                leaf pkt-payload-offset {
                  type uint32;
                  mandatory true;
                  description
                    "The offset keyword designates from which byte
                     in the payload will be checked to fined to find
                     a match.";
                }

                leaf pkt-payload-distance {
                  type uint32;
                  mandatory true;
                  description
                    "The keyword distance is a relative content
                     modifier. This means it indicates a relation
                     between this content keyword and the content
                     preceding it.";
                }

                leaf pkt-payload-within {
                  type uint32;
                  mandatory true;
                  description
                    "The keyword within is relative to the preceding
                     match. The keyword within comes with a mandatory
                     numeric value.";
                }

                leaf pkt-payload-isdataat {
                  type uint32;
                  mandatory true;
                  description
                    "The purpose of the isdataat keyword is to
                     look if there is still data at a specific part
                     of the payload.";
                }

                leaf pkt-payload-dsize {
                  type uint32;
                  mandatory true;
                  description
                    "With the dsize keyword, you can match on the
                     size of the packet payload.";
                }

                leaf pkt-payload-replace {
```

```
                type string;
                mandatory true;
                description
                  "The replace content modifier can only be used
                   in ips. It adjusts network traffic.";
              }

              leaf pkt-payload-pcre {
                type string;
                mandatory true;
                description
                  "For information about pcre check the pcre
                   (Perl Compatible Regular Expressions)page.";
              }

              container pkt-payload-rpc{
                description
                  "The rpc keyword can be used to match in the
                   SUNRPC CALL on the RPC procedure numbers and
                   the RPC version.";
                leaf pkt-payload-rpc-app-num {
                  type uint32;
                  mandatory true;
                  description
                    "<application number>.";
                }

                leaf pkt-payload-rpc-version-num {
                  type uint32;
                  mandatory true;
                  description
                    "<version number>|*.";
                }

                leaf pkt-payload-rpc-procedure-num {
                  type uint32;
                  mandatory true;
                  description
                    "<procedure number>|*.";
                }
              }
            }
            list target-security-condition {
              key "target-sec-cond-id";
              description
                "Under the circumstances of network, it mainly
                 refers to the service, application, and device.";
              leaf target-sec-cond-id {
```

```
                type uint8;
                mandatory true;
                description
                  "The ID of the target.
                   This must be unique.";
              }
              container service-sec-context-cond{
                description
                  "A service is an application identified by a
                   protocol type and port number, such as TCP,
                   UDP, ICMP, and IP.";
                leaf name {
                  type string;
                  mandatory true;
                  description
                    "The name of the service.
                     This must be unique.";
                }
                leaf id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the service.
                     This must be unique.";
                }
                container protocol {
                  description
                    "Protocol types:
                     TCP, UDP, ICMP, ICMPv6, IP, and etc.";
                  leaf tcp  {
                    type boolean;
                    mandatory true;
                    description
                      "TCP protocol type.";
                  }
                  leaf udp {
                    type boolean;
                    mandatory true;
                    description
                      "UDP protocol type.";
                  }
                  leaf icmp {
                    type boolean;
                    mandatory true;
                    description
                      "ICMP protocol type.";
                  }
                  leaf icmpv6 {
```

```
                    type boolean;
                    mandatory true;
                    description
                      "ICMPv6 protocol type.";
                  }
                  leaf ip {
                    type boolean;
                    mandatory true;
                    description
                      "IP protocol type.";
                  }
                }
                leaf src-port{
                  type inet:port-number;
                  description
                    "It can be used for finding programs.";
                }
                leaf dest-port{
                  type inet:port-number;
                  description
                    "It can be used for finding programs.";
                }
              }
              container application-sec-context-cond {
                description
                  "An application is a computer program for
                   a specific task or purpose. It provides
                   a finer granularity than service in matching
                   traffic.";
                leaf name{
                  type string;
                  mandatory true;
                  description
                    "The name of the application.
                     This must be unique.";
                }
                leaf id{
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the application.
                     This must be unique.";
                }
                container category{
                  description
                    "Category types: Business system, Entertainment,
                     Interest, Network, General, and etc.";
                  leaf business-system {
```

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

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

```
                    Malware vehicle, Bandwidth consuming,
                    Tunneling, and etc.";
                  leaf exploitable {
                    type boolean;
                    description
                      "Exploitable risk level.";
                  }
                  leaf productivity-loss {
                    type boolean;
                    description
                      "Productivity loss risk level.";
                  }
                  leaf evasive {
                    type boolean;
                    description
                      "Evasive risk level.";
                  }
                  leaf data-loss {
                    type boolean;
                    description
                      "Data loss risk level.";
                  }
                  leaf malware-vehicle {
                    type boolean;
                    description
                      "Malware vehicle risk level.";
                  }
                  leaf bandwidth-consuming {
                    type boolean;
                    description
                      "Bandwidth consuming risk level.";
                  }
                  leaf tunneling {
                    type boolean;
                    description
                      "Tunneling risk level.";
                  }
                }
              }
              container device-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 tablet {
                  type boolean;
                  description
                    "If type of a device is tablet.";
                }
                leaf voip-volte-phone {
                  type boolean;
                  description
                    "If type of a device is voip-volte-phone.";
                }
              }
            }
            list user-security-cond {
              key "usr-sec-cond-id";
              description
                "TBD";
              leaf usr-sec-cond-id {
                type uint8;
                description
                  "The ID of the user-sec-cond.
                   This is key for user-sec-cond-list.
                   This must be unique.";
              }
              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.";
                  }
                }
              }
            }
          }
          list generic-context-condition {
            key "gen-context-cond-id";
            description
              "TBD";
            leaf gen-context-cond-id {
              type uint8;
              description
                "The ID of the gen-context-cond.
                 This is key for gen-context-cond-list.
                 This must be unique.";
            }
            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 geographic-location {
                type uint8;
                description
                  "This is mapped to ip address. We can acquire
                   region through ip address stored the database.";
              }
            }
          }
        }
        container action {
          description
            "TBD.";
          choice action-type {
          description
            "The flow-based NSFs realize the network security
             functions by executing various Actions, which at least
             includes ingress-action, egress-action, and
             advanced-action.";
          case ingress-action {
```

```
            description
              "The ingress actions consist of permit, deny,
               and mirror.";
            choice ingress-action-type {
              description
                "Ingress action type: permit, deny, and mirror.";
              case pass {
                description
                  "Pass case.";
                leaf pass {
                  type boolean;
                  mandatory true;
                  description
                    "Packet flow is passed.";
                }
              }
              case drop {
                description
                  "Drop case.";
                leaf drop {
                  type boolean;
                  mandatory true;
                  description
                    "Packet flow is droped.";
                }
              }
              case reject {
                description
                  "Reject case.";
                leaf reject {
                  type boolean;
                  mandatory true;
                  description
                    "Packet flow is rejected.";
                }
              }
              case alert {
                description
                  "Alert case.";
                leaf alert {
                  type boolean;
                  mandatory true;
                  description
                    "Packet flow is alerted.";
                }
              }
              case mirror {
                description
```

```
                  "Mirror case.";
                leaf mirror {
                  type boolean;
                  mandatory true;
                  description
                    "Packet flow is mirroried.";
                }
              }
            }
          }
          case egress-action {
            description
              "The egress actions consist of invoke-signaling,
              tunnel-encapsulation, and forwarding.";
            choice egress-action-type {
              description
                "Egress-action-type: invoke-signaling,
                 tunnel-encapsulation, and forwarding.";
              case invoke-signaling {
                description
                  "Invoke-signaling case.";
                leaf invoke-signaling {
                  type boolean;
                  mandatory true;
                  description
                    "TBD.";
                }
              }
              case tunnel-encapsulation {
                description
                  "tunnel-encapsulation case.";
                leaf tunnel-encapsulation {
                  type boolean;
                  mandatory true;
                  description
                    "TBD.";
                }
              }
              case forwarding {
                description
                  "forwarding case.";
                leaf forwarding {
                  type boolean;
                  mandatory true;
                  description
                    "TBD.";
                }
              }
```

```
            }
          }
          case apply-profile-action {
            description
              "Applying a specific Functional Profile or signature
               - e.g., an IPS Profile, a signature file, an
               anti-virus file, or a URL filtering file. The
               functional profile or signature file corresponds to
               the security capability for the content security
               control and attack mitigation control which will be
               described afterwards. It is one of the key properties
               that determine the effectiveness of the NSF, and is
               mostly vendor specific today. One goal of I2NSF is
               to standardize the form and functional interface of
               those security capabilities while supporting vendor-
               specific implementations of each.";
            choice apply-profile-action-type {
              description
                "Advanced action types: Content Security Control
                 and Attack Mitigation Control.";
              case content-security-control {
                description
                  "Content security control is another category of
                  security capabilities applied to application layer.
                  Through detecting the contents carried over the
                  traffic in application layer, these capabilities
                  can realize various security purposes, such as
                  defending against intrusion, inspecting virus,
                  filtering malicious URL or junk email, and blocking
                  illegal web access or data retrieval.";

                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.";
                  container antivirus {
                    description
                      "Antivirus is computer software used to
                       prevent, detect and remove malicious
                       software.";
                    leaf antivirus-insp {
                      type boolean;
                      description
                        "Additional inspection of antivirus.";
                    }
                  }
```

```
                container ips {
                  description
                    "Intrusion prevention systems (IPS) are
                     network security appliances that monitor
                     network and/or system activities for
                     malicious activities.";
                  leaf ips-insp {
                    type boolean;
                    description
                      "Additional inspection of IPS.";
                  }
                }
                container ids {
                  description
                    "IDS security service.";
                  leaf ids-insp {
                    type boolean;
                    description
                      "Additional inspection of IDS.";
                  }
                }
                container url-filtering {
                  description
                    "URL filtering security service.";
                  leaf url-filtering-insp {
                    type boolean;
                    description
                      "Additional inspection of URL filtering.";
                  }
                }
                container data-filtering {
                  description
                    "Data filtering security service.";
                  leaf data-filtering-insp {
                    type boolean;
                    description
                      "Additional inspection of data filtering.";
                  }
                }
                container mail-filtering {
                  description
                    "Mail filtering security service.";
                  leaf mail-filtering-insp {
                    type boolean;
                    description
                      "Additional inspection of mail filtering.";
                  }
                }
```

```
                container file-blocking {
                  description
                    "File blocking security service.";
                  leaf file-blocking-insp {
                    type boolean;
                    description
                      "Additional inspection of file blocking.";
                  }
                }
                container file-isolate {
                  description
                    "File isolate security service.";
                  leaf file-isolate-insp {
                    type boolean;
                    description
                      "Additional inspection of file isolate.";
                  }
                }
                container pkt-capture {
                  description
                    "Packet capture security service.";
                  leaf pkt-capture-insp {
                    type boolean;
                    description
                      "Additional inspection of packet capture.";
                  }
                }
                container application-control {
                  description
                    "app-control security service.";
                  leaf application-control-insp {
                    type boolean;
                    description
                      "Additional inspection of app control.";
                  }
                }
                container voip-volte {
                  description
                    "VoIP/VoLTE security service.";
                  leaf voip-volte-insp {
                    type boolean;
                    description
                      "Additional inspection of VoIP/VoLTE.";
                  }
                }
              }
            }
            case attack-mitigation-control {
```

```
                description
                  "This category of security capabilities is
                   specially used to detect and mitigate various
                   types of network attacks.";
                choice attack-mitigation-control-type {
                  description
                    "Attack-mitigation types: DDoS-attack and
                     Single-packet attack.";
                  case ddos-attack {
                    description
                      "A distributed-denial-of-service (DDoS) is
                       where the attack source is more than one,
                       often thousands of unique IP addresses.";
                    choice ddos-attack-type {
                      description
                        "DDoS-attack types: Network Layer DDoS Attacks
                         and Application Layer DDoS Attacks.";
                      case network-layer-ddos-attack {
                        description
                          "Network layer DDoS-attack.";
                        container network-layer-ddos-attack-types {
                          description
                            "Network layer DDoS attack types:
                             Syn Flood Attack, UDP Flood Attack,
                             ICMP Flood Attack, IP Fragment Flood,
                             IPv6 Related Attacks, and etc";
                          container syn-flood-attack {
                            description
                              "If the network layer DDoS-attack is
                               a syn flood attack.";
                            leaf syn-flood-insp {
                              type boolean;
                              mandatory true;
                              description
                                "Additional Inspection of
                                 Syn Flood Attack.";
                            }
                          }
                          container udp-flood-attack {
                            description
                              "If the network layer DDoS-attack is
                               a udp flood attack.";
                            leaf udp-flood-insp {
                              type boolean;
                              mandatory true;
                              description
                                "Additional Inspection of
                                 UDP Flood Attack.";
```

```
                    }
                  }
                  container icmp-flood-attack {
                    description
                      "If the network layer DDoS-attack is
                       an icmp flood attack.";
                    leaf icmp-flood-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         ICMP Flood Attack.";
                    }
                  }
                  container ip-frag-flood-attack {
                    description
                      "If the network layer DDoS-attack is
                       an ip fragment flood attack.";
                    leaf ip-frag-flood-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         IP Fragment Flood.";
                    }
                  }
                  container ipv6-related-attacks {
                    description
                      "If the network layer DDoS-attack is
                       ipv6 related attacks.";
                    leaf ipv6-related-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         IPv6 Related Attacks.";
                    }
                  }
                }
              }
              case 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 http-flood-attack {
                    description
                      "If the application layer DDoS-attack is
                       a http flood attack.";
                    leaf http-flood-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         Http Flood Attack.";
                    }
                  }
                  container https-flood-attack {
                    description
                      "If the application layer DDoS-attack is
                       a https flood attack.";
                    leaf https-flood-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         Https Flood Attack.";
                    }
                  }
                  container dns-flood-attack {
                    description
                      "If the application layer DDoS-attack is
                       a dns flood attack.";
                    leaf dns-flood-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         DNS Flood Attack.";
                    }
                  }
                  container dns-amp-flood-attack {
                    description
                      "If the application layer DDoS-attack is
                       a dns amplification flood attack.";
                    leaf dns-amp-flood-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         DNS Amplification Flood Attack.";
```

```
                    }
                  }
                  container ssl-ddos-attack {
                    description
                      "If the application layer DDoS-attack is
                       an ssl DDoS attack.";
                    leaf ssl-ddos-insp {
                      type boolean;
                      mandatory true;
                      description
                        "Additional Inspection of
                         SSL Flood Attack.";
                    }
                  }
                }
              }
            }
            case single-packet-attack {
              description
                "Single Packet Attacks.";
              choice single-packet-attack-type {
                description
                  "DDoS-attack types: Scanning Attack,
                   Sniffing Attack, Malformed Packet Attack,
                   Special Packet Attack, and etc.";
                case scan-and-sniff-attack {
                  description
                    "Scanning and Sniffing Attack.";
                  container scan-and-sniff-attack-types {
                    description
                      "Scanning and sniffing attack types:
                       IP Sweep attack, Port Scanning,
                       and etc.";
                    container ip-sweep-attack {
                      description
                        "If the scanning and sniffing attack is
                         an ip sweep attack.";
                      leaf ip-sweep-insp {
                        type boolean;
                        mandatory true;
                        description
                          "Additional Inspection of
                           IP Sweep Attack.";
                      }
                    }
                    container port-scanning-attack {
                      description
```

```
                    "If the scanning and sniffing attack is
                     a port scanning attack.";
                  leaf port-scanning-insp {
                    type boolean;
                    mandatory true;
                    description
                      "Additional Inspection of
                       Port Scanning Attack.";
                  }
                }
              }
            }
            case 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.";
                container ping-of-death-attack {
                  description
                    "If the malformed packet attack is
                     a ping of death attack.";
                  leaf ping-of-death-insp {
                    type boolean;
                    mandatory true;
                    description
                      "Additional Inspection of
                       Ping of Death Attack.";
                  }
                }
                container teardrop-attack {
                  description
                    "If the malformed packet attack is
                     a teardrop attack.";
                  leaf teardrop-insp {
                    type boolean;
                    mandatory true;
                    description
                      "Additional Inspection of
                       Teardrop Attack.";
                  }
                }
              }
            }
            case special-packet-attack {
              description
```

```
                          "special Packet Attack.";
                        container special-packet-attack-types {
                          description
                            "Special packet attack types:
                             Oversized ICMP Attack, Tracert Attack,
                             and etc.";
                          container oversized-icmp-attack {
                            description
                              "If the special packet attack is
                               an oversized icmp attack.";
                            leaf oversized-icmp-insp {
                              type boolean;
                              mandatory true;
                              description
                                "Additional Inspection of
                                 Oversize ICMP Attack.";
                            }
                          }
                          container tracert-attack {
                            description
                              "If the special packet attack is
                               a tracert attack.";
                            leaf tracert-insp {
                              type boolean;
                              mandatory true;
                              description
                                "Additional Inspection of
                                 Tracrt Attack.";
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
 }

 grouping cfg-content-security-conrol {
  description
    "Configuration for Content Security Control.";
```

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

    case cfg-antivirus {
      description
        "Antivirus Case.";

      list antivirus-rule {
        key rule-id;
        description
          "Rule of Antivirus.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about antivirus.";
        }
      }
    }
    case cfg-ips {
      description
        "IPS Case.";

      list ips-rule {
        key rule-id;
        description
          "Rule of IPS.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about IPS.";
        }
      }
    }
    case cfg-ids {
      description
        "IDS Case.";

      list ids-rule {
        key rule-id;
        description
```

```
          "Rule of IDS.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about IDS.";
        }
      }
    }
    case cfg-url-filter {
      description
        "URL Filter Case.";

      list url-filter-rule {
        key rule-id;
        description
          "Rule of URL filter.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about URL filter.";
        }
      }
    }
    case cfg-data-filter {
      description
        "Data Filter Case.";

      list data-filter-rule {
        key rule-id;
        description
          "Rule of Data Filter.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about data filter.";
        }
      }
    }
    case cfg-mail-filter {
      description
        "Mail Filter Case.";
```

```
      list mail-filter-rule {
        key rule-id;
        description
          "Rule of Mail Filter.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about mail filter.";
        }
      }
    }
    case cfg-file-blocking {
      description
        "File Blocking Case.";

      list file-blocking-rule {
        key rule-id;
        description
          "Rule of File Blocking.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about file blocking.";
        }
      }
    }
    case cfg-file-isolate {
      description
        "File Isolate Case.";

      list file-isolate-rule {
        key rule-id;
        description
          "Rule of File Isolate.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about file isolate.";
        }
      }
    }
    case cfg-pkt-capture {
```

```
      description
        "Packet Capture Case.";

      list pkt-capture-rule {
        key rule-id;
        description
          "Rule of Packet Capture.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about pacekt capture.";
        }
      }
    }
    case cfg-app-control {
      description
        "App Control Case.";

      list app-control-rule {
        key rule-id;
        description
          "Rule of App Control.";

        leaf rule-id {
          type uint8;
          mandatory true;
          description
            "The ID of the rule about app control.";
        }
      }
    }
    case cfg-voip-volte {
      description
       "VoIP/VoLTE Case.";

      list voip-volte-rule {
        key "rule-id";
        description
          "For the VoIP/VoLTE security system, a VoIP/
           VoLTE security system can monitor each
           VoIP/VoLTE flow and manage VoIP/VoLTE
           security rules controlled by a centralized
           server for VoIP/VoLTE security service
           (called VoIP IPS). The VoIP/VoLTE security
           system controls each switch for the
           VoIP/VoLTE call flow management by
```

```
           manipulating the rules that can be added,
           deleted, or modified dynamically.";

        leaf rule-id {
           type uint8;
           mandatory true;
           description
             "The ID of the voip-volte-rule.
              This is the key for voip-volte-rule-list.
              This must be unique.";
        }
        container event {
          description
            "Event types: VoIP and VoLTE.";
          leaf called-voip {
            type boolean;
            mandatory true;
            description
              "If content-security-control-type is
               voip.";
          }
          leaf called-volte {
            type boolean;
            mandatory true;
            description
               "If content-security-control-type is
                volte.";
          }
        }
        container condition {
          description
            "TBD.";
          list sip-header {
            key "sip-header-uri";
            description
              "TBD.";
            leaf sip-header-uri {
              type string;
              mandatory true;
              description
                "SIP header URI.";
            }
            leaf sip-header-method {
              type string;
              mandatory true;
              description
                "SIP header method.";
            }
```

```
            leaf sip-header-expire-time {
              type yang:date-and-time;
              mandatory true;
              description
                "SIP header expire time.";
            }
            leaf sip-header-user-agent {
              type uint32;
              mandatory true;
              description
                "SIP header user agent.";
            }
          }
          list cell-region {
            key "cell-id-region";
            description
              "TBD.";
            leaf cell-id-region {
              type uint32;
              mandatory true;
              description
                "Cell region.";
            }
          }
        }
        container action {
           description
             "The flow-based NSFs realize the security
              functions by executing various Actions.";
              choice action-type {
                 description
                   "Action type: ingress action and
                    egress action.";
                 case ingress-action {
                    description
                       "The ingress actions consist of permit,
                        deny, and mirror.";
                        choice ingress-action-type {
                           description
                              "Ingress-action-type: permit, deny,
                               and mirror.";
                           case pass {
                              description
                                "Pass case.";
                              leaf pass {
                                type boolean;
                                mandatory true;
                                description
```

```
                        "Packet flow is passed.";
                    }
                  }
                  case drop {
                    description
                      "Drop case.";
                    leaf drop {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is droped.";
                    }
                  }
                  case reject {
                    description
                      "Reject case.";
                    leaf reject {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is reject.";
                    }
                  }
                  case alert {
                    description
                      "Alert case.";
                    leaf alert {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is alert.";
                    }
                  }
                  case mirror {
                    description
                      "Mirror case.";
                    leaf mirror {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is mirrored.";
                    }
                  }
                }
              }
              case egress-action {
                 description
                    "The engress actions consist of
```

```
                        mirror and etc.";
                    choice egress-action-type {
                        description
                           "Engress-action-type: redirection,
                            and etc.";
                        case redirection {
                           description
                              "Redirection case.";
                           leaf redirection {
                              type boolean;
                              mandatory true;
                              description "TBD.";
                           }
                        }
                    }
                }
            }
        }
      }
    }
  }
 }

 grouping cfg-attack-mitigation-conrol {
  description
    "Configuration for Attack Mitigation Control.";

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

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

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

         case cfg-network-layer-ddos-attack {
            description
               "Network layer DDoS-attack.";

            choice cfg-network-layer-ddos-attack-type {
```

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

                case cfg-syn-flood-attack {
                   description
                     "Syn Flood Attack Case.";

                   list syn-flood-attack-rule {
                     key rule-id;
                     description
                       "Rule of Syn Flood Attack.";

                     leaf rule-id {
                       type uint8;
                       mandatory true;
                       description
                         "The ID of the rule about syn flood attack.";
                     }
                   }
                }
                case cfg-udp-flood-attack {
                   description
                     "UDP Flood Attack Case.";

                   list udp-flood-attack-rule {
                     key rule-id;
                     description
                       "Rule of UDP Flood Attack.";

                     leaf rule-id {
                       type uint8;
                       mandatory true;
                       description
                         "The ID of the rule about udp flood attack.";
                     }
                   }
                }
                case cfg-icmp-flood-attack {
                   description
                     "ICMP Flood Attack Case.";

                   list icmp-flood-attack-rule {
                     key rule-id;
                     description
                       "Rule of ICMP Flood Attack.";
```

```
                    leaf rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the rule about icmp flood attack.";
                    }
                  }
               }
               case cfg-ip-frag-flood-attack {
                  description
                    "IP Fragment Flood Attack Case.";

                  list ip-frag-flood-attack-rule {
                    key rule-id;
                    description
                      "Rule of Ip Fragment Flood Attack.";

                    leaf rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the rule about
                         ip fragment flood attack.";
                    }
                  }
               }
               case cfg-ipv6-related-attacks {
                  description
                    "IPv6 Related Attacks Case.";

                  list ipv6-related-attacks-rule {
                    key rule-id;
                    description
                      "Rule of Ipv6 Related Attacks.";

                    leaf rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the rule about
                         ipv6 related attacks.";
                    }
                  }
               }
            }
         }
         case cfg-app-layer-ddos-attack {
            description
```

```
            "Application layer DDoS-attack.";

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

             case cfg-http-flood-attack {
                description
                  "HTTP Flood Attack Case.";

                list http-flood-attack-rule {
                  key rule-id;
                  description
                    "Rule of HTTP Flood Attack.";

                  leaf rule-id {
                    type uint8;
                    mandatory true;
                    description
                      "The ID of the rule about
                       http flood attack.";
                  }
                }
             }
             case cfg-https-flood-attack {
                description
                  "HTTPs Flood Attack Case.";

                list https-flood-attack-rule {
                  key rule-id;
                  description
                    "Rule of HTTPs Flood Attack.";

                  leaf rule-id {
                    type uint8;
                    mandatory true;
                    description
                      "The ID of the rule about
                       https flood attack.";
                  }
                }
             }
             case cfg-dns-flood-attack {
                description
```

```
                    "DNS Flood Attack Case.";

                  list dns-flood-attack-rule {
                    key rule-id;
                    description
                      "Rule of DNS Flood Attack.";

                    leaf rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the rule about
                         dns flood attack.";
                    }
                  }
                }
                case cfg-dns-amp-flood-attack {
                  description
                    "DNS Amp Flood Attack Case.";

                  list dns-amp-flood-attack-rule {
                    key rule-id;
                    description
                      "Rule of DNS Amp Flood Attack.";

                    leaf rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the rule about
                         dns amp flood attack.";
                    }
                  }
                }
                case cfg-ssl-ddos-attack {
                  description
                    "SSL DDoS Attack Case.";

                  list ssl-ddos-attack-rule {
                    key rule-id;
                    description
                      "Rule of SSL DDoS Attack.";

                    leaf rule-id {
                      type uint8;
                      mandatory true;
                      description
                        "The ID of the rule about
```

```
                        ssl ddos attack.";
                    }
                  }
                }
              }
            }
          }
        }
        case cfg-single-packet-attack {
          description
            "Single Packet Attacks.";
          choice cfg-single-packet-attack-type {
            description
              "DDoS-attack types: Scanning Attack,
               Sniffing Attack, Malformed Packet Attack,
               Special Packet Attack, and etc.";
            case cfg-scan-and-sniff-attack {
               description
                 "Scanning and Sniffing Attack.";
               choice cfg-scan-and-sniff-attack-type {
                  description
                     "Scanning and sniffing attack types:
                      IP Sweep attack, Port Scanning,
                      and etc.";

                  case cfg-ip-sweep-attack {
                     description
                       "IP Sweep Attack Case.";

                     list ip-sweep-attack-rule {
                       key rule-id;
                       description
                         "Rule of IP Sweep Attack.";

                       leaf rule-id {
                         type uint8;
                         mandatory true;
                         description
                           "The ID of the rule about
                            ip sweep attack.";
                       }
                     }
                  }
                  case cfg-port-scanning-attack {
                     description
                       "Port Scanning Attack Case.";

                     list port-scanning-attack-rule {
```

```
                key rule-id;
                description
                  "Rule of Port Scanning Attack.";

                leaf rule-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the rule about
                     port scanning attack.";
                }
              }
            }
          }
        }
        case cfg-malformed-packet-attack {
          description
            "Malformed Packet Attack.";
          choice cfg-malformed-packet-attack-type {
            description
              "Malformed packet attack types:
               Ping of Death Attack, Teardrop Attack,
               and etc.";

             case cfg-ping-of-death-attack {
               description
                 "Ping of Death Attack Case.";

               list ping-of-death-attack-rule {
                 key rule-id;
                 description
                   "Rule of Ping of Death Attack.";

                 leaf rule-id {
                   type uint8;
                   mandatory true;
                   description
                     "The ID of the rule about
                      ping of death attack.";
                 }
               }
             }
             case cfg-teardrop-attack {
               description
                 "Teardrop Attack Case.";

               list teardrop-attack-rule {
                 key rule-id;
```

```
                description
                  "Rule of Teardrop Attack.";

                leaf rule-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the rule about
                     teardrop attack.";
                }
              }
            }
          }
        }
        case cfg-special-packet-attack {
          description
            "special Packet Attack.";
          choice cfg-special-packet-attack-type {
            description
              "Special packet attack types:
               Oversized ICMP Attack, Tracert Attack,
               and etc.";

            case cfg-oversized-icmp-attack {
              description
                "Oversized ICMP Attack Case.";

              list oversized-icmp-attack-rule {
                key rule-id;
                description
                  "Rule of Oversized ICMP Attack.";

                leaf rule-id {
                  type uint8;
                  mandatory true;
                  description
                    "The ID of the rule about
                     oversized icmp attack.";
                }
              }
            }
            case cfg-tracert-attack {
              description
                "Tracert Attack Case.";

              list tracert-attack-rule {
                key rule-id;
                description
```

```
                    "Rule of Tracert Attack.";

                  leaf rule-id {
                    type uint8;
                    mandatory true;
                    description
                      "The ID of the rule about
                       tracert attack.";
                  }
                }
              }
            }
          }
        }
      }
    }
  }

}

<CODE ENDS>
```

Figure 2: Data Model of I2NSF NSF Facing Interface

6. Security Considerations

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

7. Acknowledgements

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

This document has greatly benefited from inputs by Daeyoung Hyun, Hyoungshick Kim, Tae-Jin Ahn, and Se-Hui Lee.

8. References

8.1. Normative References

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

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

8.2. Informative References

[i2nsf-cap-interface-im] Xia, L., Strassner, J., Zhang, D., Li, K., Basile, C., Lioy, A., Lopez, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-00 (work in progress), Novemver 2016.

[i2rs-rib-data-model] Wang, L., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.

[supa-policy-info-model] Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-02 (work in progress), January 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-04 (work in progress), October 2016.

Appendix A. Changes from draft-kim-i2nsf-nsf-facing-interface-data-model-00

The following changes are made from draft-kim-i2nsf-nsf-facing-interface-data-model-00:

- Rules for network security (e.g., iptables) and contents security (e.g., Suricata) are added.
- Some lists are replaced with containers, and also some leafs are correspondingly replaced with leaf-lists.

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: wlsdyd0930@nate.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

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

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

Phone:
EMail: Frank.xialiang@huawei.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2017

A. Pastor
D. Lopez
Telefonica I+D
A. Shaw
Hewlett Packard Labs
July 4, 2016

Remote Attestation Procedures for Network Security Functions (NSFs) through the I2NSF Security Controller draft-pastor-i2nsf-vnsf-attestation-03

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 5, 2017.

Table of Contents

1. Introduction

As described in [I-D.pastor-i2nsf-merged-use-cases], the use of externally provided NSF implies several additional concerns in security. The most relevant threats associated with a externalized virtual platform are detailed in [I-D.ietf-i2nsf-framework]. As stated there, mutual authentication between the user and the NSF environment and, what is more important, the attestation of the elements in this environment by clients could address these threats to an acceptable level of risk. In particular:

- Any impersonation attempt (of the client or the NSF environment) will be minimized by mutual authentication, and since appropriate records of such authentications will be made available, events will be suitable for auditing in the case of an incident.

- Attestation of the NSF environment, especially when performed periodically, will allow clients to detect the alteration of the processing elements, or the installation of malformed elements, and mutual authentication will provide again an audit trail.

- 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.

- 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 initialises the client's NSFs and policies. Afterwards, user traffic from the client domain goes through the NSF platform which hosts the corresponding NSFs. The user's expectations of the platform behavior are thus twofold:

- The user traffic will be treated according to the client-specified NSFs and policies, and no other processing will be performed by the Security Controller or the platform itself (e.g. traffic eavesdropping).
- 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 this expectations.

3.1. First Step: Client-Agnostic Attestation

This is the first interaction between a client and a Security Controller: the client wants an attestation that proves it is connected to a genuine Security Controller before continuing with the authentication. In this context, two properties characterise 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 vNSFs. 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, or during the execution of the NSFs.

Support of static NSF attestation 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 a proof to the client that the instantiated NSFs and policies are the ones chosen.

Additionally to the NSF attestation at the moment of their instantiation, a continuous attestation of the NSF execution (based on the generation of periodic TPM integrity measurements) MAY be required by a client to ensure their security.

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.

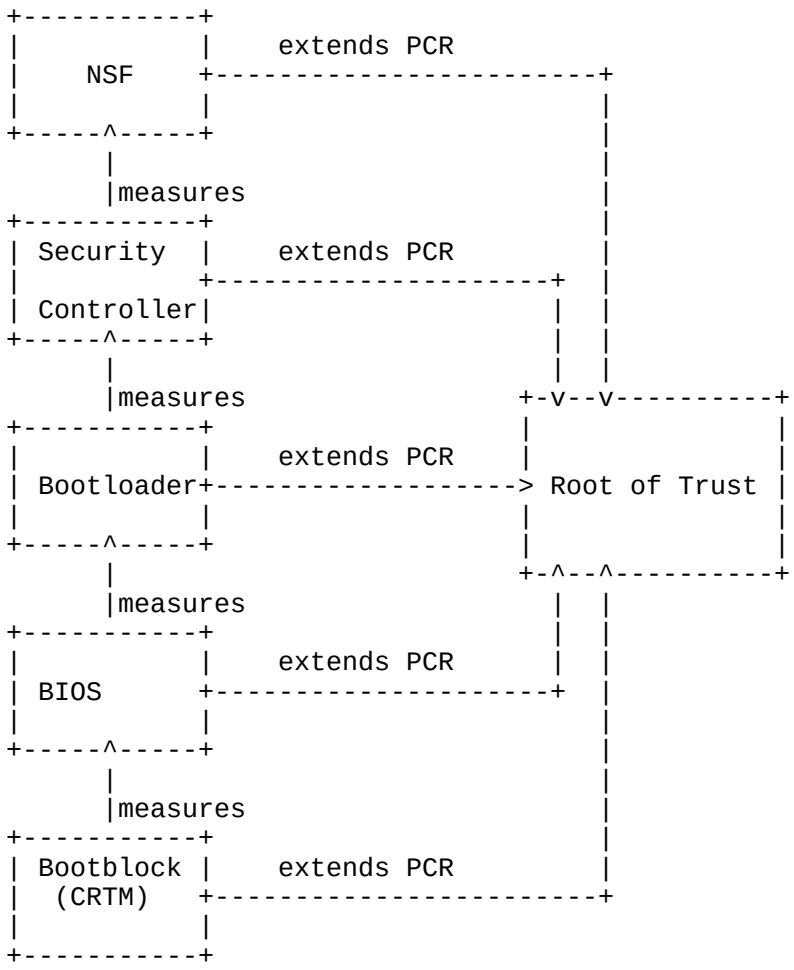


Figure 1: Applying Trusted Computing

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 the design and usage of a secure cryptoprocessor 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 bootkits 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.

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, relaying 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 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 user by the Security Controller during the initial mutual authentication phase.

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 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].

As well as for providing a signed audit log of boot measurements, 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 a criteria. This criteria can be that the platform booted into a particular state (e.g. a set of PCR values). Once the desired criteria is 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. Trusted Boot however 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 for 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.

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 unauthorised 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, that 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.

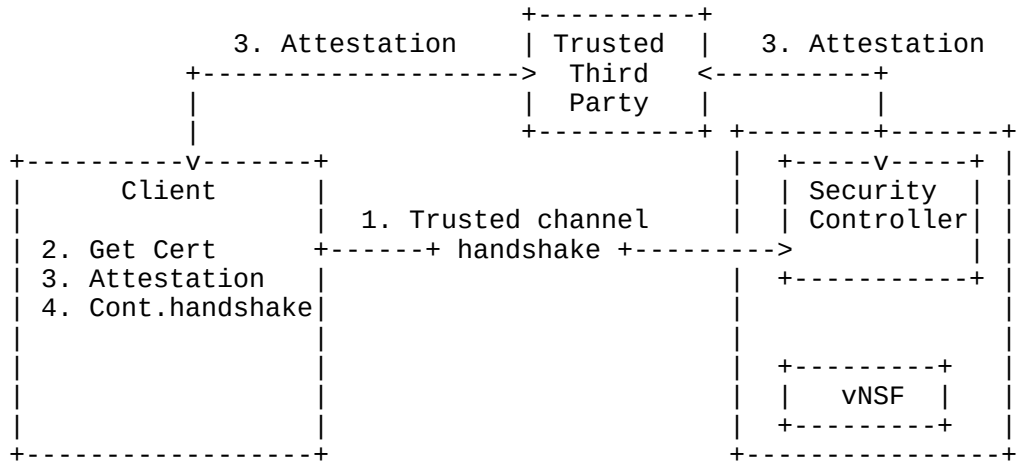


Figure 2: Steps for remote attestation

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, differently from the latter, requires the integrity verification of the contacted endpoint by the other peer during the initial handshake. 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, what 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 the hardware and software components involved in the boot process of the vNSF platform, in order to build the chain of trust.

Since a client may not have enough capabilities to perform the integrity verification of a Security Controller the client MAY request the status of a Security Controller to a Trusted Third Party (TTP), which is in charge of communicating with it. 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 performs 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, or requires an independent external element whom he can trust the measurements from, it may contact a TTP 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 within the platform. Platform attestation, instead, evaluates the integrity of the NSFs running within 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 independently 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 it was intended to be used. An attestation of the selection and policy mapping procedures constitute an interesting research matter, 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 rather costly operation, especially if periodic monitoring is required by the requested level of assurance, and there are several proposals to make them 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]
elopez@fortinet.com, e., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-02 (work in progress), July 2016.

[I-D.pastor-i2nsf-merged-use-cases]
Pastor, A., Lopez, D., Wang, K., Zhuang, X., Qi, M., Zarny, M., Majee, S., Leymann, N., Dunbar, L., and M. Georgiades, "Use Cases and Requirements for an Interface to Network Security Functions", draft-pastor-i2nsf-merged-use-cases-00 (work in progress), June 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[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
Zurbaran, 12
Madrid, 28010
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: September 13, 2017

L. Xia
Q. Lin
Huawei

March 12, 2017

Policy Object for Interface to Network Security Functions (I2NSF) draft-xia-i2nsf-security-policy-object-00

Abstract

This document describes policy objects used in the Interface to Network Security Functions (I2NSF) policy rules and defines the attributes of each policy object.

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 September 13, 2017.

Copyright Notice

Table of Contents

1. Introduction

I2NSF policy consists of policy rules that are used to provision NSF instances. The I2NSF policy rule is defined by using "Event-Condition-Action" (ECA) model described in Framework for Interface to Network Security Functions [I-D.ietf-i2nsf-framework]. In the ECA model, a condition is used to determine whether or not the predefined actions should be executed. A condition usually consists of several attributes. Information Model of NSFs Capabilities [I-D.ietf-i2nsf-capability] describes or illustrates attributes of different Condition subclasses. When configuring policy rules by

using attributes, it is no surprise to see that the same value of an attribute or the same value set of several attributes are configured for several times or more. And modifications of the policy rules are also very complex and time-consuming.

To facilitate the provisioning of NSF instances, this document describes a set of policy objects which are reusable and can be referenced by variable I2NSF policy rules. A policy object can be identified by a set of data items, such as IP addresses, TCP/UDP ports, and domain names. Each policy object is 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.

In this document, a set of policy objects are described, and for each policy object, several related attributes are defined.

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 [RFC2119].

3. Terminology

This document uses the terminology described in Interface to Network Security Functions (I2NSF) Terminology [I-D.ietf-i2nsf-terminology].

4. Policy Object

Policy objects are collections of commonly used condition attributes. Different policy objects consist of different attributes. For each policy object, a description of this policy object may be an optional attribute. The following figure shows the policy objects defined in this document.

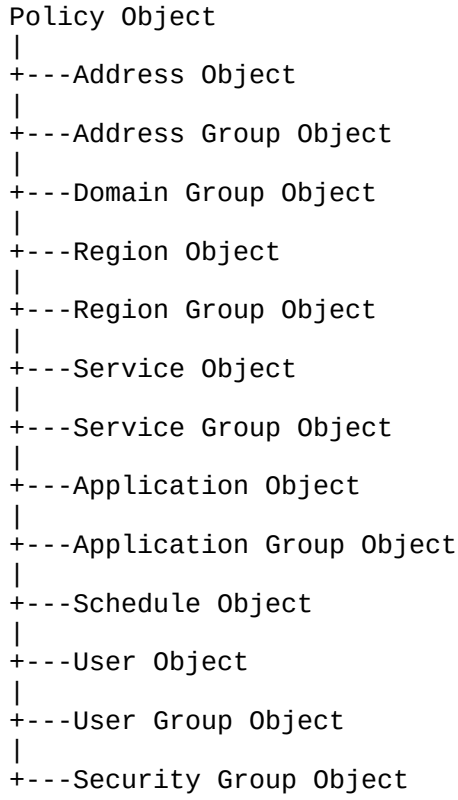


Figure 1: The policy objects

4.1. Address Object

An address object is a collection of IPv4/IPv6 addresses or MAC addresses. It consists of the following attributes:

4.1.1. The addressName Attribute

This attribute defines the unique name of the address object.

4.1.2. The addressRange Attribute

This attribute defines a set of IPv4/IPv6 addresses or MAC addresses, or a range of contiguous IPv4/IPv6 addresses.

The IPv4 address range can be defined by IPv4 address with wildcard mask, or IPv4 address with subnet mask (subnet mask address or length of the subnet mask), or the start address and the end address of the IPv4 address range.

The IPv6 address range can be defined by IPv6 address with length of the prefix, or the start address and the end address of the IPv6 address range.

4.2. Address Group Object

An address group object is composed of several address items that require the same policy enforcement. An address item can be an IPv4/IPv6 address, or a MAC address, or a range of contiguous IPv4/IPv6 addresses, or existing address object, or existing address group object. An address group object consists of the following attributes:

4.2.1. The addressGroupName Attribute

This attribute defines the unique name of the address group object.

4.2.2. The addressReference Attribute

This attribute refers to the existing address objects or existing address group objects identified by their unique names.

4.2.3. The addressRange Attribute

This attribute is the same as the addressRange attribute of address object. It can define a set of IPv4/IPv6 addresses or MAC addresses, or a range of contiguous IPv4/IPv6 addresses.

The IPv4 address range can be defined by IPv4 address with wildcard mask, or IPv4 address with subnet mask (subnet mask address or length of the subnet mask), or the start address and the end address of the IPv4 address range.

The IPv6 address range can be defined by IPv6 address with length of the prefix, or the start address and the end address of the IPv6 address range.

4.3. Domain Group Object

A domain group object is a collection of domain names that require the same policy enforcement. It consists of the following attributes:

4.3.1. The domainGroupName Attribute

This attribute defines the unique name of the domain group object.

4.3.2. The domainNameList Attribute

This attribute defines a set of domain names. The domain name can be matched in two modes: exact match and suffix match. Thus a domain name can be added by using the full string of the domain name (e.g., www.example.com) or a domain name begins with a wildcard (e.g., *.example.com).

4.4. Region Object

A region object is an IPv4/IPv6 address of a geographical region or a collection of IPv4/IPv6 addresses located in the same geographical region. A set of region objects which can be referenced directly should be predefined by NSFs. A region object consists of the following attributes:

4.4.1. The regionName Attribute

This attribute defines the unique name of the region object.

4.4.2. The regionLocation Attribute

This attribute defines the longitude and latitude of the region. It consists of two sub-attributes:

4.4.2.1. The regionLongitude Attribute

This attribute defines the longitude of the region.

4.4.2.2. The regionLatitude Attribute

This attribute defines the latitude of the region.

4.4.3. The regionIPAddress Attribute

This attribute defines a set of IPv4/IPv6 addresses or a range of contiguous IPv4/IPv6 addresses. And an IP address can only belong to one region object.

The IPv4 address range can be defined by IPv4 address with wildcard mask, IPv4 address with subnet mask (subnet mask address or length of the subnet mask), or the start address and the end address of the IPv4 address range.

The IPv6 address range can be defined by IPv6 address with length of the prefix, or the start address and the end address of the IPv6 address range.

4.5. Region Group Object

A region group object is a collection of region objects that require the same policy enforcement. It consists of the following attributes:

4.5.1. The regionGroupName Attribute

This attribute defines the unique name of the region group object.

4.5.2. The regionGroupReference Attribute

This attribute refers to the existing region objects or region group objects identified by their unique names.

4.6. Service Object

A service object is one or more services that can be identified by certain information, such as protocol type, source port number and destination port number. A set of well-known services should be predefined by NSFs as service objects to support direct reference. A service object consists of the following attributes:

4.6.1. The serviceName Attribute

This attribute defines the unique name of the service object.

4.6.2. The serviceList Attribute

This attribute defined a set of services. A service can be defined by the following sub-attributes.

4.6.2.1. The serviceProtocol Attribute

This attribute defines the protocol type of the service. The value of this attribute is selected from six types of protocols: TCP, UDP, SCTP, ICMP, ICMPv6 or IP.

4.6.2.2. The serviceProtocolNumber Attribute

This attribute defines the protocol number for IP protocol. The protocol number is the protocol field value in IP packet which identifies which kind of upper layer protocol is used.

4.6.2.3. The serviceSourcePort Attribute

This attribute defines the source port number range for TCP, UDP or SCTP protocol. A single port number or a range of port numbers can be set.

4.6.2.4. The serviceDestinationPort Attribute

This attribute defines the destination port number range for TCP, UDP or SCTP protocol. A single port number or a range of port numbers can be set.

4.6.2.5. The serviceICMPType Attribute

This attribute defines the ICMP/ICMPv6 type for ICMP or ICMPv6 protocol. The ICMP/ICMPv6 type can be identified by ICMP/ICMPv6 type number and ICMP/ICMPv6 message code. Thus, this attribute has two sub-attributes: serviceICMPTypeNumber and serviceICMPMessageCode.

The serviceICMPTypeNumber Attribute: It defines the ICMP/ICMPv6 type number and shall be defined together with the serviceICMPMessageCode attribute. For example, if the ICMP packet type is Echo, this attribute shall be set to 8 and the serviceICMPMessageCode attribute shall be set to 0.

The serviceICMPMessageCode Attribute: It defines the ICMP/ICMPv6 message code and shall be defined together with the serviceICMPTypeNumber attribute. For example, if the ICMP packet type is Echo, this attribute shall be set to 0 and the serviceICMPTypeNumber attribute shall be set to 8.

4.7. Service Group Object

A service group object is a collection of service objects that require the same policy enforcement. It consists of the following attributes:

4.7.1. The serviceGroupName Attribute

This attribute defines the unique name of the service group object.

4.7.2. The serviceReference Attribute

This attribute refers to the existing service objects or service group objects identified by their unique names.

4.8. Application Object

An application object is a kind of application that can be identified by several features, such as category, subcategory or risk level. A set of well-known application objects should be predefined by NSFs to support direct reference. An application object consists of the following attributes:

4.8.1. The applicationName Attribute

This attribute defines the unique name of the application object.

4.8.2. The applicationCategory Attribute

This attribute defines the category of the application. The value of this attribute is selected from a predefined set of categories, e.g., general category, network application category.

4.8.3. The applicationSubCategory Attribute

This attribute defines the subcategory of the application. The value of this attribute is selected from a predefined set of subcategories, e.g., search engine subcategory, electronic commerce subcategory.

4.8.4. The applicationTransmissionModel Attribute

This attribute defines the data transmission model of the application. The value of this attribute is selected from a predefined set of transmission models, e.g., client/server model, peer-to-peer model.

4.8.5. The applicationLabel Attribute

This attribute defines a set of labels for the application. The values of this attribute are selected from a predefined set of labels, e.g., database, encrypted-communication.

4.8.6. The applicationRiskLevel Attribute

This attribute defines a risk level for the application. The value of this attribute is selected from a predefined number of risk levels.

4.9. Application Group Object

An application group object is a collection of application objects that require the same policy enforcement. It consists of the following attributes:

4.9.1. The applicationGroupName Attribute

This attribute defines the unique name of the application group object.

4.9.2. The applicationReference Attribute

This attribute refers to the existing application objects or application group objects identified by their unique names.

4.10. Schedule Object

A schedule object is a set of time ranges. 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. A schedule object consists of the following attributes:

4.10.1. The scheduleName Attribute

This attribute defines the unique name of the schedule object.

4.10.2. The scheduleList Attribute

This attribute defines a set of time ranges. A time range can be defined by the following sub-attributes.

4.10.2.1. The scheduleType Attribute

This attribute defines the type of a time range. The value of this attribute is selected from the two types: periodic, absolute.

4.10.2.2. The scheduleStartTime Attribute

For a periodic time range, this attribute defines the start time in a day. For an absolute time range, this attribute defines the start time and start date.

4.10.2.3. The scheduleEndTime Attribute

For a periodic time range, this attribute defines the end time in a day. For an absolute time range, this attribute defines the end time and end date.

4.10.2.4. The scheduleWeekDay Attribute

This attribute defines the days in a week that the periodic time range takes effect.

4.11. User Object

A user object identifies a person who may access network resources. It is the basis of implementing user-based I2NSF policy. 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. A user object consists of the following attributes:

4.11.1. The userName Attribute

This attribute refers to the user name that used for user authentication.

4.11.2. The userParentGroup Attribute

This attribute refers to the existing parent user group object to which this user object belongs. The parent user group object is identified by its unique name. A user object can only belong to one user group object.

4.11.3. The userSecurityGroup Attribute

This attribute refers to the existing security group object to which this user object belongs. The security user group object is identified by its unique name. A user object can belong to several security group objects.

4.11.4. The userDomain Attribute

This attribute refers to the authentication domain to which this user object belongs.

4.11.5. The userPassword Attribute

If user is authenticated locally on the NSF, this attribute is mandatory. It defines the password corresponding to the user name.

4.11.6. The userExpirationTime Attribute

This attribute defines when will this user object expire.

4.11.7. The userAllowSharing Attribute

This attribute defines whether this user account identified by the userName and userPassword attribute is allowed to be shared by different persons. If allowed, this user object can be logged on to several devices simultaneously.

4.11.8. The userBindingStatus Attribute

This attribute defines whether the user object is bound to IP addresses, or MAC addresses, or IP/MAC address pairs. It is selected from three binding modes: no binding, unidirectional binding, and bidirectional binding. For no binding mode, the user object is not bound to any IP or MAC address or IP/MAC address pair. For unidirectional binding mode, the addresses or address pairs bound to this user object also can be bound to other users. For bidirectional binding mode, the addresses or address pairs bound to this user should not be bound to other bidirectional binding user object.

4.11.9. The userBindingAddress Attribute

This attribute defines the bound IP addresses, or MAC addresses, or IP/MAC address pairs. If the userBindingStatus is unidirectional binding or bidirectional binding, this attribute is mandatory.

4.12. User Group Object

A user object group is a collection of user objects that require the same policy enforcement and it usually corresponds to a physical entity such as a department. The user group objects are organized as a hierarchical structure. A user group object may belong to another user group object. The user group objects may be created locally on the NSFs, or be imported from third parties, such as authentication servers. It consists of the following attributes:

4.12.1. The userGroupName Attribute

This attribute defines the unique name of the user group object.

4.12.2. The userGroupParentGroup Attribute

This attribute refers to the existing parent user group object to which this user group object belongs. The parent user group object is identified by its unique name. A user group object can only belong to one parent user group object.

4.12.3. The userGroupDomain Attribute

This attribute refers to the authentication domain to which this user group object belongs.

4.12.4. The userGroupReference Attribute

This attribute refers to the existing user objects or user group objects which belong to this user group object.

4.12.5. The userGroupAllowSharing Attribute

This attribute defines whether the user objects of this user group object are allowed to be shared by different persons. If allowed, all user objects of this user group object can be logged on to several devices simultaneously.

4.13. Security Group Object

A security group object consists of user objects from different user group objects that require the same policy enforcement. The security group objects may be created locally on the NSFs, or be imported from third parties, such as authentication servers. This attribute consists of the following attributes:

4.13.1. The securityGroupName Attribute

This attribute defines the unique name of the security group object.

4.13.2. The securityGroupParentGroup Attribute

This attribute refers to the existing parent security group objects to which this security group object belongs. The parent security group objects are identified by their unique names.

4.13.3. The securityGroupDomain Attribute

This attribute refers to the authentication domain to which this security group object belongs.

4.13.4. The securityGroupType Attribute

This attribute defines the type of the security group object. There are two types: static and dynamic. For static security group, the member objects are fixed and added as required. For dynamic security group, the member objects are dynamically generated by setting filtering rules.

4.13.5. The securityGroupReference Attribute

This attribute defines the member objects for static security group object. It refers to the existing user objects or security group objects which belong to this security group object.

4.13.6. The securityGroupFilters Attribute

This attribute defines the filtering rules for dynamic security group object.

4.13.7. The securityGroupAllowSharing Attribute

This attribute defines whether the user objects of this security group object are allowed to be shared by different persons. If allowed, all user objects of this security group object can be logged on to several devices simultaneously.

5. Acknowledgements

6. IANA Considerations

This document requires no IANA actions.

7. Security Considerations

When the policy objects are transmitted, the integrity of these policy objects should be guaranteed. NSFs should verify that the modifications of policy objects come from the authenticated security controller. And NSF should protect the stored policy objects from being tampered.

8. References

8.1. Normative References

[I-D.ietf-i2nsf-capability]
Xia, L., Strassner, J., Zhang, D., Li, K., Basile, C., Lioy, A., Lopez, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of NSFs Capabilities", 2016, <https://tools.ietf.org/pdf/draft-xibassnez-i2nsf-capability-00.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[I-D.ietf-i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", 2016, <https://tools.ietf.org/pdf/draft-ietf-i2nsf-framework-04.pdf>.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", 2016, <https://tools.ietf.org/pdf/draft-ietf-i2nsf-terminology-03.pdf>.

Authors' Addresses

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

Email: linqiushi@huawei.com

I2NSF
Internet-Draft
Intended status: Standard Track
Expires: January 5, 2018

L. Xia
J. Strassner
Huawei
C. Basile
PoliTO
D. Lopez
TID

July 3, 2017

Information Model of NSFs Capabilities
draft-xibassnez-i2nsf-capability-02.txt

Abstract

This document defines the concept of an NSF (Network Security Function) Capability, as well as its information model. Capabilities are a set of features that are available from a managed entity, and are represented as data that unambiguously characterizes an NSF. Capabilities enable management entities to determine the set offer features from available NSFs that will be used, and simplify the management of 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 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 5, 2018.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

1. Introduction

The rapid development of virtualized systems requires advanced security protection in various scenarios. Examples include network devices in an enterprise network, User Equipment in a mobile network, devices in the Internet of Things, or residential access users [I-D.draft-ietf-i2nsf-problem-and-use-cases].

NSFs produced by multiple security vendors provide various security Capabilities to customers. Multiple NSFs can be combined together to provide security services over the given network traffic, regardless of whether the NSFs are implemented as physical or virtual functions.

Security Capabilities describe the set of network security-related features that are available to use for security policy enforcement purposes. Security Capabilities are independent of the actual security control mechanisms that will implement them. Every NSF registers the set of Capabilities it offers. Security Capabilities are a market enabler, providing a way to define customized security protection by unambiguously describing the security features offered by a given NSF. Moreover, Security Capabilities enable security functionality to be described in a vendor-neutral manner. That is, it is not required to refer to a specific product when designing the network; rather, the functionality characterized by their Capabilities are considered.

According to [I-D.draft-ietf-i2nsf-framework], there are two types of I2NSF interfaces available for security policy provisioning:

- Interface between I2NSF users and applications, and a security controller (Consumer-Facing Interface): this is a service-oriented interface that provides a communication channel between consumers of NSF data and services and the network operator's security controller. This enables security information to be exchanged between various applications (e.g., OpenStack, or various BSS/OSS components) and the security controller. The design goal of the Consumer-Facing Interface is to decouple the specification of security services from their implementation.

- o Interface between NSFs (e.g., firewall, intrusion prevention, or anti-virus) and the security controller (NSF-Facing Interface): The NSF-Facing Interface is used to decouple the security management scheme from the set of NSFs and their various implementations for this scheme, and is independent of how the NSFs are implemented (e.g., run in Virtual Machines or physical appliances). This document defines an object-oriented information model for network security, content security, and attack mitigation Capabilities, along with associated I2NSF Policy objects.

This document is organized as follows. Section 2 defines conventions and acronyms used. Section 3 discusses the design principles for the I2NSF Capability information model and related policy model objects. Section 4 defines the structure of the information model, which describes the policy and capability objects design; details of the model elements are contained in the appendices.

2. Conventions used in this document

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 [RFC2119].

This document uses terminology defined in [I-D.draft-ietf-i2nsf-terminology] for security related and I2NSF scoped terminology.

2.1. Acronyms

AAA:	Access control, Authorization, Authentication
ACL:	Access Control List
(D)DoD:	(Distributed) Denial of Service (attack)
ECA:	Event-Condition-Action
FMR:	First Matching Rule (resolution strategy)
FW:	Firewall
GNSF:	Generic Network Security Function
HTTP:	HyperText Transfer Protocol
I2NSF:	Interface to Network Security Functions
IPS:	Intrusion Prevention System
LMR:	Last Matching Rule (resolution strategy)
MIME:	Multipurpose Internet Mail Extensions
NAT:	Network Address Translation
NSF:	Network Security Function
RPC:	Remote Procedure Call
SMA:	String Matching Algorithm
URL:	Uniform Resource Locator
VPN:	Virtual Private Network

3. Information Model Design

The starting point of the design of the Capability information model is the categorization of types of security functions. For instance, experts agree on what is meant by the terms "IPS", "Anti-Virus", and "VPN concentrator". Network security experts unequivocally refer to "packet filters" as stateless devices able to allow or deny packet forwarding based on various conditions (e.g., source and destination IP addresses, source and destination ports, and IP protocol type fields) [Alshaer].

However, more information is required in case of other devices, like stateful firewalls or application layer filters. These devices filter packets or communications, but there are differences in the packets and communications that they can categorize and the states they maintain. Analogous considerations can be applied for channel protection protocols, where we all understand that they will protect packets by means of symmetric algorithms whose keys could have been negotiated with asymmetric cryptography, but they may work at different layers and support different algorithms and protocols. To ensure protection, these protocols apply integrity, optionally confidentiality, anti-reply protections, and authenticate peers.

3.1. Capability Information Model Overview

This document defines a model of security Capabilities that provides the foundation for automatic management of NSFs. This includes enabling the security controller to properly identify and manage NSFs, and allow NSFs to properly declare their functionality, so that they can be used in the correct way.

Some basic design principles for security Capabilities and the systems that have to manage them are:

- Independence: each security Capability should be an independent function, with minimum overlap or dependency on other Capabilities. This enables each security Capability to be utilized and assembled together freely. More importantly, changes to one Capability will not affect other Capabilities. This follows the Single Responsibility Principle [Martin] [OODSRP].
- Abstraction: each Capability should be defined in a vendor-independent manner, and associated to a well-known interface to provide a standardized ability to describe and report its processing results. This facilitates multi-vendor interoperability.
- Automation: the system must have the ability to auto-discover, auto-negotiate, and auto-update its security Capabilities (i.e., without human intervention). These features are especially useful for the management of a large number of NSFs. They are essential to add smart services (e.g., analysis,

refinement, Capability reasoning, and optimization) for the security scheme employed. These features are supported by many design patterns, including the Observer Pattern [OODOP], the Mediator Pattern [OODMP], and a set of Message Exchange Patterns [Hohpe].

- o Scalability: the management system must have the Capability to scale up/down or scale in/out. Thus, it can meet various performancerequirements derived from changeable network traffic or service requests. In addition, security Capabilities that are affected by scalability changes must support reporting statistics to the security controller to assist its decision on whether it needs to invoke scaling or not. However, this requirement is for information only, and is beyond the scope of this document.

Based on the above principles, a set of abstract and vendor-neutral Capabilities with standard interfaces is defined. This provides a Capability model that enables a set of NSFs that are required at a given time to be selected, as well as the unambiguous definition of the security offered by the set of NSFs used. The security controller can compare the requirements of users and applications to the set of Capabilities that are currently available in order to choose which NSFs are needed to meet those requirements. Note that this choice is independent of vendor, and instead relies specifically on the Capabilities (i.e., the description) of the functions provided. The security controller may also be able to customize the functionality of selected NSFs.

Furthermore, when an unknown threat (e.g., zero-day exploits and unknown malware) is reported by a NSF, new Capabilities may be created, and/or existing Capabilities may be updated (e.g., by updating its signature and algorithm). This results in enhancing existing NSFs (and/or creating new NSFs) to address the new threats. New Capabilities may be sent to and stored in a centralized repository, or stored separately in a vendor’s local repository. In either case, a standard interface facilitates the update process.

Note that most systems cannot dynamically create a new Capability without human interaction. This is an area for further study.

3.2. ECA Policy Model Overview

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules; definitions of all I2NSF policy-related terms are also defined in [I-D.draft-ietf-i2nsf-terminology]:

- o Event: 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.

Examples of an I2NSF Event include time and user actions (e.g., logon, logoff, and actions that violate an ACL).

- Condition: 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.
- Action: 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.

An I2NSF Policy Rule is made up of three Boolean clauses: an Event clause, a Condition clause, and an Action clause. A Boolean clause is a logical statement that evaluates to either TRUE or FALSE. It may be made up of one or more terms; if more than one term, then a Boolean clause connects the terms using logical connectives (i.e., AND, OR, and NOT). It has the following semantics:

```
IF <event-clause> is TRUE
   IF <condition-clause> is TRUE
      THEN execute <action-clause>
   END-IF
END-IF
```

Technically, the "Policy Rule" is really a container that aggregates the above three clauses, as well as metadata.

The above ECA policy model is very general and easily extensible, and can avoid potential constraints that could limit the implementation of generic security Capabilities.

3.3. Relation with the External Information Model

Note: the symbology used from this point forward is taken from section 3.3 of [I-D.draft-ietf-supa-generic-policy-info-model].

The I2NSF NSF-Facing Interface is in charge of selecting and managing the NSFs using their Capabilities. This is done using the following approach:

1) Each NSF registers its Capabilities with the management system when it "joins", and hence makes its Capabilities available to the management system;
2) The security controller selects the set of Capabilities required to meet the needs of the security service from all available NSFs that it manages;

3) The security controller uses the Capability information model to match chosen Capabilities to NSFs, independent of vendor;
4) The security controller takes the above information and creates or uses one or more data models from the Capability information model to manage the NSFs;
5) Control and monitoring can then begin.

This assumes that an external information model is used to define the concept of an ECA Policy Rule and its components (e.g., Event, Condition, and Action objects). This enables I2NSF Policy Rules [I-D.draft-ietf-i2nsf-terminology] to be subclassed from an external information model.

Capabilities are defined as classes (e.g., a set of objects that exhibit a common set of characteristics and behavior [I-D.draft-ietf-supa-generic-policy-info-model].

Each Capability is made up of at least one model element (e.g., attribute, method, or relationship) that differentiates it from all other objects in the system. Capabilities are, generically, a type of metadata (i.e., information that describes, and/or prescribes, the behavior of objects); hence, it is also assumed that an external information model is used to define metadata (preferably, in the form of a class hierarchy). Therefore, it is assumed that Capabilities are subclassed from an external metadata model.

The Capability sub-model is used for advertising, creating, selecting, and managing a set of specific security Capabilities independent of the type and vendor of device that contains the NSF. That is, the user of the NSF-Facing Interface does not care whether the NSF is virtualized or hosted in a physical device, who the vendor of the NSF is, and which set of entities the NSF is communicating with (e.g., a firewall or an IPS). Instead, the user only cares about the set of Capabilities that the NSF has, such as packet filtering or deep packet inspection. The overall structure is illustrated in the figure below:

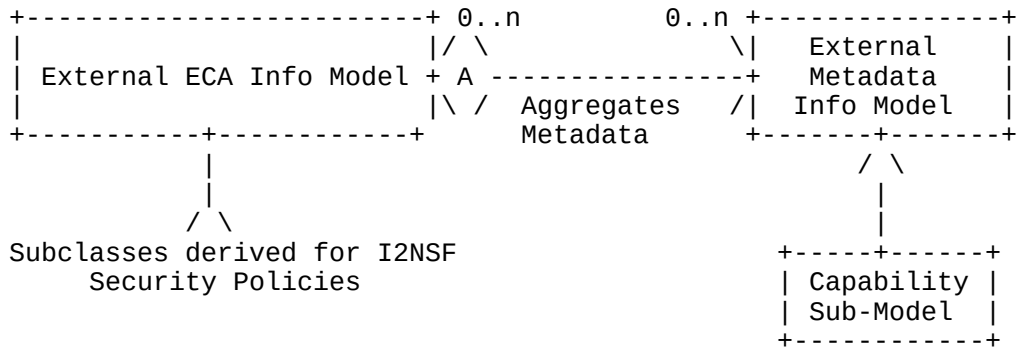


Figure 1. The Overall I2NSF Information Model Design

This draft defines a set of extensions to a generic, external, ECA Policy Model to represent various NSF ECA Security Policy Rules. It also defines the Capability Sub-Model; this enables ECA Policy Rules to control which Capabilities are seen by which actors, and used by the I2NSF system. Finally, it places requirements on what type of extensions are required to the generic, external, ECA information model and metadata models, in order to manage the lifecycle of I2NSF Capabilities.

Both of the external models shown in Figure 1 could, but do not have to, be based on the SUPA information model [I-D.draft-ietf-supa-generic-policy-info-model]. Note that classes in the Capability Sub-Model will inherit the AggregatesMetadata aggregation from the External Metadata Information Model.

The external ECA Information Model supplies at least a set of classes that represent a generic ECA Policy Rule, and a set of classes that represent Events, Conditions, and Actions that can be aggregated by the generic ECA Policy Rule. This enables I2NSF to reuse this generic model for different purposes, as well as refine it (i.e., create new subclasses, or add attributes and relationships) to represent I2NSF-specific concepts.

It is assumed that the external ECA Information Model has the ability to aggregate metadata. Capabilities are then sub-classed from an appropriate class in the external Metadata Information Model; this enables the ECA objects to use the existing aggregation between them and Metadata to add Metadata to appropriate ECA objects.

Detailed descriptions of each portion of the information model are given in the following sections.

3.4. I2NSF Capability Information Model: Theory of Operation

Capabilities are typically used to represent NSF functions that can be invoked. Capabilities are objects, and hence, can be used in the event, condition, and/or action clauses of an I2NSF ECA Policy Rule. The I2NSF Capability information model refines a predefined metadata model; the application of I2NSF Capabilities is done by refining a predefined ECA Policy Rule information model that defines how to use, manage, or otherwise manipulate a set of Capabilities. In this approach, an I2NSF Policy Rule is a container that is made up of three clauses: an event clause, a condition clause, and an action clause. When the I2NSF policy engine receives a set of events, it matches those events to events in active ECA Policy Rules. If the event matches, then this triggers the evaluation of the condition clause of the matched I2NSF Policy Rule. The condition clause is then evaluated; if it matches, then the set of actions in the matched I2NSF Policy Rule MAY be executed.

This document defines additional important extensions to both the external ECA Policy Rule model and the external Metadata model that are used by the I2NSF Information Model; examples include resolution strategy, external data, and default action. All these extensions come from the geometric model defined in [Bas12]. A more detailed description is provided in Appendix E; a summary of the important points follows.

Formally, given a set of actions in an I2NSF Policy Rule, the resolution strategy maps all the possible subsets of actions to an outcome. In other words, the resolution strategy is included in the I2NSF Policy Rule to decide how to evaluate all the actions in a particular I2NSF Policy Rule. This is then extended to include all possible I2NSF Policy Rules that can be applied in a particular scenario. Hence, the final action set from all I2NSF Policy Rules is deduced.

Some concrete examples of resolution strategy are the First Matching Rule (FMR) or Last Matching Rule (LMR) resolution strategies. When no rule matches a packet, the NSFs may select a default action, if they support one.

Resolution strategies may use, besides intrinsic rule data (i.e., event, condition, and action clauses), "external data" associated to each rule, such as priority, identity of the creator, and creation time. Two examples of this are attaching metadata to the policy action and/or policy rule, and associating the policy rule with another class to convey such information.

3.4.1. I2NSF Condition Clause Operator Types

After having analyzed the literature and some existing NSFs, the types of selectors are categorized as exact-match, range-based, regex-based, and custom-match [Bas15][Lunt].

Exact-match selectors are (unstructured) sets: elements can only be checked for equality, as no order is defined on them. As an example, the protocol type field of the IP header is an unordered set of integer values associated to protocols. The assigned protocol numbers are maintained by the IANA (http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml).

In this selector, it is only meaningful to specify condition clauses that use either the "equals" or "not equals" operators:

```
proto = tcp, udp       (protocol type field equals to TCP or UDP)
proto != tcp           (protocol type field different from TCP)
```

No other operators are allowed on exact-match selectors. For example, the following is an invalid condition clause, even if protocol types map to integers:

```
proto < 62              (invalid condition)
```

Range-based selectors are ordered sets where it is possible to naturally specify ranges as they can be easily mapped to integers. As an example, the ports in the TCP protocol may be represented with a range-based selector (e.g., 1024-65535). As another example, the following are examples of valid condition clauses:

```
source_port = 80
source_port < 1024
source_port < 30000 && source_port >= 1024
```

We include, in range-based selectors, the category of selectors that have been defined by Al-Shaer et al. as "prefix-match" [Alshaer]. These selectors allow the specification of ranges of values by means of simple regular expressions. The typical case is the IP address selector (e.g., 10.10.1.*).

There is no need to distinguish between prefix match and range-based selectors; for example, the address range "10.10.1.*" maps to "[10.10.1.0,10.10.1.255]".

Another category of selector types includes those based on regular expressions. This selector type is used frequently at the application layer, where data are often represented as strings of text. The regex-based selector type also includes string-based selectors, where matching is evaluated using string matching algorithms (SMA) [Cormen]. Indeed, for our purposes, string matching can be mapped to regular expressions, even if in practice SMA are much faster. For instance, Squid (http://www.squid-cache.org/), a popular Web caching proxy that offers various access control Capabilities, allows the definition of conditions on URLs that can be evaluated with SMA (e.g., dstdomain) or regex matching (e.g., dstdom_regex).

As an example, the condition clause:

```
"URL = *.website.*"
```

matches all the URLs that contain a subdomain named website and the ones whose path contain the string ".website.". As another example, the condition clause:

```
"MIME_type = video/*"
```

matches all MIME objects whose type is video.

Finally, the idea of a custom check selector is introduced. For instance, malware analysis can look for specific patterns, and returns a Boolean value if the pattern is found or not.

In order to be properly used by high-level policy-based processing systems (such as reasoning systems and policy translation systems), these custom check selectors can be modeled as black-boxes (i.e., a function that has a defined set of inputs and outputs for a particular state), which provide an associated Boolean output.

More examples of custom check selectors will be presented in the next versions of the draft. Some examples are already present in Section 6.

3.4.2. Capability Selection and Usage

Capability selection and usage are based on the set of security traffic classification and action features that an NSF provides; these are defined by the Capability model. If the NSF has the classification features needed to identify the packets/flows required by a policy, and can enforce the needed actions, then that particular NSF is capable of enforcing the policy.

NSFs may also have specific characteristics that automatic processes or administrators need to know when they have to generate configurations, like the available resolution strategies and the possibility to set default actions.

The Capability information model can be used for two purposes: describing the features provided by generic security functions, and describing the features provided by specific products. The term Generic Network Security Function (GNSF) refers to the classes of security functions that are known by a particular system. The idea is to have generic components whose behavior 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. GNSF examples include packet filter, URL filter, HTTP filter, VPN gateway, anti-virus, anti-malware, content filter, monitoring, and anonymity proxy; these will be described later in a revision of this draft as well as in an upcoming data model contribution.

The next section will introduce the algebra to define the information model of Capability registration. This associates NSFs to Capabilities, and checks whether a NSF has the Capabilities needed to enforce policies.

3.4.3. Capability Algebra

We introduce a Capability Algebra to ensure that the actions of different policy rules do not conflict with each other.

Formally, two I2NSF Policy Actions conflict with each other if:

- o the event clauses of each evaluate to TRUE
- o the condition clauses of each evaluate to TRUE
- o the action clauses affect the same object in different ways

For example, if we have two Policies:

P1: During 8am-6pm, if traffic is external, then run through FW
P2: During 7am-8pm, conduct anti-malware investigation

There is no conflict between P1 and P2, since the actions are different. However, consider these two policies:

P3: During 8am-6pm, John gets GoldService
P4: During 10am-4pm, FTP from all users gets BronzeService

P3 and P4 are now in conflict, because between the hours of 10am and 4pm, the actions of P3 and P4 are different and apply to the same user (i.e., John).

Let us define the concept of a "matched" policy rule as one in which its event and condition clauses both evaluate to true. This enables the actions in this policy rule to be evaluated. Then, the conflict matrix is defined by a 5-tuple {Ac, Cc, Ec, RSc, Dc}, where:

- o Ac is the set of Actions currently available from the NSF;
- o Cc is the set of Conditions currently available from the NSF;
- o Ec is the set of Events the NSF is able to respond to. Therefore, the event clause of an I2NSF ECA Policy Rule that is written for an NSF will only allow a set of designated events in Ec. For compatibility purposes, we will assume that if Ec={} (that is, Ec is empty), the NSF only accepts CA policies.
- o RSc is the set of Resolution Strategies that 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;
- o Dc defines the notion of a Default action that 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. This field of the Capability algebra can take the following values:
  - An explicit action (that has been predefined; typically, this means that it is fixed and not configurable), denoted as Dc ={a}. In this case, the NSF will always use the action as as the default action.
  - A set of explicit actions, denoted Dc={a1,a2, ...}; typically, this means that any **one** action can be used as the default action. This enables the policy writer to choose one of a predefined set of actions {a1, a2, ...} to serve as the default action.

- A fully configurable default action, denoted as Dc={F}. Here, F is a dummy symbol (i.e., a placeholder value) that can be used to indicate that the default action can be freely selected by the policy editor from the actions Ac available at the NSF. In other words, one of the actions Ac may be selected by the policy writer to act as the default action.
- No default action, denoted as Dc={}, for cases where the NSF does not allow the explicit selection of a default action.

```
*** Note to WG: please review the following paragraphs
*
*  Interesting Capability concepts that could be considered for a next
*  version of the Capability model and algebra include:
*
*     o Event clause representation (e.g., conjunctive vs. disjunctive
*       normal form for Boolean clauses)
*     o Event clause evaluation function, which would enable more
*       complex expressions than simple Boolean expressions to be used
*
*
*     o Condition clause representation (e.g., conjunctive vs.
*       disjunctive normal form for Boolean clauses)
*     o Condition clause evaluation function, which would enable more
*       complex expressions than simple Boolean expressions to be used
*     o Action clause evaluation strategies (e.g., execute first
*       action only, execute last action only, execute all actions,
*       execute all actions until an action fails)
*     o The use of metadata, which can be associated to both an I2NSF
*       Policy Rule as well as objects contained in the I2NSF Policy
*       Rule (e.g., an action), that describe the object and/or
*       prescribe behavior. Descriptive examples include adding
*       authorship information and defining a time period when an NSF
*       can be used to be defined; prescriptive examples include
*       defining rule priorities and/or ordering.
*
*  Given two sets of Capabilities, denoted as
*
*     cap1=(Ac1,Cc1,Ec1,RSc1,Dc1) and
*     cap2=(Ac2,Cc2,Ec2,RSc2,Dc2),
*
*  two set operations are defined for manipulating Capabilities:
*
*     o Capability addition:
*          cap1+cap2 = {Ac1 U Ac2, Cc1 U Cc2, Ec1 U Ec2, RSc1, Dc1}
*     o Capability subtraction:
*          cap1-cap2 = {Ac1 \ Ac2, Cc1 \ Cc2, Ec1 \ Ec2, RSc1, Dc1}
*
*  In the above formulae, "U" is the set union operator and "\" is the
*  set difference operator.
```

* The addition and subtraction of Capabilities are defined as the
* addition (set union) and subtraction (set difference) of both the
* Capabilities and their associated actions. Note that **only** the
* leftmost (in this case, the first matched policy rule) Resolution
* Strategy and Default Action are used.
*
* Note: actions, events, and conditions are **symmetric**. This means
* that when two matched policy rules are merged, the resultant actions
* and Capabilities are defined as the union of each individual matched
* policy rule. However, both resolution strategies and default actions
* are **asymmetric** (meaning that in general, they can **not** be
* combined, as one has to be chosen). In order to simplify this, we
* have chosen that the **leftmost** resolution strategy and the
* **leftmost** default action are chosen. This enables the developer
* to view the leftmost matched rule as the "base" to which other
* elements are added.
*
* As an example, assume that a packet filter Capability, Cpf, is
* defined. Further, assume that a second Capability, called Ctime,
* exists, and that it defines time-based conditions. Suppose we need
* to construct a new generic packet filter, Cpfgen, that adds
* time-based conditions to Cpf.
*
*
* Conceptually, this is simply the addition of the Cpf and Ctime
* Capabilities, as follows:
*   Apf   = {Allow, Deny}
*   Cpf   = {IPsrc,IPdst,Psrc,Pdst,protType}
*   Epf   = {}
*   RSpf = {FMR}
*   Dpf   = {A1}
*
*   Atime = {Allow, Deny, Log}
*   Ctime = {timestart, timeend, datestart, datestop}
*   Etime = {}
*   RStime = {LMR}
*   Dtime = {A2}
*
* Then, Cpfgen is defined as:
*   Cpfgen = {Apf U Atime, Cpf U Ctime, Epf U Etime, RSpf, Dpf}
*   = {Allow, Deny, Log},
*   {{IPsrc, IPdst, Psrc, Pdst, protType} U
*   {timestart, timeend, datestart, datestop}},
*   {},
*   {FMR},
*   {A1}
*
* In other words, Cpfgen provides three actions (Allow, Deny, Log),
* filters traffic based on a 5-tuple that is logically ANDed with a
* time period, and uses FMR; it provides A1 as a default action, and
* it does not react to events.

* Note: We are investigating, for a next revision of this draft, the
* possibility to add further operations that do not follow the
* symmetric vs. asymmetric properties presented in the previous note.
* We are looking for use cases that may justify the complexity added
* by the availability of more Capability manipulation operations.
*
*** End Note to WG

3.5. Initial NSFs Capability Categories

The following subsections define three common categories of Capabilities: network security, content security, and attack mitigation. Future versions of this document may expand both the number of categories as well as the types of Capabilities within a given category.

3.5.1. Network Security Capabilities

Network security is a category that describes the inspecting and processing of network traffic based on the use of pre-defined security policies.

The inspecting portion may be thought of as a packet-processing engine that inspects packets traversing networks, either directly or in the context of flows with which the packet is associated. From the perspective of packet-processing, implementations differ in the depths of packet headers and/or payloads they can inspect, the various flow and context states they can maintain, and the actions that can be applied to the packets or flows.

3.5.2. Content Security Capabilities

Content security is another category of security Capabilities applied to the application layer. Through analyzing traffic contents carried in, for example, the application layer, content security Capabilities can be used to identify various security functions that are required. These include defending against intrusion, inspecting for viruses, filtering malicious URL or junk email, blocking illegal web access, or preventing malicious data retrieval.

Generally, each type of threat in the content security category has a set of unique characteristics, and requires handling using a set of methods that are specific to that type of content. Thus, these Capabilities will be characterized by their own content-specific security functions.

3.5.3. Attack Mitigation Capabilities

This category of security Capabilities is used to detect and mitigate various types of network attacks. Today's common network attacks can be classified into the following sets:

- o DDoS attacks:
  - Network layer DDoS attacks: Examples include SYN flood, UDP flood, ICMP flood, IP fragment flood, IPv6 Routing header attack, and IPv6 duplicate address detection attack;
  - Application layer DDoS attacks: Examples include HTTP flood, https flood, cache-bypass HTTP floods, WordPress XML RPC floods, and ssl DDoS.
- o Single-packet attacks:
  - Scanning and sniffing attacks: IP sweep, port scanning, etc.
  - malformed packet attacks: Ping of Death, Teardrop, etc.
  - special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc.

Each type of network attack has its own network behaviors and packet/flow characteristics. Therefore, each type of attack needs a special security function, which is advertised as a set of Capabilities, for detection and mitigation. The implementation and management of this category of security Capabilities of attack mitigation control is very similar to the content security control category. A standard interface, through which the security controller can choose and customize the given security Capabilities according to specific requirements, is essential.

4. Information Sub-Model for Network Security Capabilities

The purpose of the Capability Information Sub-Model is to define the concept of a Capability, and enable Capabilities to be aggregated to appropriate objects. The following sections present the Network Security, Content Security, and Attack Mitigation Capability sub-models.

4.1. Information Sub-Model for Network Security

The purpose of the Network Security Information Sub-Model is to define how network traffic is defined, and determine if one or more network security features need to be applied to the traffic or not. Its basic structure is shown in the following figure:

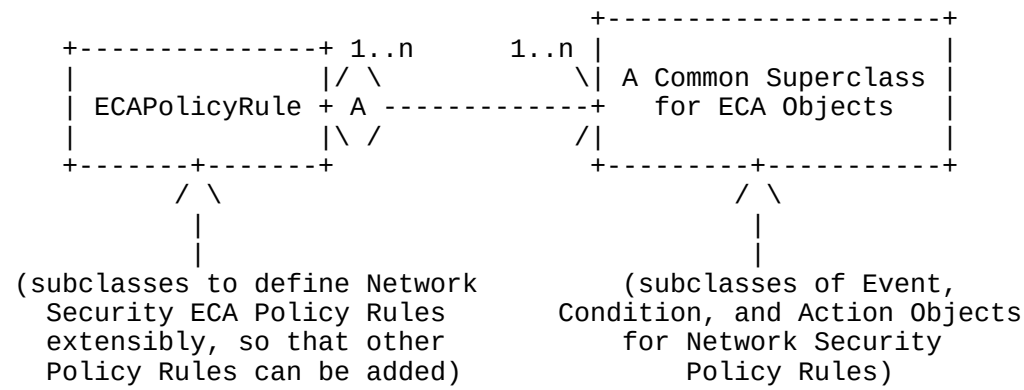


Figure 2. Network Security Information Sub-Model Overview

In the above figure, the ECAPolicyRule, along with the Event, Condition, and Action Objects, are defined in the external ECA Information Model. The Network Security Sub-Model extends all of these objects in order to define security-specific ECA Policy Rules, as well as extensions to the (generic) Event, Condition, and Action objects.

An I2NSF Policy Rule is a special type of Policy Rule that is in event-condition-action (ECA) form. It consists of the Policy Rule, components of a Policy Rule (e.g., events, conditions, actions, and some extensions like resolution policy, default action and external data), and optionally, metadata. It can be applied to both uni- and bi-directional traffic across the NSF.

Each rule is triggered by one or more events. If the set of events evaluates to true, then a set of conditions are evaluated and, if true, enable a set of actions to be executed. This takes the following conceptual form:

```
IF <event-clause> is TRUE
   IF <condition-clause> is TRUE
      THEN execute <action-clause>
   END-IF
END-IF
```

In the above example, the Event, Condition, and Action portions of a Policy Rule are all **Boolean Clauses**. Hence, they can contain combinations of terms connected by the three logical connectives operators (i.e., AND, OR, NOT). An example is:

```
((SLA==GOLD) AND ((numPackets>burstRate) OR NOT(bwAvail<minBW)))
```

Note that Metadata, such as Capabilities, can be aggregated by I2NSF ECA Policy Rules.

4.1.1. Network Security Policy Rule Extensions

Figure 3 shows an example of more detailed design of the ECA Policy Rule subclasses that are contained in the Network Security Information Sub-Model, which just illustrates how more specific Network Security Policies are inherited and extended from the SecurityECAPolicyRule class. Any new kinds of specific Network Security Policy can be created by following the same pattern of class design as below.

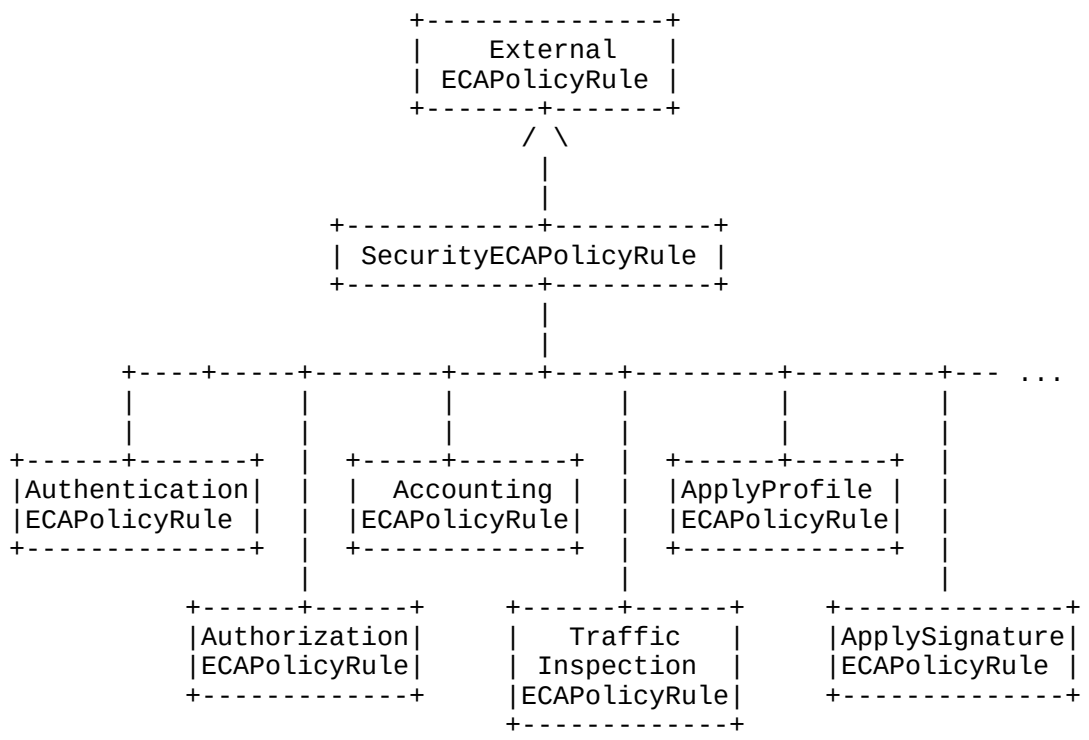


Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

The SecurityECAPolicyRule is the top of the I2NSF ECA Policy Rule hierarchy. It inherits from the (external) generic ECA Policy Rule, and represents the specialization of this generic ECA Policy Rule to add security-specific ECA Policy Rules. The SecurityECAPolicyRule contains all of the attributes, methods, and relationships defined in its superclass, and adds additional concepts that are required for Network Security (these will be defined in the next version of this draft). The six SecurityECAPolicyRule subclasses extend the SecurityECAPolicyRule class to represent six different types of Network Security ECA Policy Rules. It is assumed that the (external) generic ECAPolicyRule class defines basic information in the form of attributes, such as an unique object ID, as well as a description and other necessary information.

*** Note to WG
*
* The design in Figure 3 represents the simplest conceptual design
* for network security. An alternative model would be to use a
* software pattern (e.g., the Decorator pattern); this would result
* in the SecurityECAPolicyRule class being "wrapped" by one or more
* of the six subclasses shown in Figure 3. The advantage of such a
* pattern is to reduce the number of active objects at runtime, as
* well as offer the ability to combine multiple actions of different
* policy rules (e.g., inspect traffic and then apply a filter) into
* one. The disadvantage is that it is a more complex software design.
* The design team is requesting feedback from the WG regarding this.
*
*** End of Note to WG

It is assumed that the (external) generic ECA Policy Rule is abstract; the SecurityECAPolicyRule is also abstract. This enables data model optimizations to be made while making this information model detailed but flexible and extensible. For example, abstract classes may be collapsed into concrete classes.

The SecurityECAPolicyRule defines network security policy as a container that aggregates Event, Condition, and Action objects, which are described in Section 4.1.3, 4.1.4, and 4.1.5, respectively. Events, Conditions, and Actions can be generic or security-specific.

Brief class descriptions of these six ECA Policy Rules are provided in Appendix A.

4.1.2. Network Security Policy Rule Operation

A Network Security Policy consists of one or more ECA Policy Rules formed from the information model described above. In simpler cases, where the Event and Condition clauses remain unchanged, then the action of one Policy Rule may invoke additional network security actions from other Policy Rules. Network security policy examines and performs basic processing of the traffic as follows:

1. The NSF evaluates the Event clause of a given SecurityECAPolicyRule (which can be generic or specific to security, such as those in Figure 3). It may use security Event objects to do all or part of this evaluation, which are defined in section 4.1.3. If the Event clause evaluates to TRUE, then the Condition clause of this SecurityECAPolicyRule is evaluated; otherwise, the execution of this SecurityECAPolicyRule is stopped, and the next SecurityECAPolicyRule (if one exists) is evaluated.

2. The Condition clause is then evaluated. It may use security Condition objects to do all or part of this evaluation, which are defined in section 4.1.4. If the Condition clause evaluates to TRUE, it is defined as "matching" the SecurityECAPolicyRule; otherwise, execution of this SecurityECAPolicyRule is stopped, and the next SecurityECAPolicyRule (if one exists) is evaluated.
3. The set of actions to be executed are retrieved, and then the resolution strategy is used to define their execution order. This process includes using any optional external data associated with the SecurityECAPolicyRule.
4. Execution then takes one of the following three forms:
   a. If one or more actions is selected, then the NSF may perform those actions as defined by the resolution strategy. For example, the resolution strategy may only allow a single action to be executed (e.g., FMR or LMR), or it may allow all actions to be executed (optionally, in a particular order). In these and other cases, the NSF Capability MUST clearly define how execution will be done. It may use security Action objects to do all or part of this execution, which are defined in section 4.1.5. If the basic Action is permit or mirror, the NSF firstly performs that function, and then checks whether certain other security Capabilities are referenced in the rule. If yes, go to step 5. If no, the traffic is permitted.
   b. If no actions are selected, and if a default action exists, then the default action is performed. Otherwise, no actions are performed.
   c. Otherwise, the traffic is denied.
5. If other security Capabilities (e.g., the conditions and/or actions implied by Anti-virus or IPS profile NSFs) are referenced in the action set of the SecurityECAPolicyRule, the NSF can be configured to use the referenced security Capabilities (e.g., check conditions or enforce actions). Execution then terminates.

One policy or rule can be applied multiple times to different managed objects (e.g., links, devices, networks, VPNS). This not only guarantees consistent policy enforcement, but also decreases the configuration workload.

4.1.3. Network Security Event Sub-Model

Figure 4 shows a more detailed design of the Event subclasses that are contained in the Network Security Information Sub-Model.

The four Event classes shown in Figure 4 extend the (external) generic Event class to represent Events that are of interest to Network Security. It is assumed that the (external) generic Event class defines basic Event information in the form of attributes, such as a unique event ID, a description, as well as the date and time that the event occurred.

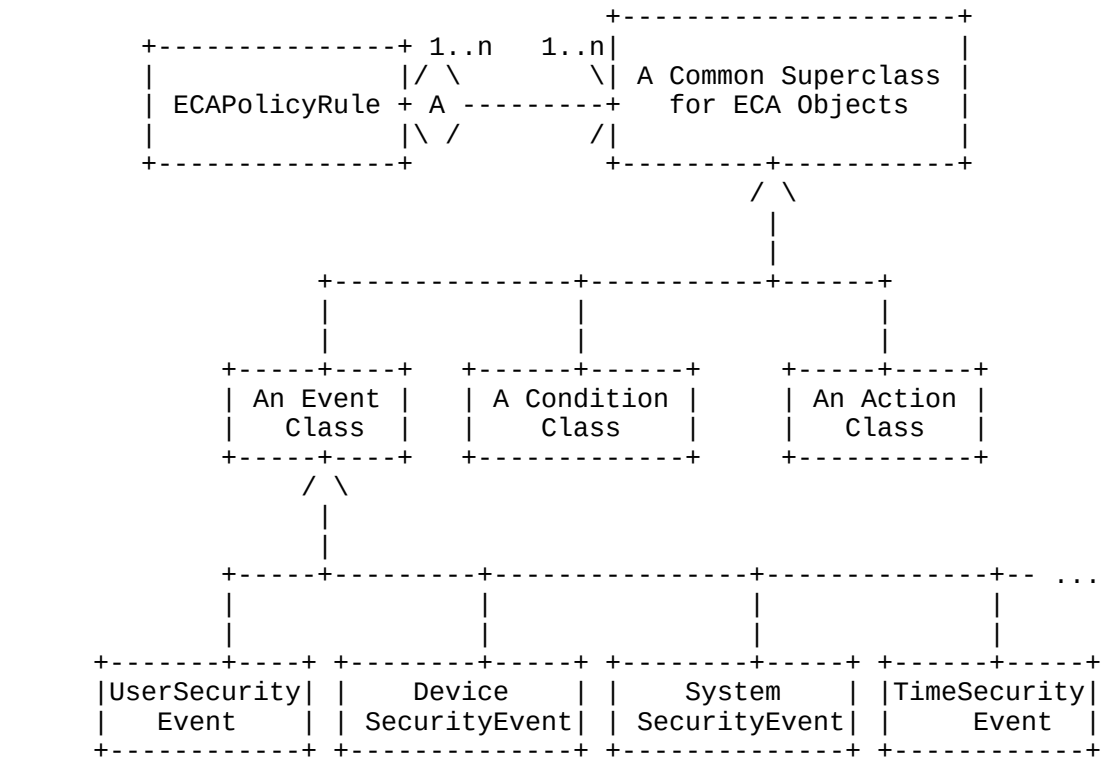


Figure 4. Network Security Info Sub-Model Event Class Extensions

The following are assumptions that define the functionality of the generic Event class. If desired, these could be defined as attributes in a SecurityEvent class (which would be a subclass of the generic Event class, and a superclass of the four Event classes shown in Figure 4). However, this makes it harder to use any generic Event model with the I2NSF events. Assumptions are:

- All four SecurityEvent subclasses are concrete
- The generic Event class uses the composite pattern, so individual Events as well as hierarchies of Events are available (the four subclasses in Figure 4 would be subclasses of the Atomic Event class); otherwise, a mechanism is needed to be able to group Events into a collection
- The generic Event class has a mechanism to uniquely identify the source of the Event
- The generic Event class has a mechanism to separate header information from its payload
- The generic Event class has a mechanism to attach zero or more metadata objects to it

*** Note to WG:
*
* The design in Figure 4 represents the simplest conceptual design
* design for describing Security Events. An alternative model would
* be to use a software pattern (e.g., the Decorator pattern); this
* would result in the SecurityEvent class being "wrapped" by one or
* more of the four subclasses shown in Figure 4. The advantage of
* such a pattern is to reduce the number of active objects at runtime,
* as well as offer the ability to combine multiple events of different
* types into one. The disadvantage is that it is a more complex
* software design.
*
*** End of Note to WG

Brief class descriptions are provided in Appendix B.

4.1.4. Network Security Condition Sub-Model

Figure 5 shows a more detailed design of the Condition subclasses that are contained in the Network Security Information Sub-Model. The six Condition classes shown in Figure 5 extend the (external) generic Condition class to represent Conditions that are of interest to Network Security. It is assumed that the (external) generic Condition class is abstract, so that data model optimizations may be defined. It is also assumed that the generic Condition class defines basic Condition information in the form of attributes, such as a unique object ID, a description, as well as a mechanism to attach zero or more metadata objects to it. While this could be defined as attributes in a SecurityCondition class (which would be a subclass of the generic Condition class, and a superclass of the six Condition classes shown in Figure 5), this makes it harder to use any generic Condition model with the I2NSF conditions.

*** Note to WG:
*
* The design in Figure 5 represents the simplest conceptual design
* for describing Security Conditions. An alternative model would be
* to use a software pattern (e.g., the Decorator pattern); this would
* result in the SecurityCondition class being "wrapped" by one or
* more of the six subclasses shown in Figure 5. The advantage of such
* a pattern is to reduce the number of active objects at runtime, as
* well as offer the ability to combine multiple conditions of
* different types into one. The disadvantage is that it is a more
* complex software design.
* The design team is requesting feedback from he WG regarding this.
*
*** End of Note to WG

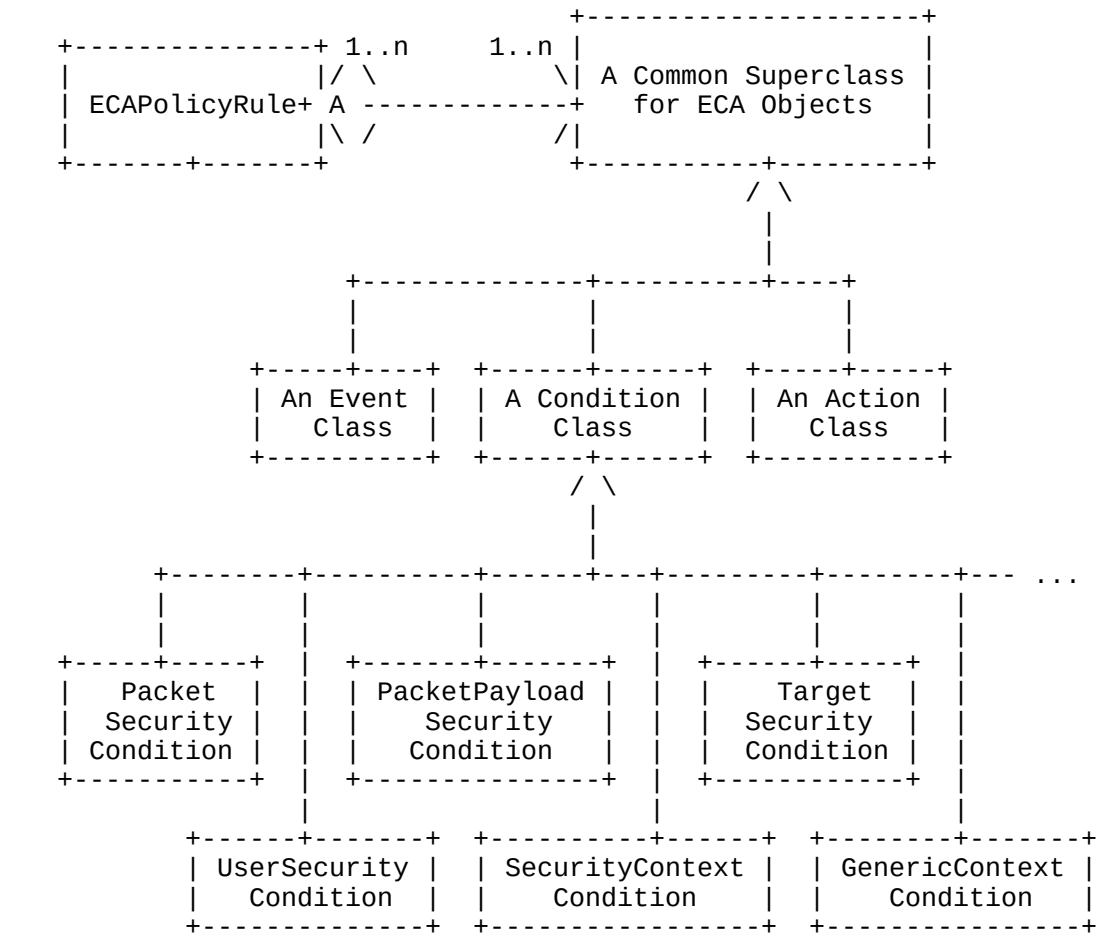


Figure 5. Network Security Info Sub-Model Condition Class Extensions

Brief class descriptions are provided in Appendix C.

4.1.5. Network Security Action Sub-Model

Figure 6 shows a more detailed design of the Action subclasses that are contained in the Network Security Information Sub-Model.

The four Action classes shown in Figure 6 extend the (external) generic Action class to represent Actions that perform a Network Security Control function.

The three Action classes shown in Figure 6 extend the (external) generic Action class to represent Actions that are of interest to Network Security. It is assumed that the (external) generic Action class is abstract, so that data model optimizations may be defined.

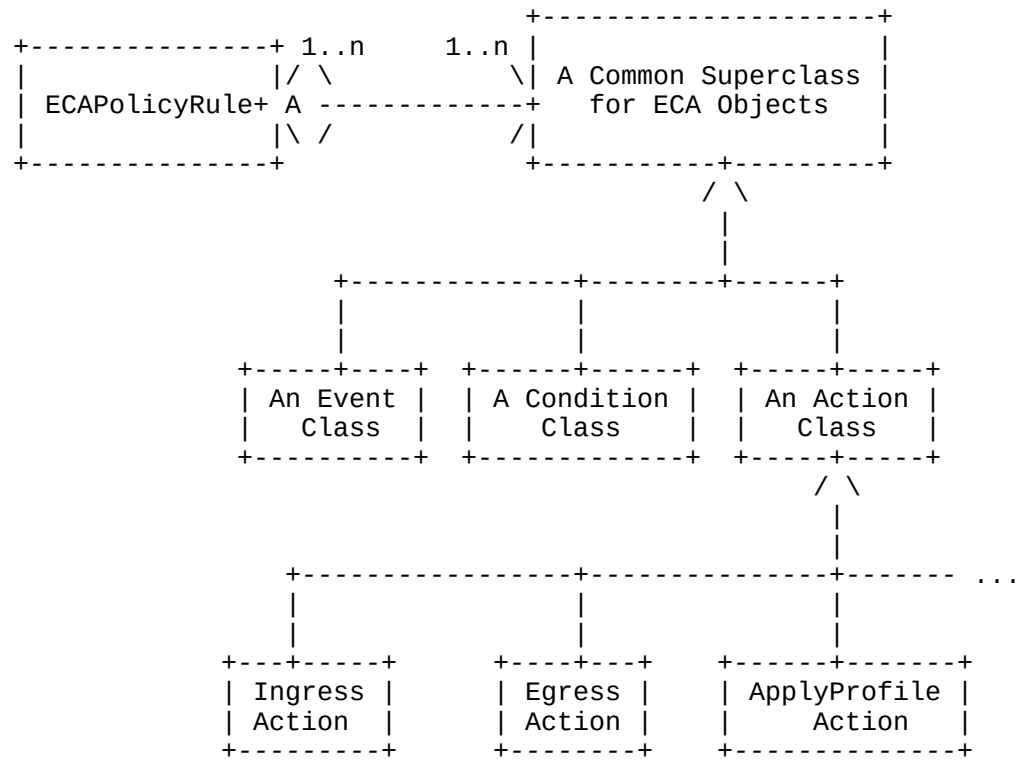


Figure 6. Network Security Info Sub-Model Action Extensions

It is also assumed that the generic Action class defines basic Action information in the form of attributes, such as a unique object ID, a description, as well as a mechanism to attach zero or more metadata objects to it. While this could be defined as attributes in a SecurityAction class (which would be a subclass of the generic Action class, and a superclass of the six Action classes shown in Figure 6), this makes it harder to use any generic Action model with the I2NSF actions.

*** Note to WG
* The design in Figure 6 represents the simplest conceptual design
* for describing Security Actions. An alternative model would be to
* use a software pattern (e.g., the Decorator pattern); this would
* result in the SecurityAction class being "wrapped" by one or more
* of the three subclasses shown in Figure 6. The advantage of such a
* pattern is to reduce the number of active objects at runtime, as
* well as offer the ability to combine multiple actions of different
* types into one. The disadvantage is that it is a more complex
* software design.
* The design team is requesting feedback from the WG regarding this.
*** End of Note to WG

Brief class descriptions are provided in Appendix D.

4.2. Information Model for I2NSF Capabilities

The I2NSF Capability Model is made up of a number of Capabilities that represent various content security and attack mitigation functions. Each Capability protects against a specific type of threat in the application layer. This is shown in Figure 7.

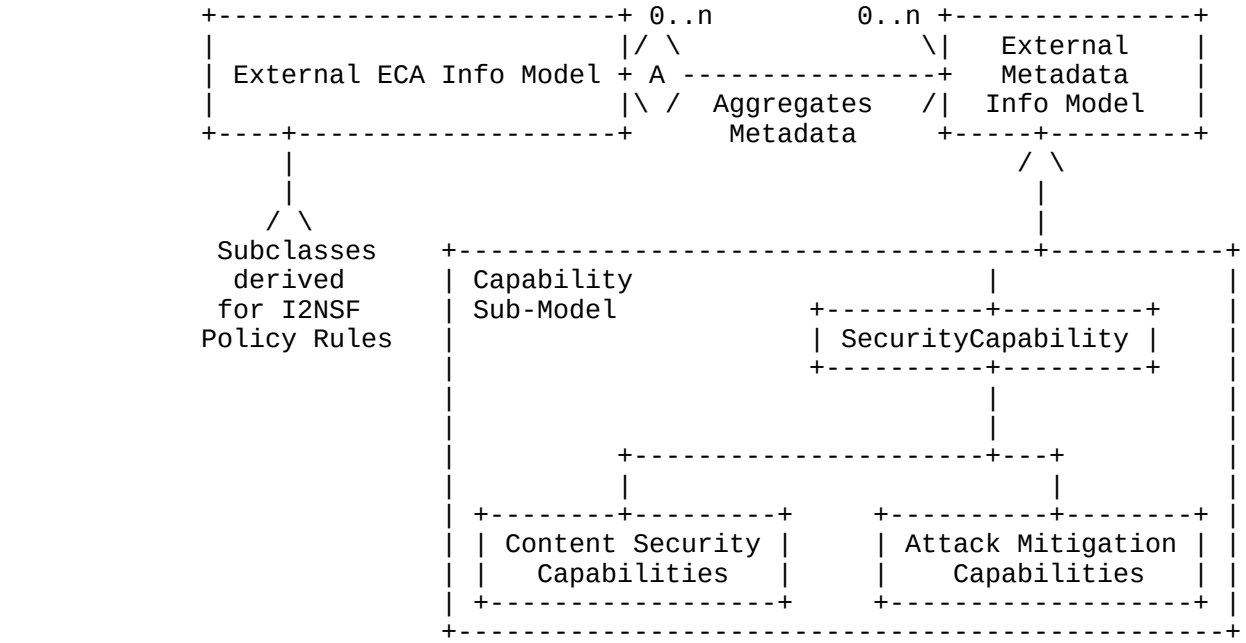


Figure 7. I2NSF Security Capability High-Level Model

Figure 7 shows a common I2NSF Security Capability class, called SecurityCapability. This enables us to add common attributes, relationships, and behavior to this class without affecting the design of the external metadata information model. All I2NSF Security Capabilities are then subclassed from the SecuritCapability class.

Note: the SecurityCapability class will be defined in the next version of this draft, after feedback from the WG is obtained.

4.3. Information Model for Content Security Capabilities

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, and was shown in Figure 7.

Figure 8 shows exemplary types of the content security GNSF.

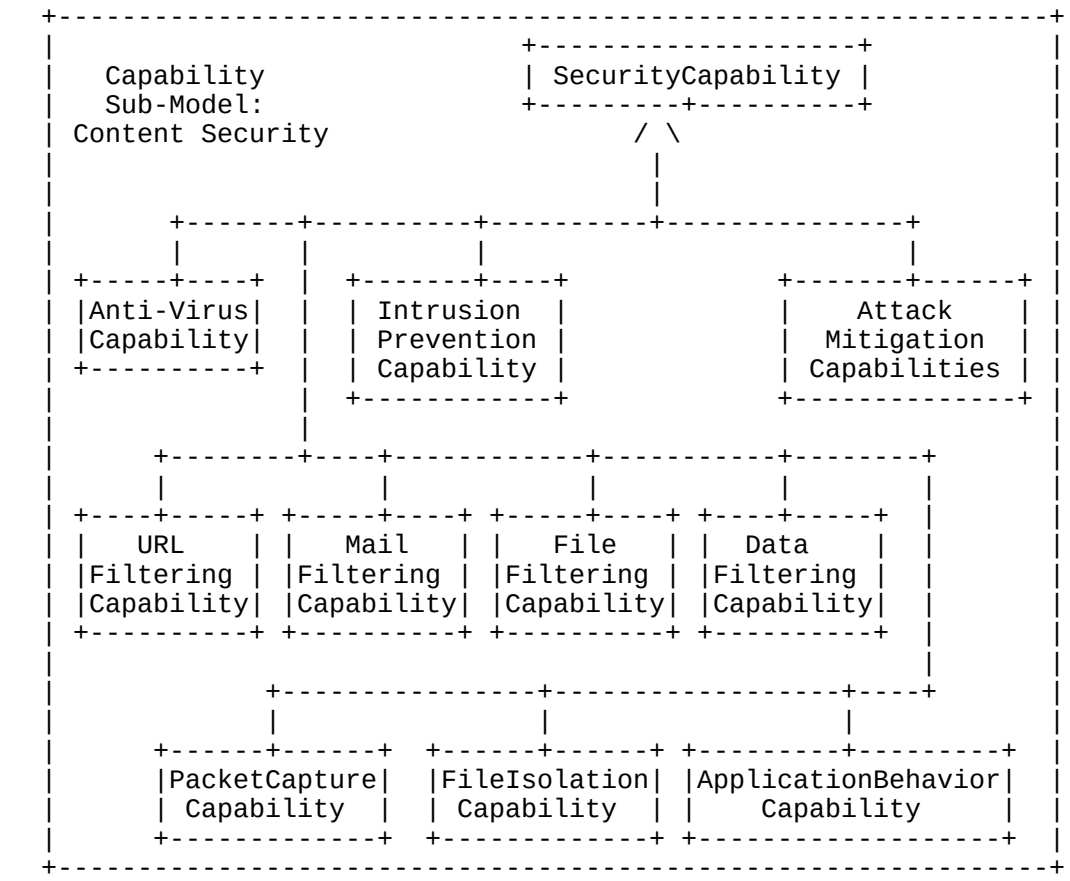


Figure 8. Network Security Capability Information Model

The detailed description about a standard interface, and the parameters for all the security Capabilities of this category, will be defined in a future version of this document.

4.4. Information Model for Attack Mitigation Capabilities

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, and was shown in Figure 7. Figure 9 shows exemplary types of Attack Mitigation GNSFs.

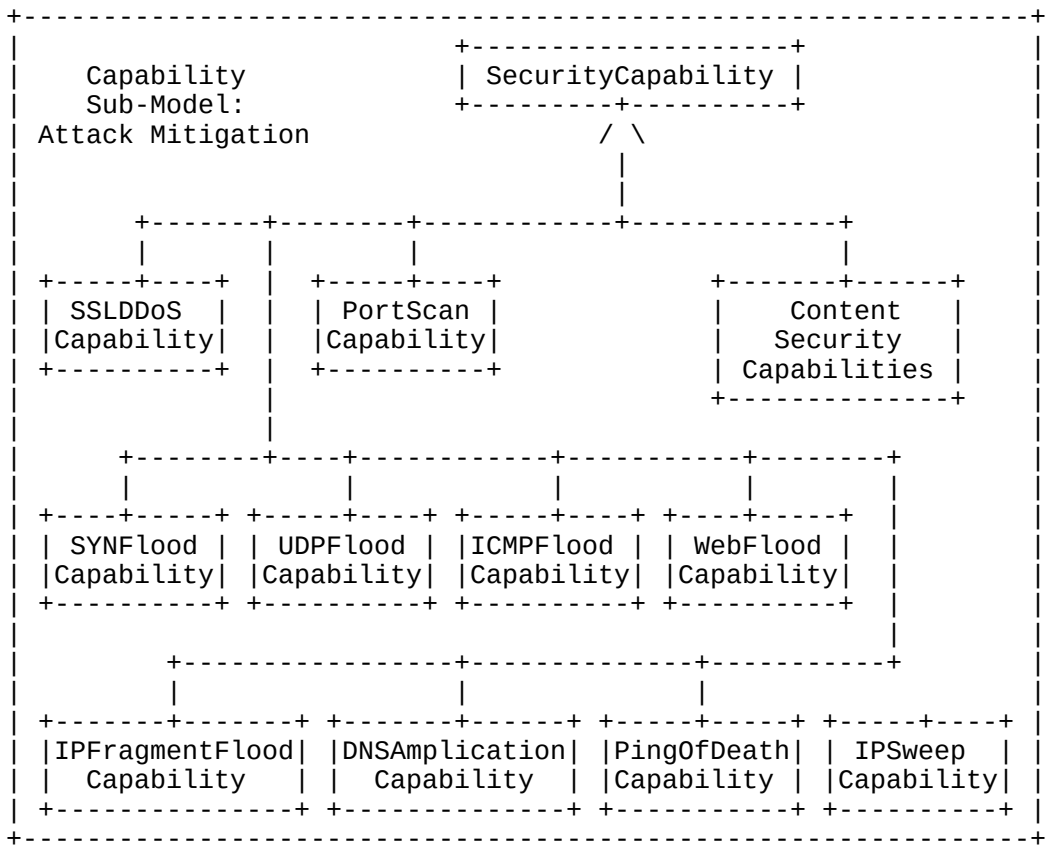


Figure 9. Attack Mitigation Capability Information Model

The detailed description about a standard interface, and the parameters for all the security Capabilities of this category, will be defined in a future version of this document.

5. Security Considerations

The security Capability policy information sent to NSFs should be protected by a secure communication channel, to ensure its confidentiality and integrity. Note that the NSFs and security controller can all be spoofed, which leads to undesirable results (e.g., security policy leakage from security controller, or a spoofed security controller sending false information to mislead the NSFs). Hence, mutual authentication MUST be supported to protected against this kind of threat. The current mainstream security technologies (i.e., TLS, DTLS, and IPSEC) can be employed to protect against the above threats.

In addition, to defend against DDoS attacks caused by a hostile security controller sending too many configuration messages to the NSFs, rate limiting or similar mechanisms should be considered.

6. IANA Considerations

TBD

7. Contributors

The following people contributed to creating this document, and are listed below in alphabetical order:

Antonio Lioy (Politecnico di Torino)
Dacheng Zhang (Huawei)
Edward Lopez (Fortinet)
Fulvio Valenza (Politecnico di Torino)
Kepeng Li (Alibaba)
Luyuan Fang (Microsoft)
Nicolas Bouthors (QoSmos)

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.

[RFC3539]
Aboba, B., and Wood, J., "Authentication, Authorization, and Accounting (AAA) Transport Profile", RFC 3539, June 2003.

8.2. Informative References

[RFC2975]
Aboba, B., et al., "Introduction to Accounting Management", RFC 2975, October 2000.

[I-D.draft-ietf-i2nsf-problem-and-use-cases]
Hares, S., et.al., "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-16, May 2017.

[I-D.draft-ietf-i2nsf-framework]
Lopez, E., et.al., "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-06, July, 2017.

[I-D.draft-ietf-i2nsf-terminology]
Hares, S., et.al., "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03, March, 2017

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

[Alshaer]
Al Shaer, E. and H. Hamed, "Modeling and management of firewall policies", 2004.

[Bas12]
Basile, C., Cappadonia, A., and A. Lioy, "Network-Level Access Control Policy Analysis and Transformation", 2012.

[Bas15]
Basile, C. and Lioy, A., "Analysis of application-layer filtering policies with application to HTTP", IEEE/ACM Transactions on Networking, Vol 23, Issue 1, February 2015.

[Cormen]
Cormen, T., "Introduction to Algorithms", 2009.

[Hohpe]
Hohpe, G. and Woolf, B., "Enterprise Integration Patterns", Addison-Wesley, 2003, ISBN 0-32-120068-3

[Lunt]
van Lunteren, J. and T. Engbersen, "Fast and scalable packet classification", IEEE Journal on Selected Areas in Communication, vol 21, Issue 4, September 2003.

[Martin]
Martin, R.C., "Agile Software Development, Principles, Patterns, and Practices", Prentice-Hall, 2002, ISBN: 0-13-597444-5

[OODMP]
http://www.oodesign.com/mediator-pattern.html

[OODOP]
http://www.oodesign.com/observer-pattern.html

[OODSRP]
http://www.oodesign.com/single-responsibility-principle.html

Appendix A. Network Security Capability Policy Rule Definitions

Six exemplary Network Security Capability Policy Rules are introduced in this Appendix to clarify how to create different kinds of specific ECA policy rules to manage Network Security Capabilities.

Note that there is a common pattern that defines how these ECAPolicyRules operate; this simplifies their implementation. All of these six ECA Policy Rules are concrete classes.

In addition, none of these six subclasses define attributes. This enables them to be viewed as simple object containers, and hence, applicable to a wide variety of content. It also means that the content of the function (e.g., how an entity is authenticated, what specific traffic is inspected, or which particular signature is applied) is defined solely by the set of events, conditions, and actions that are contained by the particular subclass. This enables the policy rule, with its aggregated set of events, conditions, and actions, to be treated as a reusable object.

A.1. AuthenticationECAPolicyRule Class Definition

The purpose of an AuthenticationECAPolicyRule is to define an I2NSF ECA Policy Rule that can verify whether an entity has an attribute of a specific value. A high-level conceptual figure is shown below.

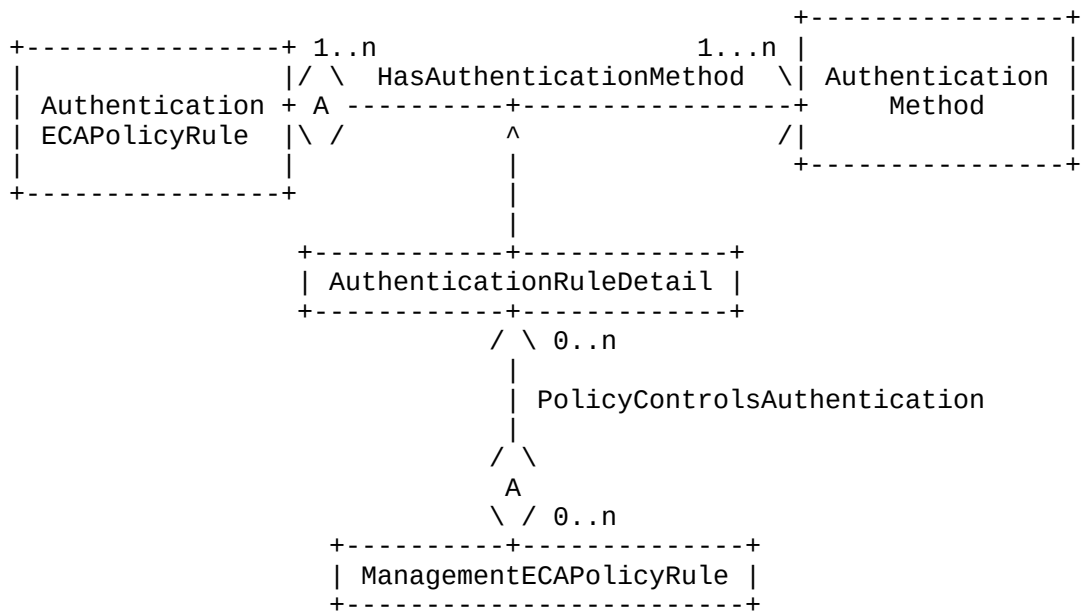


Figure 10. Modeling Authentication Mechanisms

This class does NOT define the authentication method used. This is because this would effectively "enclose" this information within the AuthenticationECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Authentication class(es) could not; they would have to associate with the AuthenticationECAPolicyRule class, and those other classes would not likely be interested in the AuthenticationECAPolicyRule. Second, the evolution of new authentication methods should be independent of the AuthenticationECAPolicyRule; this cannot happen if the Authentication class(es) are embedded in the AuthenticationECAPolicyRule.

This document only defines the AuthenticationECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 10 defines an aggregation between an external class, which defines one or more authentication methods, and an AuthenticationECAPolicyRule. This decouples the implementation of authentication mechanisms from how authentication mechanisms are managed and used.

Since different AuthenticationECAPolicyRules can use different authentication mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AuthenticationRuleDetail) to be used to define how a given AuthenticationMethod is used by a particular AuthenticationECAPolicyRule.

Similarly, the PolicyControlsAuthentication aggregation defines Policy Rules to control the configuration of the AuthenticationRuleDetail association class. This enables the entire authentication process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AuthenticationECAPolicyRule class (e.g., called authenticationMethodCurrent and authenticationMethodSupported), to represent the HasAuthenticationMethod aggregation and its association class. The former would be a string attribute that defines the current authentication method used by this AuthenticationECAPolicyRule, while the latter would define a set of authentication methods, in the form of an authentication Capability, which this AuthenticationECAPolicyRule can advertise.

A.2. AuthorizationECAPolicyRuleClass Definition

The purpose of an AuthorizationECAPolicyRule is to define an I2NSF ECA Policy Rule that can determine whether access to a resource should be given and, if so, what permissions should be granted to the entity that is accessing the resource.

This class does NOT define the authorization method(s) used. This is because this would effectively "enclose" this information within the AuthorizationECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Authorization class(es) could not; they would have to associate with the AuthorizationECAPolicyRule class, and those other classes would not likely be interested in the AuthorizationECAPolicyRule. Second, the evolution of new authorization methods should be independent of the AuthorizationECAPolicyRule; this cannot happen if the Authorization class(es) are embedded in the AuthorizationECAPolicyRule. Hence, this document recommends the following design:

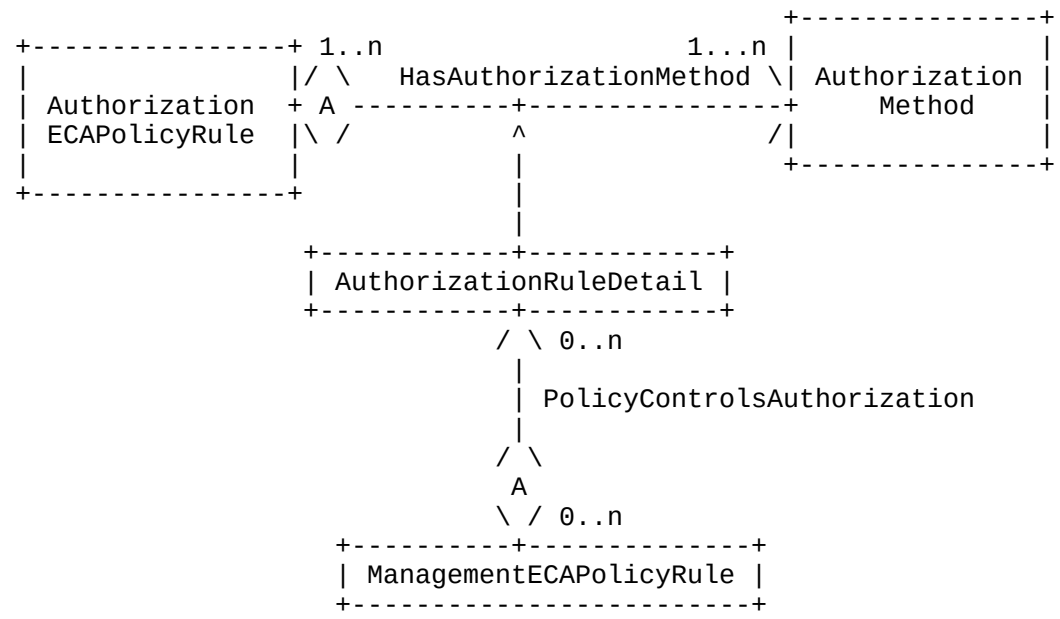


Figure 11. Modeling Authorization Mechanisms

This document only defines the AuthorizationECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 11 defines an aggregation between the AuthorizationECAPolicyRule and an external class that defines one or more authorization methods. This decouples the implementation of authorization mechanisms from how authorization mechanisms are managed and used.

Since different AuthorizationECAPolicyRules can use different authorization mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AuthorizationRuleDetail) to be used to define how a given AuthorizationMethod is used by a particular AuthorizationECAPolicyRule.

Similarly, the PolicyControlsAuthorization aggregation defines Policy Rules to control the configuration of the AuthorizationRuleDetail association class. This enables the entire authorization process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AuthorizationECAPolicyRule class, called (for example) authorizationMethodCurrent and authorizationMethodSupported, to represent the HasAuthorizationMethod aggregation and its association class. The former is a string attribute that defines the current authorization method used by this AuthorizationECAPolicyRule, while the latter defines a set of authorization methods, in the form of an authorization Capability, which this AuthorizationECAPolicyRule can advertise.

A.3. AccountingECAPolicyRuleClass Definition

The purpose of an AccountingECAPolicyRule is to define an I2NSF ECA Policy Rule that can determine which information to collect, and how to collect that information, from which set of resources for the purpose of trend analysis, auditing, billing, or cost allocation [RFC2975] [RFC3539].

This class does NOT define the accounting method(s) used. This is because this would effectively "enclose" this information within the AccountingECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Accounting class(es) could not; they would have to associate with the AccountingECAPolicyRule class, and those other classes would not likely be interested in the AccountingECAPolicyRule. Second, the evolution of new accounting methods should be independent of the AccountingECAPolicyRule; this cannot happen if the Accounting class(es) are embedded in the AccountingECAPolicyRule. Hence, this document recommends the following design:

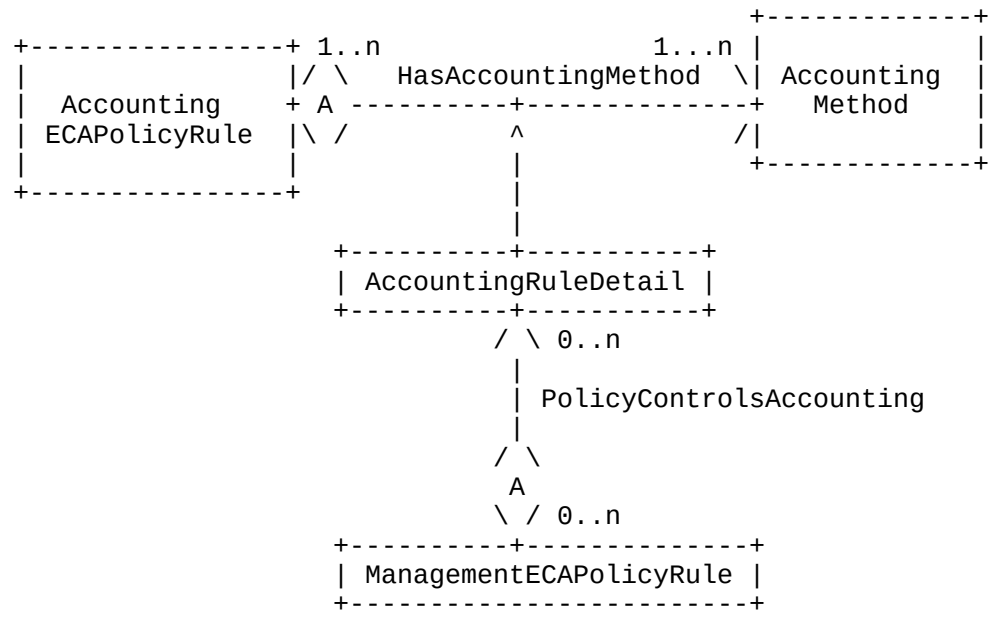


Figure 12. Modeling Accounting Mechanisms

This document only defines the AccountingECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 12 defines an aggregation between the AccountingECAPolicyRule and an external class that defines one or more accounting methods. This decouples the implementation of accounting mechanisms from how accounting mechanisms are managed and used.

Since different AccountingECAPolicyRules can use different accounting mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., AccountingRuleDetail) to be used to define how a given AccountingMethod is used by a particular AccountingECAPolicyRule.

Similarly, the PolicyControlsAccounting aggregation defines Policy Rules to control the configuration of the AccountingRuleDetail association class. This enables the entire accounting process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the AccountingECAPolicyRule class, called (for example) accountingMethodCurrent and accountingMethodSupported, to represent the HasAccountingMethod aggregation and its association class.

The former is a string attribute that defines the current accounting method used by this AccountingECAPolicyRule, while the latter defines a set of accounting methods, in the form of an accounting Capability, which this AccountingECAPolicyRule can advertise.

A.4. TrafficInspectionECAPolicyRuleClass Definition

The purpose of a TrafficInspectionECAPolicyRule is to define an I2NSF ECA Policy Rule that, based on a given context, can determine which traffic to examine on which devices, which information to collect from those devices, and how to collect that information.

This class does NOT define the traffic inspection method(s) used. This is because this would effectively "enclose" this information within the TrafficInspectionECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the TrafficInspection class(es) could not; they would have to associate with the TrafficInspectionECAPolicyRule class, and those other classes would not likely be interested in the TrafficInspectionECAPolicyRule. Second, the evolution of new traffic inspection methods should be independent of the TrafficInspectionECAPolicyRule; this cannot happen if the TrafficInspection class(es) are embedded in the TrafficInspectionECAPolicyRule. Hence, this document recommends the following design:

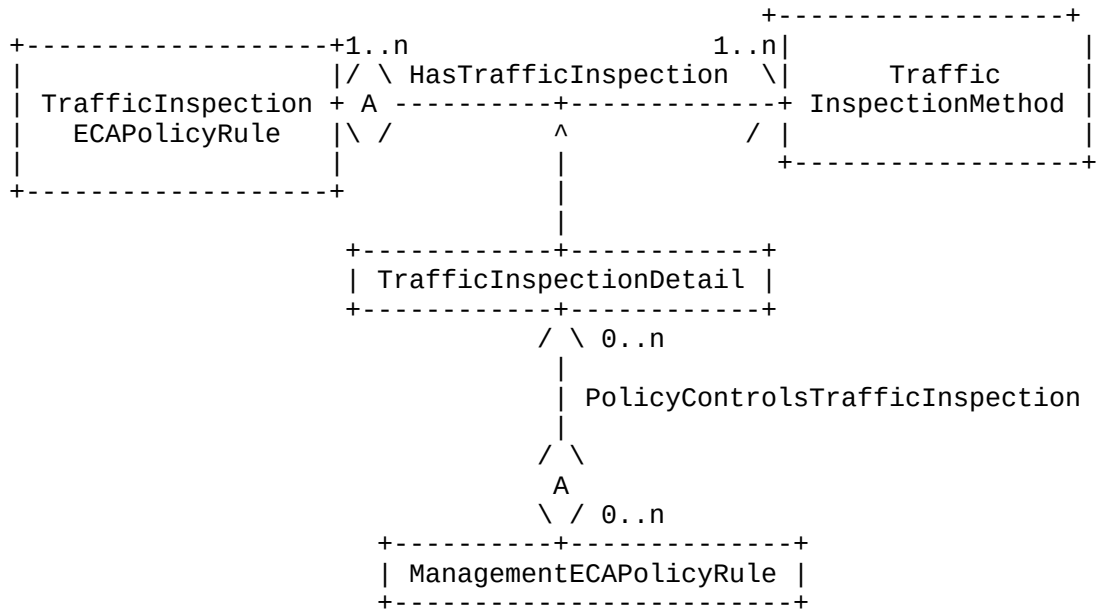


Figure 13. Modeling Traffic Inspection Mechanisms

This document only defines the TrafficInspectionECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 13 defines an aggregation between the TrafficInspectionECAPolicyRule and an external class that defines one or more traffic inspection mechanisms. This decouples the implementation of traffic inspection mechanisms from how traffic inspection mechanisms are managed and used.

Since different TrafficInspectionECAPolicyRules can use different traffic inspection mechanisms in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., TrafficInspectionDetail) to be used to define how a given TrafficInspectionMethod is used by a particular TrafficInspectionECAPolicyRule.

Similarly, the PolicyControlsTrafficInspection aggregation defines Policy Rules to control the configuration of the TrafficInspectionDetail association class. This enables the entire traffic inspection process to be managed by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the TrafficInspectionECAPolicyRule class, called (for example) trafficInspectionMethodCurrent and trafficInspectionMethodSupported, to represent the HasTrafficInspectionMethod aggregation and its association class. The former is a string attribute that defines the current traffic inspection method used by this TrafficInspectionECAPolicyRule, while the latter defines a set of traffic inspection methods, in the form of a traffic inspection Capability, which this TrafficInspectionECAPolicyRule can advertise.

A.5. ApplyProfileECAPolicyRuleClass Definition

The purpose of an ApplyProfileECAPolicyRule is to define an I2NSF ECA Policy Rule that, based on a given context, can apply a particular profile to specific traffic. The profile defines the security Capabilities for content security control and/or attack mitigation control; these will be described in sections 4.4 and 4.5, respectively.

This class does NOT define the set of Profiles used. This is because this would effectively "enclose" this information within the ApplyProfileECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Profile class(es) could not; they would have to associate with the

ApplyProfileECAPolicyRule class, and those other classes would not likely be interested in the ApplyProfileECAPolicyRule. Second, the evolution of new Profile classes should be independent of the ApplyProfileECAPolicyRule; this cannot happen if the Profile class(es) are embedded in the ApplyProfileECAPolicyRule. Hence, this document recommends the following design:

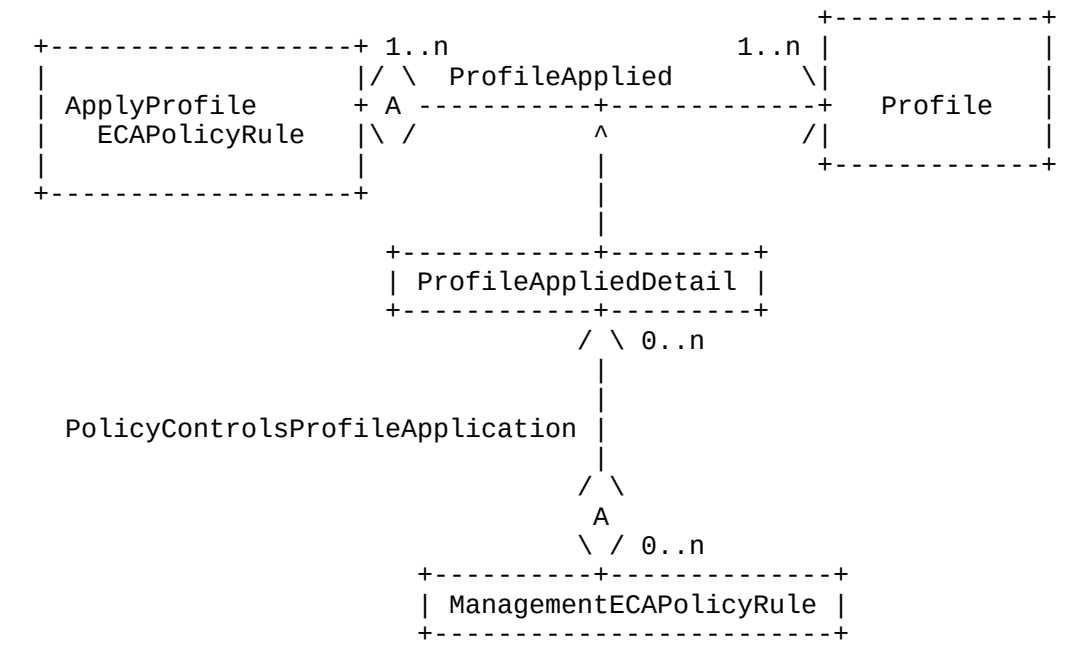


Figure 14. Modeling Profile ApplicationMechanisms

This document only defines the ApplyProfileECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 14 defines an aggregation between the ApplyProfileECAPolicyRule and an external Profile class. This decouples the implementation of Profiles from how Profiles are used.

Since different ApplyProfileECAPolicyRules can use different Profiles in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., ProfileAppliedDetail) to be used to define how a given Profile is used by a particular ApplyProfileECAPolicyRule.

Similarly, the PolicyControlsProfileApplication aggregation defines policies to control the configuration of the ProfileAppliedDetail association class. This enables the application of Profiles to be managed and used by ECA PolicyRules.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the ApplyProfileECAPolicyRuleclass, called (for example) profileAppliedCurrent and profileAppliedSupported, to represent the ProfileApplied aggregation and its association class. The former is a string attribute that defines the current Profile used by this ApplyProfileECAPolicyRule, while the latter defines a set of Profiles, in the form of a Profile Capability, which this ApplyProfileECAPolicyRule can advertise.

A.6. ApplySignatureECAPolicyRuleClass Definition

The purpose of an ApplySignatureECAPolicyRule is to define an I2NSF ECA Policy Rule that, based on a given context, can determine which Signature object (e.g., an anti-virus file, or aURL filtering file, or a script) to apply to which traffic. The Signature object defines the security Capabilities for content security control and/or attack mitigation control; these will be described in sections 4.4 and 4.5, respectively.

This class does NOT define the set of Signature objects used. This is because this would effectively "enclose" this information within the ApplySignatureECAPolicyRule. This has two drawbacks. First, other entities that need to use information from the Signature object class(es) could not; they would have to associate with the ApplySignatureECAPolicyRule class, and those other classes would not likely be interested in the ApplySignatureECAPolicyRule. Second, the evolution of new Signature object classes should be independent of the ApplySignatureECAPolicyRule; this cannot happen if the Signature object class(es) are embedded in the ApplySignatureECAPolicyRule. Hence, this document recommends the following design:

This document only defines the ApplySignatureECAPolicyRule; all other classes, and the aggregations, are defined in an external model. For completeness, descriptions of how the two aggregations are used are described below.

Figure 15 defines an aggregation between the ApplySignatureECAPolicyRule and an external Signature object class. This decouples the implementation of signature objects from how Signature objects are used.

Since different ApplySignatureECAPolicyRules can use different Signature objects in different ways, the aggregation is realized as an association class. This enables the attributes and methods of the association class (i.e., SignatureAppliedDetail) to be used to define how a given Signature object is used by a particular ApplySignatureECAPolicyRule.

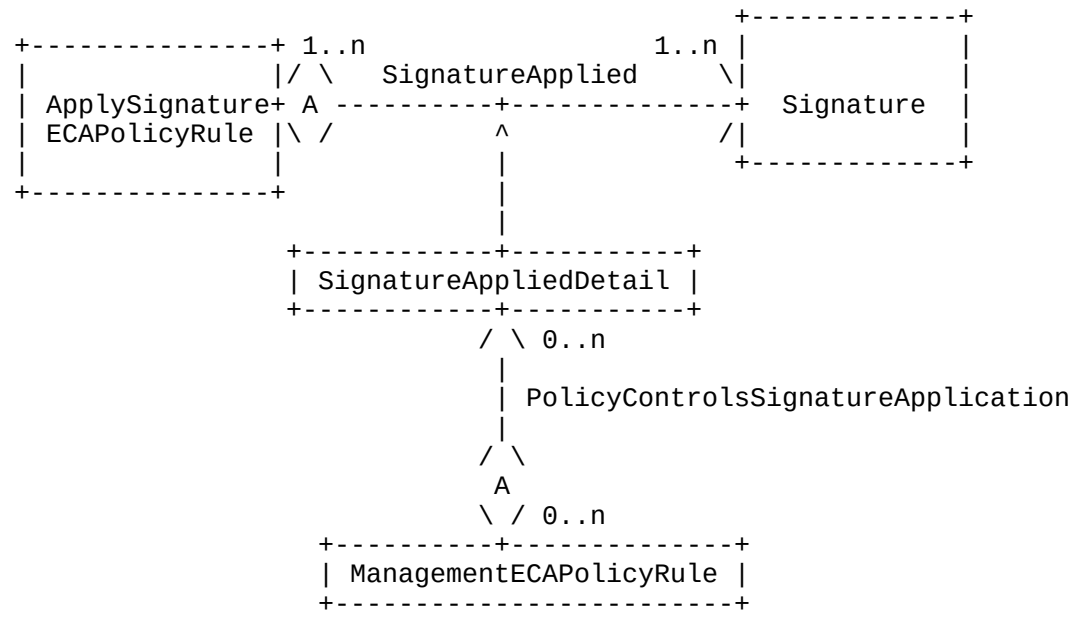


Figure 15. Modeling Sginature Application Mechanisms

Similarly, the PolicyControlsSignatureApplication aggregation defines policies to control the configuration of the SignatureAppliedDetail association class. This enables the application of the Signature object to be managed by policy.

Note: a data model MAY choose to collapse this design into a more efficient implementation. For example, a data model could define two attributes for the ApplySignatureECAPolicyRule class, called (for example) signature signatureAppliedCurrent and signatureAppliedSupported, to represent the SignatureApplied aggregation and its association class. The former is a string attribute that defines the current Signature object used by this ApplySignatureECAPolicyRule, while the latter defines a set of Signature objects, in the form of a Signature Capability, which this ApplySignatureECAPolicyRule can advertise.

Appendix B. Network Security Event Class Definitions

This Appendix defines a preliminary set of Network Security Event classes, along with their attributes.

B.1. UserSecurityEvent Class Description

The purpose of this class is to represent Events that are initiated by a user, such as logon and logoff Events. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include user identification data and the type of connection used by the user.

The UserSecurityEvent class defines the following attributes.

B.1.1. The usrSecEventContent Attribute

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 the string "hrAdmin", with the usrSecEventFormat set to 1 (GUID) and the usrSecEventType attribute set to 5 (new logon).

B.1.2. The usrSecEventFormat Attribute

This is a mandatory non-negative 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 the string "hrAdmin", with the usrSecEventFormat attribute set to 1 (GUID) and the usrSecEventType attribute set to 5 (new logon). Values include:

0: unknown
1: GUID (Generic Unique IDentifier)
2: UUID (Universal Unique IDentifier)
3: URI (Uniform Resource Identifier)
4: FQDN (Fully Qualified Domain Name)
5: FQPN (Fully Qualified Path Name)

B.1.3. The usrSecEventType Attribute

This is a mandatory non-negative 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 the string "hrAdmin", with the usrSecEventFormat attribute set to 1 (GUID), and the usrSecEventType attribute set to 5 (new logon). Values include:

0: unknown
1: new user created
2: new user group created
3: user deleted
4: user group deleted
5: user logon
6: user logoff
7: user access request
8: user access granted
9: user access violation

B.2. DeviceSecurityEvent Class Description

The purpose of a DeviceSecurityEvent is to represent Events that provide information from the Device that are important to I2NSF Security. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include alarms and various device statistics (e.g., a type of threshold that was exceeded), which may signal the need for further action.

The DeviceSecurityEvent class defines the following attributes.

B.2.1. The devSecEventContent Attribute

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).

B.2.2. The devSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the devSecEventContent attribute. Values include:

0: unknown
1: GUID (Generic Unique IDentifier)
2: UUID (Universal Unique IDentifier)
3: URI (Uniform Resource Identifier)
4: FQDN (Fully Qualified Domain Name)
5: FQPN (Fully Qualified Path Name)

B.2.3. The devSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that was generated by this device. Values include:

0: unknown
1: communications alarm
2: quality of service alarm
3: processing error alarm
4: equipment error alarm
5: environmental error alarm

Values 1-5 are defined in X.733. Additional types of errors may also be defined.

B.2.4. The devSecEventTypeInfo[0..n] Attribute

This is an optional array of strings, which is used to provide additional information describing the specifics of the Event generated by this Device. For example, this attribute could contain probable cause information in the first array, trend information in the second array, proposed repair actions in the third array, and additional information in the fourth array.

B.2.5. The devSecEventTypeSeverity Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the perceived severity of the Event generated by this Device. Values (which are defined in X.733) include:

0: unknown
1: cleared
2: indeterminate
3: critical
4: major
5: minor
6: warning

B.3. SystemSecurityEvent Class Description

The purpose of a SystemSecurityEvent is to represent Events that are detected by the management system, instead of Events that are generated by a user or a device. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include an event issued by an analytics system that warns against a particular pattern of unknown user accesses, or an Event issued by a management system that represents a set of correlated and/or filtered Events.

The SystemSecurityEvent class defines the following attributes.

B.3.1. The sysSecEventContent Attribute

This is a mandatory string that contains the content of the SystemSecurityEvent. The format of the content is specified in the sysSecEventFormat class attribute, and the type of Event is defined in the sysSecEventType class attribute. An example of the sysSecEventContent attribute is the string "sysadmin3", with the sysSecEventFormat attribute set to 1 (GUID), and the sysSecEventType attribute set to 2 (audit log cleared).

B.3.2. The sysSecEventFormat Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the data type of the sysSecEventContent attribute. Values include:

0: unknown
1: GUID (Generic Unique IDentifier)
2: UUID (Universal Unique IDentifier)
3: URI (Uniform Resource Identifier)
4: FQDN (Fully Qualified Domain Name)
5: FQPN (Fully Qualified Path Name)

B.3.3. The sysSecEventType Attribute

This is a mandatory non-negative enumerated integer, which is used to specify the type of Event that involves this device. Values include:

0: unknown
1: audit log written to
2: audit log cleared
3: policy created
4: policy edited
5: policy deleted
6: policy executed

B.4. TimeSecurityEvent Class Description

The purpose of a TimeSecurityEvent is to represent Events that are temporal in nature (e.g., the start or end of a period of time). Time events signify an individual occurrence, or a time period, in which a significant event happened. Information in this Event may be used as part of a test to determine if the Condition clause in this ECA Policy Rule should be evaluated or not. Examples include issuing an Event at a specific time to indicate that a particular resource should not be accessed, or that different authentication and authorization mechanisms should now be used (e.g., because it is now past regular business hours).

The TimeSecurityEvent class defines the following attributes.

B.4.1. The timeSecEventPeriodBegin Attribute

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).

B.4.2. The timeSecEventPeriodEnd Attribute

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 occurence, and not a time period when the Event can occur, then the timeSecEventPeriodEnd attribute may be ignored.

B.4.3. The timeSecEventTimeZone Attribute

This is a mandatory string attribute, and defines the time zone that this Event occurred in using the format specified in ISO8601.

Appendix C. Network Security Condition Class Definitions

This Appendix defines a preliminary set of Network Security Condition classes, along with their attributes.

C.1. PacketSecurityCondition

The purpose of this Class is to represent 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. This class is abstract, and serves as the superclass of more detailed conditions that act on different types of packet formats. Its subclasses are shown in Figure 16, and are defined in the following sections.

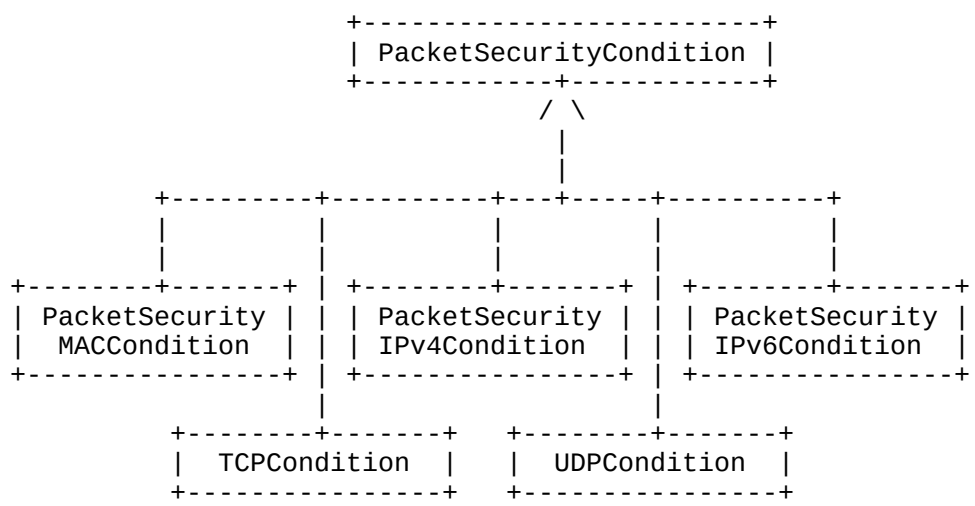


Figure 16. Network Security Info Sub-Model PacketSecurityCondition Class Extensions

C.1.1. PacketSecurityMACCondition

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 executed or not. This class is concrete, and defines the following attributes.

C.1.1.1. The pktSecCondMACDest Attribute

This is a mandatory string attribute, and defines the MAC destination address (6 octets long).

C.1.1.2. The pktSecCondMACSrc Attribute

This is a mandatory string attribute, and defines the MAC source address (6 octets long).

C.1.1.3. The pktSecCondMAC8021Q Attribute

This is an optional string attribute, and defines the 802.1Q tag value (2 octets long). This defines VLAN membership and 802.1p priority values.

C.1.1.4. The pktSecCondMACEtherType Attribute

This is a mandatory string attribute, and defines 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.

C.1.1.5. The pktSecCondMACTCI Attribute

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).

C.1.2. PacketSecurityIPv4Condition

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. This class is concrete, and defines the following attributes.

C.1.2.1. The pktSecCondIPv4SrcAddr Attribute

This is a mandatory string attribute, and defines the IPv4 Source Address (32 bits).

C.1.2.2. The pktSecCondIPv4DestAddr Attribute

This is a mandatory string attribute, and defines the IPv4 Destination Address (32 bits).

C.1.2.3. The pktSecCondIPv4ProtocolUsed Attribute

This is a mandatory string attribute, and defines the protocol used in the data portion of the IP datagram (8 bits).

C.1.2.4. The pktSecCondIPv4DSCP Attribute

This is a mandatory string attribute, and defines the Differentiated Services Code Point field (6 bits).

C.1.2.5. The pktSecCondIPv4ECN Attribute

This is an optional string attribute, and defines the Explicit Congestion Notification field (2 bits).

C.1.2.6. The pktSecCondIPv4TotalLength Attribute

This is a mandatory string attribute, and defines the total length of the packet (including header and data) in bytes (16 bits).

C.1.2.7. The pktSecCondIPv4TTL Attribute

This is a mandatory string attribute, and defines the Time To Live in seconds (8 bits).

C.1.3. PacketSecurityIPv6Condition

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. This class is concrete, and defines the following attributes.

C.1.3.1. The pktSecCondIPv6SrcAddr Attribute

This is a mandatory string attribute, and defines the IPv6 Source Address (128 bits).

C.1.3.2. The pktSecCondIPv6DestAddr Attribute

This is a mandatory string attribute, and defines the IPv6 Destination Address (128 bits).

C.1.3.3. The pktSecCondIPv6DSCP Attribute

This is a mandatory string attribute, and defines the Differentiated Services Code Point field (6 bits). It consists of the six most significant bits of the Traffic Class field in the IPv6 header.

C.1.3.4. The pktSecCondIPv6ECN Attribute

This is a mandatory string attribute, and defines the Explicit Congestion Notification field (2 bits). It consists of the two least significant bits of the Traffic Class field in the IPv6 header.

C.1.3.5. The pktSecCondIPv6FlowLabel Attribute

This is a mandatory string attribute, and defines an IPv6 flow label. This, in combination with the Source and Destination Address fields, enables efficient IPv6 flow classification by using only the IPv6 main header fields (20 bits).

C.1.3.6. The pktSecCondIPv6PayloadLength Attribute

This is a mandatory string attribute, and defines the total length of the packet (including the fixed and any extension headers, and data) in bytes (16 bits).

C.1.3.7. The pktSecCondIPv6NextHeader Attribute

This is a mandatory string attribute, and defines the type of the next header (e.g., which extension header to use) (8 bits).

C.1.3.8. The pktSecCondIPv6HopLimit Attribute

This is a mandatory string attribute, and defines the maximum number of hops that this packet can traverse (8 bits).

C.1.4. PacketSecurityTCPCondition

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. This class is concrete, and defines the following attributes.

C.1.4.1. The pktSecCondTPCSrcPort Attribute

This is a mandatory string attribute, and defines the Source Port number (16 bits).

C.1.4.2. The pktSecCondTPCDestPort Attribute

This is a mandatory string attribute, and defines the Destination Port number (16 bits).

C.1.4.3. The pktSecCondTCPSeqNum Attribute

This is a mandatory string attribute, and defines the sequence number (32 bits).

C.1.4.4. The pktSecCondTCPFlags Attribute

This is a mandatory string attribute, and defines the nine Control bit flags (9 bits).

C.1.5. PacketSecurityUDPCondition

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. This class is concrete, and defines the following attributes.

C.1.5.1.1. The pktSecCondUDPSrcPort Attribute

This is a mandatory string attribute, and defines the UDP Source Port number (16 bits).

C.1.5.1.2. The pktSecCondUDPDestPort Attribute

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

C.1.5.1.3. The pktSecCondUDPLength Attribute

This is a mandatory string attribute, and defines the length in bytes of the UDP header and data (16 bits).

C.2. PacketPayloadSecurityCondition

The purpose of this Class is to represent packet payload data 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. Examples include a specific set of bytes in the packet payload.

C.3. TargetSecurityCondition

The purpose of this Class is to represent information about different targets of this policy (i.e., entities to which this Policy Rule should be applied), which 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. Examples include whether the targeted entities are playing the same role, or whether each device is administered by the same set of users, groups, or roles. This Class has several important subclasses, including:

a. ServiceSecurityContextCondition is the superclass for all information that can be used in an ECA Policy Rule that specifies data about the type of service to be analyzed (e.g., the protocol type and port number)
b. ApplicationSecurityContextCondition is the superclass for all information that can be used in a ECA Policy Rule that specifies data that identifies a particular application (including metadata, such as risk level)
c. DeviceSecurityContextCondition is the superclass for all information that can be used in a ECA Policy Rule that specifies data about a device type and/or device OS that is being used

C.4. UserSecurityCondition

The purpose of this Class is to represent data about the user or group referenced in this ECA Policy Rule that can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include the user or group id used, the type of connection used, whether a given user or group is playing a particular role, or whether a given user or group has failed to login a particular number of times.

C.5. SecurityContextCondition

The purpose of this Class is to represent security conditions that are part of a specific context, which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include testing to determine if a particular pattern of security-related data have occurred, or if the current session state matches the expected session state.

C.6. GenericContextSecurityCondition

The purpose of this Class is to represent generic contextual information in which this ECA Policy Rule is being executed, which can be used as part of a test to determine if the set of Policy Actions in this ECA Policy Rule should be evaluated or not. Examples include geographic location and temporal information.

Appendix D. Network Security Action Class Definitions

This Appendix defines a preliminary set of Network Security Action classes, along with their attributes.

D.1. IngressAction

The purpose of this Class is to represent actions performed on packets that enter an NSF. Examples include pass, dropp, or mirror traffic.

D.2. EgressAction

The purpose of this Class is to represent actions performed on packets that exit an NSF. Examples include pass, drop, or mirror traffic, signal, and encapsulate.

D.3. ApplyProfileAction

The purpose of this Class is to define the application of a profile to packets to perform content security and/or attack mitigation control.

Appendix E. Geometric Model

The geometric model defined in [Bas12] is summarized here. Note that our work has extended the work of [Bas12] to model ECA Policy Rules, instead of just condition-action Policy Rules. However, the geometric model in this Appendix is simplified in this version of this I-D, and is used to define just the CA part of the ECA model.

All the actions available to the security function are well known and organized in an action set A.

For filtering controls, the enforceable actions are either Allow or Deny, thus A={Allow,Deny}. For channel protection controls, they may be informally written as "enforce confidentiality", "enforce data authentication and integrity", and "enforce confidentiality and data authentication and integrity". However, these actions need to be instantiated to the technology used. For example, AH-transport mode and ESP-transport mode (and combinations thereof) are a more precise definition of channel protection actions.

Conditions are typed predicates concerning a given selector. A selector describes the values that a protocol field may take. For example, the IP source selector is the set of all possible IP addresses, and it may also refer to the part of the packet where the values come from (e.g., the IP source selector refers to the IP source field in the IP header). Geometrically, a condition is the subset of its selector for which it evaluates to true. A condition on a given selector matches a packet if the value of the field referred to by the selector belongs to the condition. For instance, in Figure 17 the conditions are s1 <= S1 (read as s1 subset of or equal to S1) and s2 <= S2 (s2 subset of or equal to S2), both s1 and s2 match the packet x1, while only s2 matches x2.

To consider conditions in different selectors, the decision space is extended using the Cartesian product because distinct selectors refer to different fields, possibly from different protocol headers. Hence, given a policy-enabled element that allows the definition of conditions on the selectors S1, S2,..., Sm (where m is the number of Selectors available at the security control we want to model), its selection space is:

S=S1 X S2 X ... X Sm

To consider conditions in different selectors, the decision space is extended using the Cartesian product because distinct selectors refer to different fields, possibly from different protocol headers.

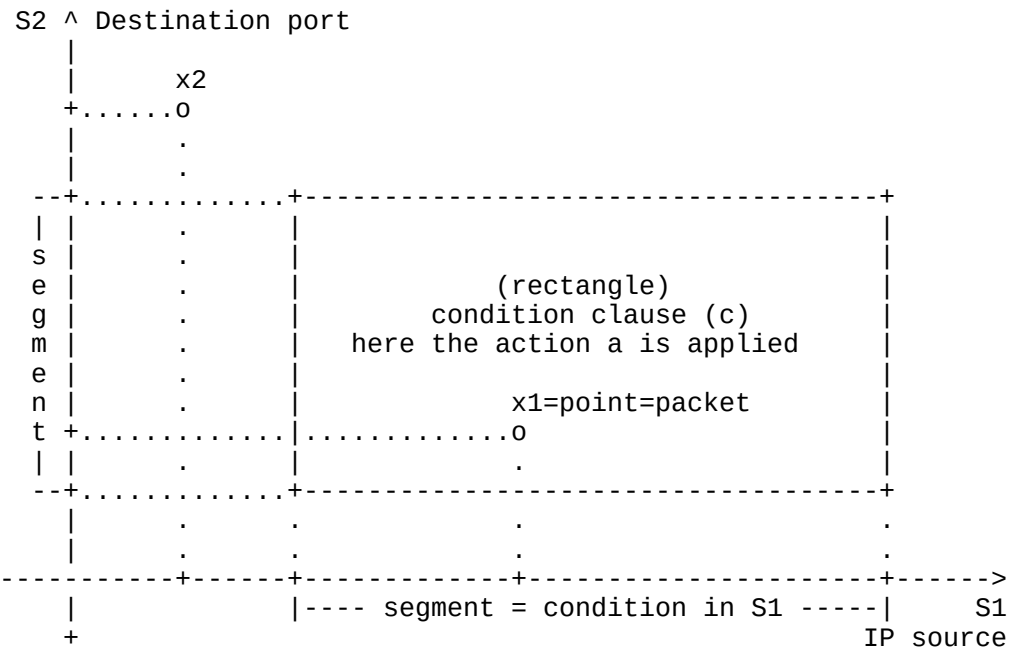


Figure 17: Geometric representation of a rule r=(c,a) that matches x1, but does not match x2.

Accordingly, the condition clause c is a subset of S:

c = s1 X s2 X ... X sm <= S1 X S2 X ... X Sm = S

S represents the totality of the packets that are individually selectable by the security control to model when we use it to enforce a policy. Unfortunately, not all its subsets are valid condition clauses: only hyper-rectangles, or the union of hyper-rectangles (as they are Cartesian product of conditions), are valid. This is an intrinsic constraint of the policy language, as it specifies rules by defining a condition for each selector. Languages that allow specification of conditions as relations over more fields are modeled by the geometric model as more complex geometric shapes determined by the equations. However, the algorithms to compute intersections are much more sophisticated than intersection hyper-rectangles. Figure 17 graphically represents a condition clause c in a two-dimensional selection space.

In the geometric model, a rule is expressed as r=(c,a), where c <= S (the condition clause is a subset of the selection space), and the action a belongs to A. Rule condition clauses match a packet (rules match a packet), if all the conditions forming the clauses match the packet. In Figure 17, the rule with condition clause c matches the packet x1 but not x2.

The rule set R is composed of n rules ri=(ci,ai).

The decision criteria for the action to apply when a packet matches two or more rules is abstracted by means of the resolution strategy

RS: Pow(R) -> A

where Pow(R) is the power set of rules in R.

Formally, given a set of rules, the resolution strategy maps all the possible subsets of rules to an action a in A. When no rule matches a packet, the security controls may select the default action d in A, if they support one.

Resolution strategies may use, besides intrinsic rule data (i.e., condition clause and action clause), also external data associated to each rule, such as priority, identity of the creator, and creation time. Formally, every rule ri is associated by means of the function e(.):

e(ri) = (ri,f1(ri),f2(ri),...)

where E={fj:R -> Xj} (j=1,2,...) is the set that includes all functions that map rules to external attributes in Xj. However, E, e, and all the Xj are determined by the resolution strategy used.

A policy is thus a function p: S -> A that connects each point of the selection space to an action taken from the action set A according to the rules in R. By also assuming RS(0)=d (where 0 is the empty-set) and RS(ri)=ai, the policy p can be described as:

p(x)=RS(match{R(x)}).

Therefore, in the geometric model, a policy is completely defined by the 4-tuple (R,RS,E,d): the rule set R, the resolution function RS, the set E of mappings to the external attributes, and the default action d.

Note that, the geometric model also supports ECA paradigms by simply modeling events like an additional selector.

Authors' Addresses
Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China
Email: Frank.xialiang@huawei.com

John Strassner
Huawei
Email: John.sc.Strassner@huawei.com

Cataldo Basile
Politecnico di Torino
Corso Duca degli Abruzzi, 34
Torino, 10129
Italy
Email: cataldo.basile@polito.it

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain
Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2018

L. Xia
D. Zhang
Huawei
Y. Wu
Aliababa Group
R. Kumar
A. Lohiya
Juniper Networks
H. Birkholz
Fraunhofer SIT
July 04, 2017

An Information Model for the Monitoring of Network Security Functions (NSF)
draft-zhang-i2nsf-info-model-monitoring-04

Abstract

The Network Security Functions (NSF) NSF-facing interface exists between the Service Provider's management system (or Security Controller) and the NSFs to enforce the security policy provisioning and network security status monitoring. This document focuses on the monitoring part of it and proposes the information model for it.

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 5, 2018.

Copyright Notice

Table of Contents

1. Introduction

According to [I-D.ietf-i2nsf-terminology], the interface provided by a NSF (e.g., FW, IPS, Anti-DDOS, or Anti-Virus function) to administrative entities (e.g., NMS, security controller) for configuring security function in the NSF and monitoring the NSF is referred to as a "I2NSF NSF-Facing Interface". The monitoring part of it is meant to acquire vital information about the NSF via, e.g. notifications, events, records, counters. The monitoring of the NSF plays a very important role in the overall security framework if done in a timely and comprehensive way. The monitoring information generated by a NSF can very well be an early indication of malicious activity, or anomalous behavior, or a potential sign of denial of service attacks.

This draft proposes a comprehensive NSF monitoring information model that provides visibility into NSFs. This document will not go into the design details of NSF-Facing Interfaces. Instead, this draft is focused on specifying the information and illustrates the methods that enable a NSF to provide the information required in order to be monitored in a scalable and efficient way via the NSF-Facing Interface. The information model for monitoring presented in this document is a complement to the information model for the security policy provisioning part of the NSF-Facing Interface specified in [I-D.xibassnez-i2nsf-capability].

2. Terminology

2.1. Key Words

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].

2.2. Definition of Terms

This document uses the terms defined in [I-D.ietf-i2nsf-terminology].

3. Use cases for NSF Monitoring Data

As mentioned earlier, monitoring plays a very critical role in the overall security framework. The monitoring of the NSF provides very valuable information to the security controller in maintaining the provisioned security posture. Besides this, there are various other reasons to monitor the NSFs as listed below:

- The security administrator can configure a policy that is triggered on a specific event happened in the NSF or the network. The security controller would monitor for the specified event and once it happens, it configures additional security functions as per the policy.

- The events triggered by NSFs as a result of security policy violation can be used by SIEM to detect any suspicious activity.

- The events and activity logs from NSFs can be used to build advanced analytics such as behavior and predictive to improve the security posture.

- The security controller can use events from the NSF for achieving high availability. It can take corrective actions such as restarting a failed NSF, horizontally scaling the NSF etc.

- The events and activity logs from the NSF can aid in debugging and root cause analysis of an operational issue.

- The activity logs from the NSF can be used to build historical data for operational and business reasons.

4. Classification of NSF Monitoring Data

In order to maintain a strong security posture, it is not only necessary to configure NSF security policies but also to continuously monitor NSF by consuming acquirable observable information. This

enables security admins to assess what is happening in the network timely. It is not possible to block all the internal and external threats based on static security posture. To achieve this goal, a very dynamic posture with constant visibility is required. This draft defines a set of information elements (and their scope) that can be acquired from NSF and can be used as monitoring information. In essence, these types of monitoring information can be leveraged to support constant visibility on multiple levels of granularity and can be consumed by corresponding functions.

Three basic domains about the monitoring of information originating from a system entity [RFC4949] or a NSF are highlighted in this document.

- Retention and Emission
- Notifications and Events
- Unsolicited Poll and Solicited Push

The Alarm Management Framework in [RFC3877] defines an Event as "something that happens which may be of interest. A fault, a change in status, crossing a threshold, or an external input to the system, for example." In the I2NSF domain, I2NSF events [I-D.ietf-i2nsf-terminology] are created and the scope of the Alarm Management Framework Events is still applicable due to its broad definition. The model presented in this document elaborates on the work-flow of creating I2NSF events in the context of NSF monitoring and on how initial I2NSF events are created.

As with I2NSF components, every generic system entity can include a set of capabilities [I-D.ietf-i2nsf-terminology] that creates information about the context, composition, configuration, state or behavior of that system entity. This information is intended to be provided to other consumers of informations--and in the scope of this document, to monitor that information in an automated fashion.

4.1. Retention and Emission

Typically, a system entity populates standardized interface, such as SNMP, NETCONF, RESTCONF or CoMI to provide and emit created information directly via NSF-Facing Interfaces [I-D.ietf-i2nsf-terminology]. Alternatively, the created information is retained inside the system entity (or hierarchy of system entities in a composite device) via records or counters that are not exposed directly via NSF-Facing Interfaces.

Information emitted via standardized interfaces can be consumed by an I2NSF Agent [I-D.ietf-i2nsf-terminology] that includes the capability to consume information not only via I2NSF Interfaces but also via interfaces complementary to the standardized interfaces a generic system entity provides.

Information retained on a system entity requires a corresponding I2NSF Agent to access aggregated records of information, typically in the form of logfiles or databases. There are ways to aggregate records originating from different system entities over a network, for examples via Syslog [RFC5424] or Syslog over TCP [RFC6587]. But even if records are conveyed, the result is the same kind of retention in form of a bigger aggregate of records on another system entity.

An I2NSF Agent is required to process fresh [RFC4949] records created by I2NSF Functions in order to provide them to other I2NSF Components via corresponding I2NSF Interfaces timely. This process is effectively based on homogenizing functions that can access and convert specific kinds of records into information that can be provided and emitted via I2NSF interfaces.

Retained or emitted, the information required to support monitoring processes has to be processed by an I2NSF Agent at some point in the work-flow. Typical locations of these I2NSF Agents are:

- a system entity that creates the information
- a system entity that retains an aggregation of records
- an I2NSF Component that includes the capabilities of using standardized interfaces provided by other system entities that are not I2NSF Components
- an I2NSF Component that creates the information

4.2. Notifications and Events

A specific task of I2NSF Agents is to process I2NSF Policy Rules [I-D.ietf-i2nsf-terminology]. Rules are composed of three clauses: Events, Conditions, and Actions. In consequence, an I2NSF Event is required to trigger an I2NSF Policy Rule. "An I2NSF 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." [I-D.ietf-i2nsf-terminology], which aligns well with the generic definition of Event from [RFC3877].

The model illustrated in this document introduces a complementary type of information that can be conveyed--notification.

Notification: An occurrence of a change of context, composition, configuration, state or behavior of a system entity that can be directly or indirectly observed by an I2NSF Agent and can be used as input for an event-clause in I2NSF Policy Rules.

A notification is similar to an I2NSF Event with the exception that it is created by a system entity that is not an I2NSF Component and that its importances is yet to be assessed. Semantically, a notification is not an I2NSF Event in the context of I2NSF, although they can potentially use the exact same information or data model. In respect to [RFC3877], a Notification is a specific subset of events, because they convey information about "something that happens which may be of interest". In consequence, Notifications can contain information with very low expressiveness or relevance. Hence, additional post-processing functions, such as aggregation, correlation or simple anomaly detection, might have to be employed to satisfy a level of expressiveness that is required for an event-clause of an I2NSF Policy Rule.

It is important to note that the consumer of a notification (the observer) assesses the importance of a notification and not the producer. The producer can include metadata in a notification that supports the observer in assessing the importance (even metadata about severity), but the deciding entity is an I2NSF Agent.

4.3. Unsolicited Poll and Solicited Push

The freshness of the monitored information depends on the acquisition method. Ideally, an I2NSF Agent is accessing every relevant information about the I2NSF Component and is emitting I2NSF Events to a monitoring NSF timely. Publication of events via a pubsub/broker model, peer-2-peer meshes, or static defined channels are only a few examples on how a solicited push of I2NSF Events can be facilitated. The actual mechanic implemented by an I2NSF Component is out of the scope of this document.

Often, corresponding management interfaces have to be queried in intervals or on-demand if required by an I2NSF Policy rule. In some cases, a collection of information has to be conducted via login mechanics provided by a system entity. Accessing records of information via this kind of unsolicited polls can introduce a significant latency in regard to the freshness of the monitored information. The actual definition of intervals implemented by an I2NSF Component is also out of scope of this document.

4.4. I2NSF Monitoring Terminology for Retained Information

Records: Unlike information emitted via notifications and events, records do not require immediate attention from an analyst but may be useful for visibility and retroactive cyber forensic. Depending on the record format, there are different qualities in regard to structure and detail. Records are typically stored in logfiles or databases on a system entity or NSF. Records in the form of logfiles usually include less structures but potentially more detailed information in regard to changes of an system entity's characteristics. In contrast, databases often use more strict schemas or data models, therefore enforcing a better structure, but inhibit storing information that do not match those models ('closed world assumption'). Records can be continuously processed by I2NSF Agents that act as I2NSF Producer and emit events via functions specifically tailored to a certain type of record. Typically, records are information generated by NSF or system entity about their operational and informational data, or various changes in system characteristics, such as user activities, network/traffic status, network activity, etc. They are important for debugging, auditing and security forensic.

Counters: A specific representation of continuous value changes of information elements that potentially occur in high frequency. A prominent example are network interface counters, e.g. PDU amount or byte amount, drop counters, error counters etc. Counters are useful in debugging and visibility into operational behavior of the NSF. An I2NSF Agent that observes the progression of counters can act as an I2NSF Producer and emit events in respect to I2NSF Policy Rules.

5. Conveyance of NSF Monitoring Information

As per the use cases of NSF monitoring data, information needs to be conveyed to various I2NSF Consumers based on requirements imposed by I2NSF Capabilities and work-flows. There are multiple aspects to be considered in regard to emission of monitoring information to requesting parties as listed below:

- Pull-Push Model: A set of data can be pushed by a NSF to the requesting party or pulled by the requesting party from a NSF. Specific types of information might need both the models at the same time if there are multiple I2NSF Consumers with varying requirements. In general, any I2NSF Event including a high severity assessment is considered to be of great importance and should be processed as soon as possible (push-model). Records, in contrast, are typically not as critical (pull-model). The I2NSF

Architecture does not mandate a specific scheme for each type of information and is therefore out of scope of this document.

- Pub-Sub Model: In order for an I2NSF Provider to push monitoring information to multiple appropriate I2NSF Consumers, a subscription can be maintained by both I2NSF Components. Discovery of available monitoring information can be supported by an I2NSF Controller that takes on the role of a broker and therefore includes I2NSF Capabilities that support registration.

- Export Frequency: Monitoring information can be emitted immediately upon generation by a NSF to requesting I2NSF Consumers or can be pushed periodically. The frequency of exporting the data depends upon its size and timely usefulness. It is out of the scope of I2NSF and left to each NSF implementation.

- Authentication: There may be a need for authentication between I2NSF Producer of monitoring information and corresponding I2NSF Consumer to ensure that critical information remains confidential. Authentication in the scope of I2NSF can also require a corresponding content authorization. This may be necessary, for example, if a NSF emits monitoring information to I2NSF Consumer outside its administrative domain. The I2NSF Architecture does not mandate when and how specific authentication has to be implemented.

- Data-Transfer Model: Monitoring information can be pushed by NSF using a connection-less model that does require a persistent connection or streamed over a persistent connection. An appropriate model depends on the I2NSF Consumer requirements and the semantics of the information to be conveyed.

- Data Model and Interaction Model for Data in Motion: There are a lot of

- transport mechanisms such as IP, UDP, TCP. There are also open source implementations for specific set of data such as systems counter, e.g. IPFIX [RFC7011] or NetFlow [RFC3954]. The I2NSF does not mandate any specific method for a given data set, it is up to each implementation.

6. Basic Information Model for All Monitoring Data

As explained in the above section, there is a wealth of data available from the NSF that can be monitored. Firstly, there must be some general information with each monitoring message sent from an NSF that helps consumer in identifying meta data with that message, which are listed as below:

- message_version: Indicate the version of the data format and is a two-digit decimal numeral starting from 01
- message_type: Event, Alert, Alarm, Log, Counter, etc
- time_stamp: Indicate the time when the message is generated
- vendor_name: The name of the NSF vendor
- NSF_name: The name (or IP) of the NSF generating the message
- Module_name: The module name outputting the message
- Severity: Indicates the level of the logs. There are total eight levels, from 0 to 7. The smaller the numeral is, the higher the severity is.

7. Extended Information Model for Monitoring Data

This section covers the additional information associated with the system messages. The extended information model is only for the structured data such as alarm. Any unstructured data is specified with basic information model only.

[Editors' note]: This section remains the same as -02 version, although the classification of the monitoring data has been changed from -02 version. The new inconsistency will be addressed in next verion.

7.1. System Alarm

7.1.1. Memory Alarm

The following information should be included in a Memory Alarm:

- event_name: 'MEM_USAGE_ALARM'
- module_name: Indicate the NSF module responsible for generating this alarm
- usage: specifies the amount of memory used
- threshold: The threshold triggering the alarm
- severity: The severity of the alarm such as critical, high, medium, low
- message: 'The memory usage exceeded the threshold'

7.1.2. CPU Alarm

The following information should be included in a CPU Alarm:

- event_name: 'CPU_USAGE_ALARM'
- usage: Specifies the amount of CPU used
- threshold: The threshold triggering the event
- severity: The severity of the alarm such as critical, high, medium, low
- message: 'The CPU usage exceeded the threshold'

7.1.3. Disk Alarm

The following information should be included in a Disk Alarm:

- event_name: 'DISK_USAGE_ALARM'
- usage: Specifies the amount of disk space used
- threshold: The threshold triggering the event
- severity: The severity of the alarm such as critical, high, medium, low
- message: 'The disk usage exceeded the threshold'

7.1.4. Hardware Alarm

The following information should be included in a Hardware Alarm:

- event_name: 'HW_FAILURE_ALARM'
- component_name: Indicate the HW component responsible for generating this alarm
- threshold: The threshold triggering the alarm
- severity: The severity of the alarm such as critical, high, medium, low
- message: 'The HW component has failed or degraded'

7.1.5. Interface Alarm

The following information should be included in a Interface Alarm:

- event_name: 'IFNET_STATE_ALARM'
- interface_Name: The name of interface
- interface_state: 'UP', 'DOWN', 'CONGESTED'
- threshold: The threshold triggering the event
- severity: The severity of the alarm such as critical, high, medium, low
- message: 'Current interface state'

7.2. System Events

7.2.1. Access Violation

The following information should be included in this event:

- event_name: 'ACCESS_DENIED'
- user: Name of a user
- group: Group to which a user belongs
- login_ip_address: Login IP address of a user
- authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- message: 'access denied'

7.2.2. Configuration Change

The following information should be included in this event:

- event_name: 'CONFIG_CHANGE'
- user: Name of a user
- group: Group to which a user belongs
- login_ip_address: Login IP address of a user

- authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication

- message: 'Configuration modified'

7.3. System Log

7.3.1. Access Logs

Access logs record administrators' login, logout, and operations on the device. By analyzing them, security vulnerabilities can be identified. The following information should be included in operation report:

- Administrator: Administrator that operates on the device

- login_ip_address: IP address used by an administrator to log in

- login_mode: Specifies the administrator logs in mode e.g. root, user

- operation_type: The operation type that the administrator execute, e.g., login, logout, configuration, etc

- result: Command execution result

- content: Operation performed by an administrator after login.

7.3.2. Resource Utilization Logs

Running reports record the device system's running status, which is useful for device monitoring. The following information should be included in running report:

- system_status: The current system's running status

- CPU_usage: Specifies the CPU usage

- memory_usage: Specifies the memory usage

- disk_usage: Specifies the disk usage

- disk_left: Specifies the available disk space left

- session_number: Specifies total concurrent sessions

- process_number: Specifies total number of system processes

- in_traffic_rate: The total inbound traffic rate in pps
- out_traffic_rate: The total outbound traffic rate in pps
- in_traffic_speed: The total inbound traffic speed in bps
- out_traffic_speed: The total outbound traffic speed in bps

7.3.3. User Activity Logs

User activity logs provide visibility into users' online records (such as login time, online/lockout duration, and login IP addresses) and the actions users perform. User activity reports are helpful to identify exceptions during user login and network access activities.

- user: Name of a user
- group: Group to which a user belongs
- login_ip_address: Login IP address of a user
- authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- access_mode: User access mode. e.g., PPP, SVN, LOCAL
- online_duration: Online duration
- lockout_duration: Lockout duration
- type: 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
- cause: Cause of a failed user activity

7.4. System Counters

7.4.1. Interface counters

Interface counters provide visibility into traffic into and out of NSF, bandwidth usage.

- interface_name: Network interface name configured in NSF
- in_total_traffic_pkts: Total inbound packets

- out_total_traffic_pkts: Total outbound packets
- in_total_traffic_bytes: Total inbound bytes
- out_total_traffic_bytes: Total outbound bytes
- in_drop_traffic_pkts: Total inbound drop packets
- out_drop_traffic_pkts: Total outbound drop packets
- in_drop_traffic_bytes: Total inbound drop bytes
- out_drop_traffic_bytes: Total outbound drop bytes
- in_traffic_ave_rate: Inbound traffic average rate in pps
- in_traffic_peak_rate: Inbound traffic peak rate in pps
- in_traffic_ave_speed: Inbound traffic average speed in bps
- in_traffic_peak_speed: Inbound traffic peak speed in bps
- out_traffic_ave_rate: Outbound traffic average rate in pps
- out_traffic_peak_rate: Outbound traffic peak rate in pps
- out_traffic_ave_speed: Outbound traffic average speed in bps
- out_traffic_peak_speed: Outbound traffic peak speed in bps.

7.5. NSF Events

7.5.1. DDoS Event

The following information should be included in a DDoS Event:

- event_name: 'SEC_EVENT_DDoS'
- sub_attack_type: 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.
- dst_ip: The IP address of a victum under attack
- dst_port: The port numbers that the attrack traffic aims at.
- start_time: The time stamp indicating when the attack started

- o end_time: The time stamp indicating when the attack ended. If the attack is still undergoing when sending out the alarm, this field can be empty.
- o attack_rate: The PPS of attack traffic
- o attack_speed: the bps of attack traffic
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches.

7.5.2. Session Table Event

The following information should be included in a Session Table Event:

- o event_name: 'SESSION_USAGE_HIGH'
- o current: The number of concurrent sessions
- o max: The maximum number of sessions that the session table can support
- o threshold: The threshold triggering the event
- o message: 'The number of session table exceeded the threshold'

7.5.3. Virus Event

The following information should be included in a Virus Event:

- o event_Name: 'SEC_EVENT_VIRUS'
- o virus_type: Type of the virus, e.g., trojan, worm, macro Virus type
- o virus_name
- o dst_ip: The destination IP address of the packet where the virus is found
- o src_ip: The source IP address of the packet where the virus is found
- o src_port: The source port of the packet where the virus is found

- dst_port: The destination port of the packet where the virus is found
- src_zone: The source security zone of the packet where the virus is found
- dst_zone: The destination security zone of the packet where the virus is found
- file_type: The type of the file where the virus is hided within
- file_name: The name of the file where the virus is hided within
- virus_info: The brief introduction of virus
- raw_info: The information describing the packet triggering the event.
- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered
- profile: Security profile that traffic matches.

7.5.4. Intrusion Event

The following information should be included in a Intrustion Event:

- event_name: The name of event: 'SEC_EVENT_Intrusion'
- sub_attack_type: Attack type, e.g., brutal force, buffer overflow
- src_ip: The source IP address of the packet
- dst_ip: The destination IP address of the packet
- src_port:The source port number of the packet
- dst_port: The destination port number of the packet
- src_zone: The source security zone of the packet
- dst_zone: The destination security zone of the packet
- protocol: The employed transport layer protocol, e.g.,TCP, UDP
- app: The employed application layer protocol, e.g.,HTTP, FTP

- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered
- profile: Security profile that traffic matches
- intrusion_info: Simple description of intrusion
- raw_info: The information describing the packet triggering the event.

7.5.5. Botnet Event

The following information should be included in a Botnet Event:

- event_name: the name of event: 'SEC_EVENT_Botnet'
- botnet_name: The name of the detected botnet
- src_ip: The source IP address of the packet
- dst_ip: The destination IP address of the packet
- src_port: The source port number of the packet
- dst_port: The destination port number of the packet
- src_zone: The source security zone of the packet
- dst_zone: The destination security zone of the packet
- protocol: The employed transport layer protocol, e.g.,TCP, UDP
- app: The employed application layer protocol, e.g.,HTTP, FTP
- role: The role of the communicating parties within the botnet:
  1. the packet from zombie host to the attacker
  2. The packet from the attacker to the zombie host
  3. The packet from the IRC/WEB server to the zombie host
  4. The packet from the zombie host to the IRC/WEB server
  5. The packet from the attacker to the IRC/WEB server
  6. The packet from the IRC/WEB server to the attacker

7. The packet from the zombie host to the victim

- botnet_info: Simple description of Botnet
- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered
- profile: Security profile that traffic matches
- raw_info: The information describing the packet triggering the event.

7.5.6. Web Attack Event

The following information should be included in a Web Attack Alarm:

- event_name: the name of event: 'SEC_EVENT_WebAttack'
- sub_attack_type: Concret web attack type, e.g., sql injection, command injection, XSS, CSRF
- src_ip: The source IP address of the packet
- dst_ip: The destination IP address of the packet
- src_port: The source port number of the packet
- dst_port: The destination port number of the packet
- src_zone: The source security zone of the packet
- dst_zone: The destination security zone of the packet
- req_method: The method of requirement. For instance, 'PUT' or 'GET' in HTTP
- req_url: Requested URL
- url_category: Matched URL category
- filtering_type: URL filtering type, e.g., Blacklist, Whitelist, User-Defined, Predefined, Malicious Category, Unknown
- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered

- profile: Security profile that traffic matches.

7.6. NSF Logs

7.6.1. DDoS Logs

Besides the fields in an DDoS Alarm, the following information should be included in a DDoS Logs:

- attack_type: DDoS
- attack_ave_rate: The average pps of the attack traffic within the recorded time
- attack_ave_speed: The average bps of the attack traffic within the recorded time
- attack_pkt_num: The number attack packets within the recorded time
- attack_src_ip: 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.
- action: Actions against DDoS attacks, e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service.

7.6.2. Virus Logs

Besides the fields in an Virus Alarm, the following information should be included in a Virus Logs:

- attack_type: Virus
- protocol: The transport layer protocol
- app: The name of the application layer protocol
- times: The time of detecting the virus
- action: The actions dealing with the virus, e.g., alert, block
- os: The OS that the virus will affect, e.g., all, android, ios, unix, windows

7.6.3. Intrusion Logs

Besides the fields in an Intrusion Alarm, the following information should be included in a Intrusion Logs:

- attack_type: Intrusion
- times: The times of intrusions happened in the recorded time
- os: The OS that the intrusion will affect, e.g., all, android, ios, unix, windows
- action: The actions dealing with the intrusions, e.g., e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service
- attack_rate: NUM the pps of attack traffic
- attack_speed: NUM the bps of attack traffic

7.6.4. Botnet Logs

Besides the fields in an Botnet Alarm, the following information should be included in a Botnet Logs:

- attack_type: Botnet
- botnet_pkt_num:The number of the packets sent to or from the detected botnet
- action: The actions dealing with the detected packets, e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service, etc
- os: The OS that the attack aiming at, e.g., all, android, ios, unix, windows, etc.

7.6.5. DPI Logs

DPI Logs provide statistics on uploaded and downloaded files and data, sent and received emails, and alert and block records on websites. It’s helpful to learn risky user behaviors and why access to some URLs is blocked or allowed with an alert record.

- type: DPI action types. e.g., File Blocking, Data Filtering, Application Behavior Control
- file_name: The file name

- file_type: The file type
- src_zone: Source security zone of traffic
- dst_zone: Destination security zone of traffic
- src_region: Source region of the traffic
- dst_region: Destination region of the traffic
- src_ip: Source IP address of traffic
- src_user: User who generates traffic
- dst_ip: Destination IP address of traffic
- src_port: Source port of traffic
- dst_port: Destination port of traffic
- protocol: Protocol type of traffic
- app: Application type of traffic
- policy_id: Security policy id that traffic matches
- policy_name: Security policy name that traffic matches
- action: Action defined in the file blocking rule, data filtering rule, or application behavior control rule that traffic matches.

7.6.6. Vulnerabillity Scanning Logs

Vulnerability scanning logs record the victim host and its related vulnerability information that should to be fixed. the following information should be included in the report:

- victim_ip: IP address of the victim host which has vulnerabilities
- vulnerability_id: The vulnerability id
- vulnerability_level: The vulnerability level. e.g., high, middle, low
- OS: The operating system of the victim host
- service: The service which has vulnerabillity in the victim host

- protocol: The protocol type. e.g., TCP, UDP
- port: The port number
- vulnerability_info: The information about the vulnerability
- fix_suggestion: The fix suggestion to the vulnerability.

7.6.7. Web Attack Logs

Besides the fields in an Web Attack Alarm, the following information should be included in a Web Attack Report:

- attack_type: Web Attack
- rsp_code: Response code
- req_clientapp: The client application
- req_cookies: Cookies
- req_host: The domain name of the requested host
- raw_info: The information describing the packet triggering the event.

7.7. NSF Counters

7.7.1. Firewall counters

Firewall counters provide visibility into traffic signatures, bandwidth usage, and how the configured security and bandwidth policies have been applied.

- src_zone: Source security zone of traffic
- dst_zone: Destination security zone of traffic
- src_region: Source region of the traffic
- dst_region: Destination region of the traffic
- src_ip: Source IP address of traffic
- src_user: User who generates traffic
- dst_ip: Destination IP address of traffic

- src_port: Source port of traffic
- dst_port: Destination port of traffic
- protocol: Protocol type of traffic
- app: Application type of traffic
- policy_id: Security policy id that traffic matches
- policy_name: Security policy name that traffic matches
- in_interface: Inbound interface of traffic
- out_interface: Outbound interface of traffic
- total_traffic: Total traffic volume
- in_traffic_ave_rate: Inbound traffic average rate in pps
- in_traffic_peak_rate: Inbound traffic peak rate in pps
- in_traffic_ave_speed: Inbound traffic average speed in bps
- in_traffic_peak_speed: Inbound traffic peak speed in bps
- out_traffic_ave_rate: Outbound traffic average rate in pps
- out_traffic_peak_rate: Outbound traffic peak rate in pps
- out_traffic_ave_speed: Outbound traffic average speed in bps
- out_traffic_peak_speed: Outbound traffic peak speed in bps.

7.7.2. Policy Hit Counters

Policy Hit Counters record the security policy that traffic matches and its hit count. It can check if policy configurations are correct.

- src_zone: Source security zone of traffic
- dst_zone: Destination security zone of traffic
- src_region: Source region of the traffic
- dst_region: Destination region of the traffic

- src_ip: Source IP address of traffic
- src_user: User who generates traffic
- dst_ip: Destination IP address of traffic
- src_port: Source port of traffic
- dst_port: Destination port of traffic
- protocol: Protocol type of traffic
- app: Application type of traffic
- policy_id: Security policy id that traffic matches
- policy_name: Security policy name that traffic matches
- hit_times: The hit times that the security policy matches the specified traffic.

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. Security Considerations

The monitoring information of NSF should be protected by the secure communication channel, to ensure its confidentiality and integrity. In another side, the NSF and security controller can all be faked, which lead to undesireable results, i.e., leakage of NSF’s important operational information, faked NSF sending false information to mislead security controller. The mutual authentication is essential to protected against this kind of attack. The current mainstream security technologies (i.e., TLS, DTLS, IPSEC, X.509 PKI) can be employed approriately to provide the above security functions.

In addition, to defend against the DDoS attack caused by a lot of NSFs sending massive monitoring information to the security controller, the rate limiting or similar mechanisms should be considered in NSF and security controller, whether in advance or just in the process of DDoS attack.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <http://www.rfc-editor.org/info/rfc3877>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <http://www.rfc-editor.org/info/rfc4949>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[RFC6587] Gerhards, R. and C. Lonvick, "Transmission of Syslog Messages over TCP", RFC 6587, DOI 10.17487/RFC6587, April 2012, <http://www.rfc-editor.org/info/rfc6587>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <http://www.rfc-editor.org/info/rfc7011>.

10.2. Informative References

[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-03 (work in progress), March 2017.

[I-D.xibassnez-i2nsf-capability] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-02 (work in progress), July 2017.

[RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <http://www.rfc-editor.org/info/rfc3954>.

Acknowledgements

Authors' Addresses

Liang Xia
Huawei

Email: frank.xialiang@huawei.com

Dacheng Zhang
Huawei

Email: dacheng.zhang@huawei.com

Yi Wu
Aliababa Group

Email: anren.wy@alibaba-inc.com

Rakesh Kumar
Juniper Networks

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks

Email: alohiya@juniper.net

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de