Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

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

October 31, 2016

I2NSF Capability YANG Data Model
draft-hares-i2nsf-capability-data-model-00

Abstract

This document defines a YANG data model for capability that enables an I2NSF client to control various network security functions in network security devices via an I2NSF 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 May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

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

- North-bound interface (NBI) provided by the network security functions (NSFs)
- Interface between I2NSF user/client with network controller

This document provides a yang model that defines the capabilities for security devices that can be utilized by I2NSF NBI between the I2RS network controller and the NSF devices to express the NSF devices capabilities. It can also be used by the IN2SF user application (or I2NSF client) to network controller to provide a complete list of the I2NSF capabilities the Network controller can control. This document defines a YANG [RFC6020] data model based on the [i2nsf-cap-inf-im], and initial work done in [i2nsf-service-inf-dm]. Terms used in document are defined in [i2nsf-terminology]. [i2nsf-cap-inf-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-inf-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  inet:ipv4-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-inf-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 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 file-isolate
|  +--rw file-isolate-support?  boolean
|  +--rw file-isolate-fcn*  [file-isolate-fcn-name]
|    +--rw file-isolate-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-inf-im].

<CODE BEGINS> file "ietf-i2nsf-capability@2016-10-31.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: Jinyong Tim Kim
        <mailto:wlsdyd0930@nate.com>

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

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

    revision "2016-10-31"{
      description "Third revision";
      reference
        "draft-xia-i2nsf-capability-interface-im-06
         draft-hares-i2nsf-capability-yang-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 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";
        }
        leaf nsf-address {
          type inet:ipv4-address;
          mandatory true;
          description
            "nsf-address";
        }
        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-inf-im] Xia, L., Strassner, J., Li, K., Zhang, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-06 (work in progress), June 2016.

[i2nsf-problem-statement] Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-02 (work in progress), October 2016.

[i2nsf-service-inf-dm] Xia, L., Strassner, J., and D. Bogdanovic, "Data Model of Interface to Network Security Functions Service Interface", draft-xia-i2nsf-service-interface-dm-00 (work in progress), February 2015.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-02 (work in progress), October 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-06 (work in progress), July 2016.

[supa-policy-info-model] Strassner, J., halpern, j., and S. van der

Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-01 (work in progress), July 2016.

[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.

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

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

October 31, 2016

NSF-Triggered Traffic Steering Framework
draft-hyun-i2nsf-nsf-triggered-steering-00

Abstract

This document describes an architecture of Interface to Network Security Functions (I2NSF) framework which enables traffic steering between Network Security Functions (NSFs) for security policy enforcement. Such traffic steering enables 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.

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 May 4, 2017.

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-statement] [i2nsf-cap-interface-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. [i2nsf-cap-interface-im] proposes an information model for NSF facing interface of the I2NSF framework that enables a network security function to trigger further inspection by calling another network security function based on its own analysis results [i2nsf-framework]. 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 Network Security Functions (NSFs) into Interface to Network Security Functions (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 if needed.

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 [i2nsf-cap-interface-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 [i2nsf-cap-interface-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.

- o 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 SFF. 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.

- o Security Function Forwarder (SFF): 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 SFF is responsible for transporting traffic to another SFF (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 Client 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.

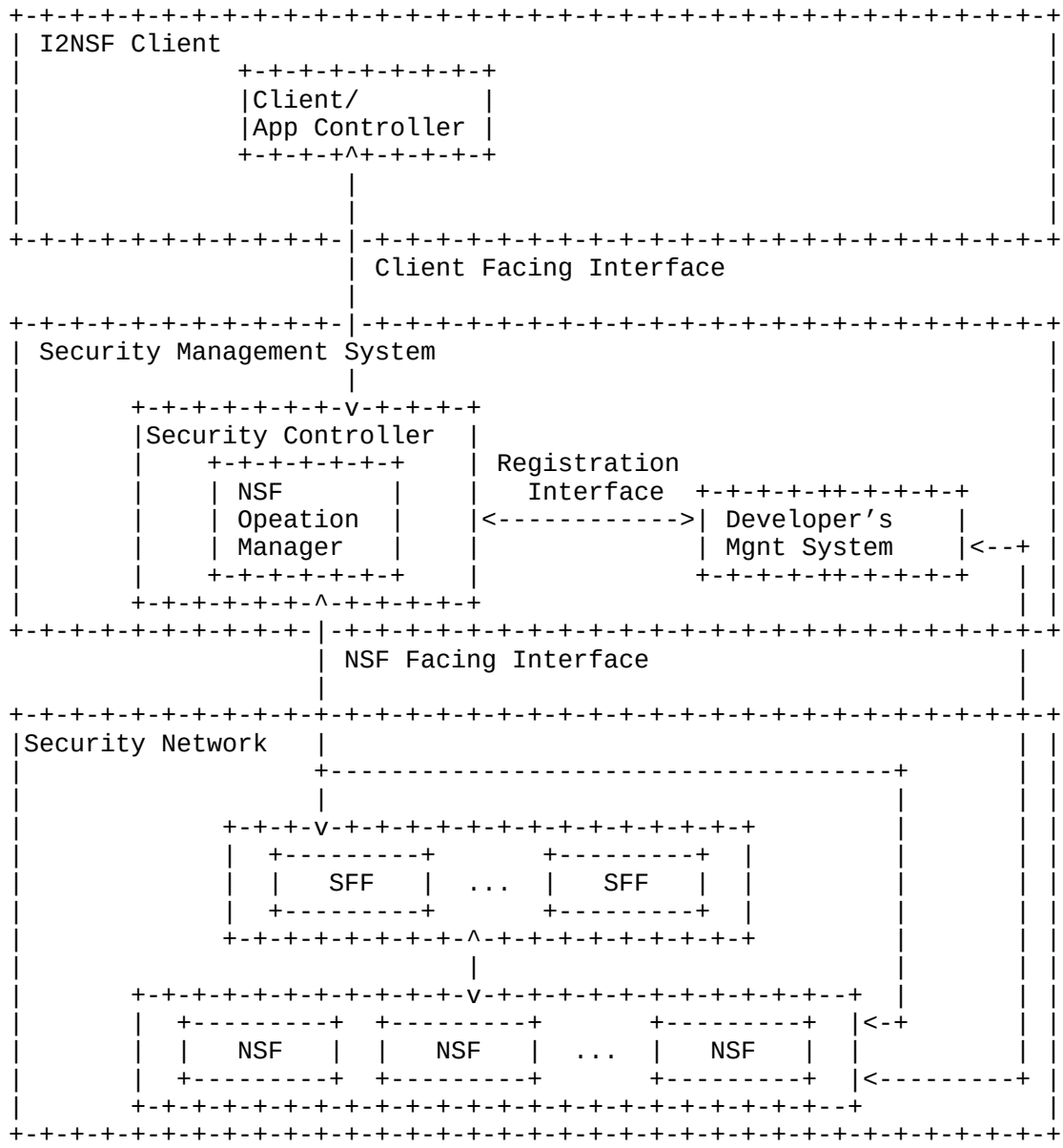


Figure 1: NSF-triggered Traffic Steering Architecture

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 SFF 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 described in Figure 1, 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 SFF. Once receiving a query of a certain NSF profile from SFF, 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 SFF.

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. Packet Forwarding Header

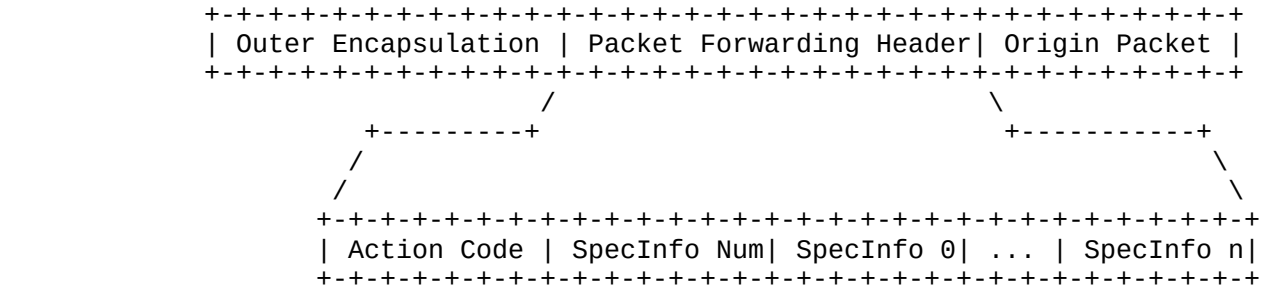


Figure 2: Packet Forwarding Header Format

Packet Forwarding Header is used to convey inspection result and required inspection to an SFF, so it has variable length of fields like Figure 2. It contains fixed Action and the SpecInfo Num fields and variable SpecInfo fields. Action field has a value out of "allow", "deny", "advanced", and "mirror". SpecInfo Num field represents how many SpecInfos are included in the Packet Forwarding Header and each SpecInfo can include a part of NSF Profile which is required for the next inspection. For instance, SepcInfo can be "syn-flood-mitigate", "udp-flood-mitigate", "content-matching-tcp" etc, which are the service profile of an NSF.

4.4. Security Function Forwarder (SFF)

It is responsible for the following two functionalities: (1) Initiallyu forwarding the incoming traffic/packets to Network Security Sub-Module, as described in the I2NSF information model for NSF facing interface [i2nsf-cap-interface-im]. (2) Forwarding the traffic/packets to the matched NSF with the NSF profile which is specified in a Packet Forwarding Header.

An SFF 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 [i2nsf-cap-interface-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

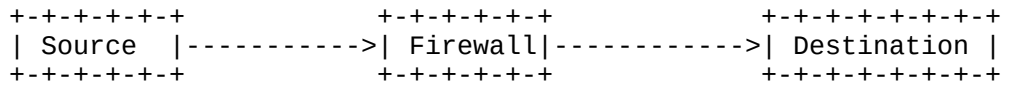
specific NSF profile, an SFF searches for an available NSF instance which provides the network security service corresponding to (matching with) the NSF profile and forward the packet to the NSF instance. If an NSF decides that the packet requires further inspection via another type of network security 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 SFF. Once receiving the packet, the SFF 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. Use Cases

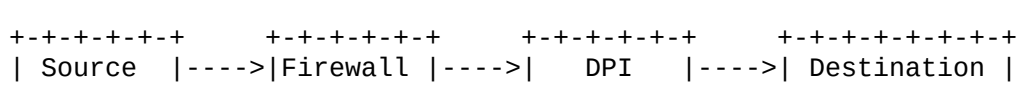
This section introduces two use cases for the NSF-triggered Traffic Steering Framework: (1) Enforcing Different NSFs Depending on a Packet Source’s Trust Level, (2) Effective Load Balancing with Dynamic NSF Instantiation.

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

In the proposed architecture, all incoming packets initially arrive at the SFF. We assume that the current security policy forces all incoming packets to be by default inspected by a firewall in this scenario. Thus the SFF 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 3-(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 SFF. Once receiving the packet, the SFF forwards the packet to the DPI instance which will perform detailed inspection for the packet payload. Figure 3-(b) illustrates this case.



(a) Traffic flow of trusted source



(b) Traffic flow of untrusted source

Figure 3: Different path allocation depending on source of traffic

5.2. 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 SFF of this update.

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

6. 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.

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

[RFC7665] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7665, March 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-01 (work in progress), March 2016.

8.2. Informative References

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

[i2nsf-cap-interface-im] Xia, L., Strassner, J., Li, K., Zhang, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-06 (work in progress), June 2016.

[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-statement] Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-02 (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-02 (work in progress), October 2016.

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

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/

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

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 305-700
Republic of Korea

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

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: October 28, 2017

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei

April 26, 2017

Requirements for Client-Facing Interface to Security Controller
draft-ietf-i2nsf-client-facing-interface-req-01

Abstract

This document captures requirements for Client-Facing interface to Security Controller. The interface is expressed using objects and constructs understood by Security Admin as opposed to vendor or device specific expressions associated with individual product and feature. This document identifies a broad set of requirements needed to express security policies based on User-constructs which are well understood by User Community. This gives ability to decouple policy definition from policy enforcement on a specific element, be it a physical or virtual.

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 October 28, 2017.

Table of Contents

1. Introduction

Programming security policies in a network has been a fairly complex task that often requires very deep knowledge of vendor specific device and features. This has been the biggest challenge for both Service Provider and Enterprise, henceforth named as Security Admin in this document. This challenge is further amplified due to network virtualization with security functions deployed in physical and virtual form factor, henceforth named as network security function (NSF) in this document, from multiple vendors with proprietary interface.

Even if a Security Admin deploys a single vendor solution with one or more security appliances across its entire network, it is still very difficult to manage security policies due to complexity of security features, and mapping of business requirements to vendor specific configuration. The Security Admin may use vendor provided management systems to provision and manage security policies. But, the single vendor approach is highly restrictive in today's network for following reasons:

- An organization may not be able to rely on a single vendor because the changing security requirements may not align with vendor's release cycle.

- A large organization may have a presence across different sites and regions; which means, it may not be possible to deploy same solution from the same vendor because of regional regulatory and compliance policy.

- If and when an organization migrates from one vendor to another, it is almost impossible to migrate security policies from one vendor to another without complex and time consuming manual workflows.

- An organization may deploy multiple network security functions in virtual and physical forms to attain the flexibility, elasticity, performance scale, and operational efficiency they require. Practically, that often requires different sources (vendor, open source) to get the best of breed for any such security function.

- o An organization may choose all or part of their assets such as routers, switches, firewalls, and overlay-networks as policy enforcement points for operational and cost efficiency. It would be highly complex to manage policy enforcement with different tool set for each type of device.

In order to facilitate deployment of security policies across different vendor provided NSFs, the Interface to Network Security Functions (I2NSF) working group in the IETF is defining a Client-Facing interface to Security Controller [I-D. ietf-i2nsf-framework] [I-D. ietf-i2nsf-terminology]. Deployment facilitation should be agnostic to the type of device, be it physical or virtual, or type of enforcement point. Using these interfaces, it becomes possible to write different kinds of security management applications (e.g. GUI portal, template engine, etc.) allowing Security Admin to express security policies in an abstract form with choice of wide variety of NSF for policy enforcement. The implementation of security management applications or controller is completely out of the scope of the I2NSF working group, which is only focused on interface definition.

This document captures the requirements for Client-Facing interface that can be easily used by Security Admin without a need for expertise in vendor and device specific feature set. We refer to this as "User-construct" based interfaces. To further clarify, in the scope of this document, the "User-construct" here does not mean some free-from natural language input or an abstract intent such as "I want my traffic secure" or "I don’t want DDoS attacks in my network"; rather the User-construct here means that policies are described using expressions such as application names, application groups, device groups, user groups etc. with a vocabulary of verbs (e.g., drop, tap, throttle), prepositions, conjunctions, conditionals, adjectives, and nouns instead of using standard n-tuples from the packet header.

2. Conventions Used in this Document

BSS: Business Support System

CLI: Command Line Interface

CMDB: Configuration Management Database

Controller: Used interchangeably with Security Controller or management system throughout this document

CRUD: Create, Retrieve, Update, Delete

FW: Firewall

GUI: Graphical User Interface

IDS: Intrusion Detection System

IPS: Intrusion Protection System

LDAP: Lightweight Directory Access Protocol

NSF: Network Security Function, defined by [I-D.ietf-i2nsf-problem-and-use-cases]

OSS: Operation Support System

RBAC: Role Based Access Control

SIEM: Security Information and Event Management

URL: Universal Resource Locator

vNSF: Refers to NSF being instantiated on Virtual Machines

3. Guiding principle for Client-Facing Interface defintion

Client-Facing Interface must ensure that a Security Admin can deploy any NSF from any vendor and should still be able to use the same consistent interface. In essence, this interface must allow a Security Admin to express security policies independent of how NSFs are implemented in their deployment. Henceforth, in this document, we use "security policy management interface" interchangeably when we refer to Client-Facing interface.

3.1. User-construct based modeling

Traditionally, security policies have been expressed using proprietary interface. The interface is defined by a vendor based on proprietary command line text or a GUI based system with implementation specific constructs such IP address, protocol and L4-L7 information. This requires Security Admin to translate their business objectives into vendor provided constructs in order to express a security policy. But, this alone is not sufficient to render a policy in the network; the admin must also understand network and application design to locate a specific policy enforcement point to make sure policy is effective. This may be highly manual task based on specific network design and may become unmanageable in virtualized networks.

The User-construct based framework does not rely on lower level semantics due to problem explained above, but rather uses higher level constructs such as User-group, Application-group, Device-group, Location-group, etcetera. A Security Admin would use these constructs to express a security policy instead of proprietary implementation or feature specific constructs. The policy defined in such a manner is referred to User-construct based policies in this draft. The idea is to enable Security Admin to use constructs they understand best in expressing security policies which simplify their tasks and help avoiding human errors in complex security provisioning.

3.2. Basic rules for Client-Facing Interface definition

The basic rules in defining the Client-Facing interfaces are as follows:

- Not dependent on particular Network topology or the NSF location in the network
- Not requiring deep knowledge of proprietary features and capabilities supported in the deployed NSFsa€
- Independent of NSF type that will implement the user security policy be it a stateful firewall,IDP, IDS, Router, Switch
- Declarative/Descriptive model instead of Imperative/Prescriptive model - What security policy need to be enforced (declarative) instead of how it is implemented (imperative)
- Not dependent on any specific vendor implementation or form-factor (physical, virtual) of the NSF
- Not dependent on how a NSF becomes operational - Network connectivity and other hosting requirements.
- Not dependent on NSF control plane implementation (if there is one) E.g., cluster of NSFs active as one unified service for scale and/ or resilience.
- Not depending on specific data plane implementation of NSF i.e. Encapsulation, Service function chains.

Note that the rules stated above only apply to the Client-Facing interface, which a user would use to express a high level policy. These rules do not apply to the lower layers e.g. Security Controller that convert higher level policies into lower level constructs. The lower layers may still need some intelligence such

as topology awareness, capability of the NSF and its functions, supported encapsulations etc. to convert and apply the policies accurately on the NSF.

3.3. Deployment Models for Implementing Security Policies

Traditionally, medium and large Enterprise deploy management systems to manage their statically defined security policies. This approach may not be suitable nor sufficient for modern automated and dynamic data centers that are largely virtualized and rely on various management systems and controllers to dynamically implement security policies over any types of NSF.

There are two different deployment models in which the Client-Facing interface referred to in this document could be implemented. These models have no direct impact on the Client-Facing interface, but illustrate the overall security policy and management framework and where various processing functions reside. These models are:

a. Policy management without an explicit management system for control of NSFs. In this deployment, Security Controller acts as a NSF management system; it takes information passed over Client-Facing interface and translates into data on I2NSF NSF-facing interface. The NSF-Facing interface is implemented by NSF vendors; this would usually be done by having an I2NSF agent embedded in the NSF. This deployment model is shown in Figure 1.

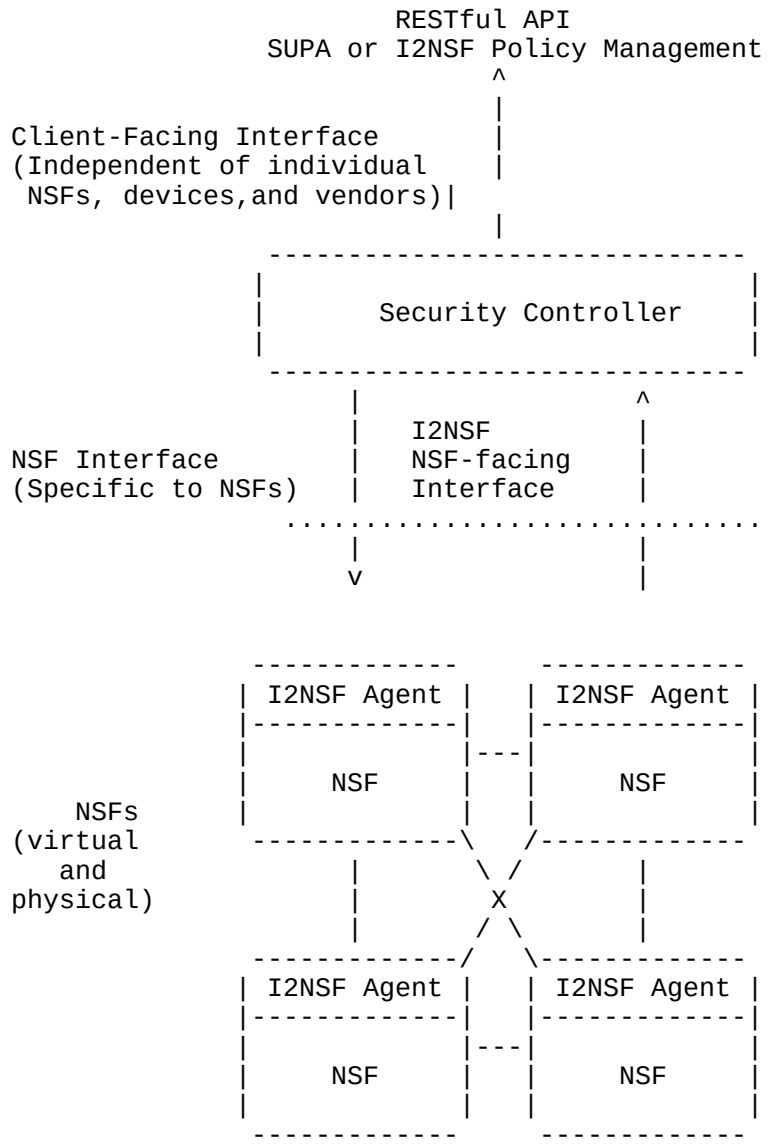
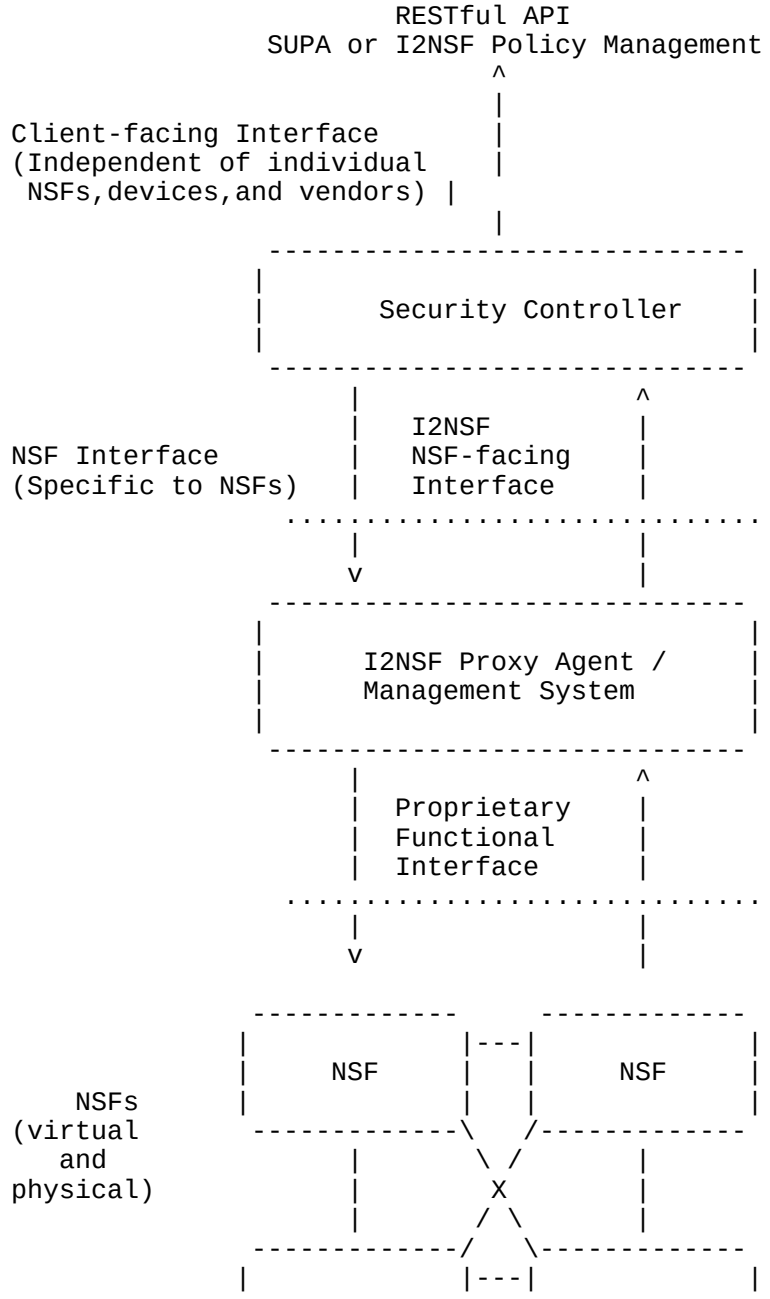


Figure 1: Deployment without Management System

b. Policy management with an explicit management system for control of NSFs. This model is similar to the model above except that Security Controller interacts with a vendor’s dedicated management system that proxy I2NSF NSF-Facing interfaces as NSF

may not support NSF-Facing interface. This is a useful model to support legacy NSF. This deployment model is shown in Figure 2.



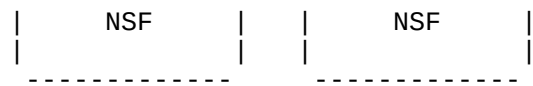


Figure 2: Deployment with Management System or I2NSF Proxy Agent

Although the deployment models discussed here don't necessarily affect the definition of Client-Facing interface, they do give an overall context for defining a security policy interface based on abstraction.

4. Functional Requirements for the Client-Facing Interface

As stated in the guiding principle for defining the I2NSF Client-Facing interface, the security policies and the Client-Facing interface shall be defined from user's or client's perspective and abstracted away from type of NSF, NSF specific implementation, controller implementation, network topology, controller NSF-Facing interface. Thus, the security policy definition shall be declarative, expressed using User-construct, and driven by how Security Admin view security policies from definition, communication and deployment perspective.

Security Controller's' implementation is outside the scope of this document and the I2NSF working group.

In order to express and build security policies, high level requirement for Client-Facing interface is as follows:

- Multi-Tenancy
- Authentication and Authorization
- Role-Based Access Control (RBAC)
- Protection from Attacks
- Protection from Misconfiguration
- Policy Lifecycle Management
- Dynamic Policy Endpoint Groups
- Policy Rules
- Policy Actions

- Generic Policy Model
- Policy Conflict Resolution
- Backward Compatibility
- Third-Party Integration
- Telemetry Data

The above requirements are by no means a complete list and may not be sufficient for all use-cases and all Security Admins, but should be a good starting point for a wide variety of use-cases in Service Provider and Enterprise networks.

4.1. Requirement for Multi-Tenancy in client interface

A Security Admin may have internal tenants and might want a framework wherein each tenant manages its own security policies with isolation from other tenants.

A Security Admin may be a cloud service provider with multi-tenant deployment, where each tenant is a different customer. Each tenant or customer must be allowed to manage its own security policies.

It should be noted that tenants may have their own tenants, so a recursive relation may exist. For instance, a tenant in a Cloud Service Provider may have multiple departments or organizations that need to manage their own security rules for compliance.

Some key concepts are listed below and used throughout the document hereafter:

Policy-Tenant: An entity that owns and manages the security policies applied on its resources.

Policy-Administrator: A user authorized to manage the security policies for a Policy-Tenant.

Policy-User: A user within a Policy-Tenant who is authorized to access certain resources of that tenant according to the privileges assigned to it.

4.2. Requirement for Authentication and Authorization of client interface

Security Admin MUST authenticate to and be authorized by Security Controller before they are able to issue control commands and any policy data exchange commands.

There must be methods defined for Policy-Administrator to be authenticated and authorized to use Security Controller. There are several authentication methods available such as OAuth, XAuth and X.509 certificate based; the authentication may be mutual or single-sided based on business needs and outside the scope of I2NSF. In addition, there must be a method o authorize the Policy-Administrator to perform certain action. It should be noted that, Policy-Administrator authentication and authorization to perform actions could be part of Security Controller or outside; this document does not mandate any specific implementation but requires that such a scheme must be implemented.

4.3. Requirement for Role-Based Access Control (RBAC) in client interface

Policy-Authorization-Role represents a role assigned to a Policy-User that determines whether a user has read-write access, read-only access, or no access for certain resources. A User can be mapped to a Policy-Authorization-Role using an internal or external identity provider or mapped statically.

4.4. Requirement to protect client interface from attacks

The interface must be protections from attacks, malicious or otherwise, from clients or a client impersonator. Potential attacks could come from Botnets, hosts infected with virus or some unauthorized entity. It is recommended that Security Controller use a dedicated IP interface for Client-Facing communications and those communications should be carried over an isolated out-of-band network. In addition, it is recommended that traffic between clients and Security controller be encrypted. Furthermore, some straightforward traffic/session control mechanisms (i.e., Rate-limit, ACL, White/Black list) can be employed on Security Controller to defend against DDoS flooding attacks.

4.5. Requirement to protect client interface from misconfiguration

There must be protections from mis-configured clients. System and policy validations should be implemented to detect this. Validation may be based on a set of default parameters or custom tuned

thresholds such as the number of policy changes submitted, number of objects requested in a given time interval, etc.

4.6. Requirement to manage policy lifecycle with diverse needs

In order to provide more sophisticated security framework, there should be a mechanism so that a policy becomes dynamically active/ enforced or inactive based on Security Admin's manual intervention or some external event.

One example of dynamic policy management is when Security Admin pre-configures all the security policies, but the policies get activated or deactivated based on dynamic threats. Basically, a threat event may activate certain inactive policies, and once a new event indicates that the threat has gone away, the policies become inactive again.

There are following ways for dynamically activating policies:

o The policy may be dynamically activated by Security Admin or associated management entity

o The policy may be dynamically activated by Security Controller upon detecting an external event or an event from I2NSF monitoring interface

o The policy can be statically configured but activated or deactivated upon policy attributes, such as timing calendar

Client-Facing interface should support the following policy attributes for policy enforcement:

Admin-Enforced: A policy, once configured, remains active/enforced until removed by Security Admin.

Time-Enforced: A policy configuration specifies the time profile that determines when the policy is to be activated/enforced. Otherwise, it is de-activated.

Event-Enforced: A policy configuration specifies the event profile that determines when the policy is to be activated/enforced. It also specifies the duration attribute of that policy once activated based on event. For instance, if the policy is activated upon detecting an application flow, the policy could be de-activated when the corresponding session is closed or the flow becomes inactive for certain time.

A policy could be a composite policy, that is composed of many rules, and subject to updates and modification. For the policy maintenance, enforcement, and audit-ability purposes, it becomes important to name and version Security Policy. Thus, the policy definition SHALL support policy naming and versioning. In addition, the i2NSF Client-Facing interface SHALL support the activation, deactivation, programmability, and deletion of policies based on name and version. In addition, it should support reporting operational state of policies by name and version. For instance, a client may probe Security Controller whether a Security Policy is enforced for a tenant and/or a sub-tenant (organization) for audit-ability or verification purposes.

4.7. Requirement to define dynamic policy Endpoint group

When Security Admin configures a Security Policy, it may have requirement to apply this policy to certain subsets of the network. The subsets may be identified based on criteria such as Users, Devices, and Applications. We refer to such a subset of the network as a "Policy Endpoint Group".

One of the biggest challenges for a Security Admin is how to make sure that a Security Policy remain effective while constant changes are happening to the "Policy Endpoint Group" for various reasons (e.g., organizational, network and application changes). If a policy is created based on static information such as user names, application, or network subnets; then every time this static information change, policies need to be updated. For example, if a policy is created that allows access to an application only from the group of Human Resource users (HR-users group), then each time the HR-users group changes, the policy needs to be updated.

We call these dynamic Policy Endpoint Groups "Meta-data Driven Groups". The meta-data is a tag associated with endpoint information such as User, Application, or Device. The mapping from meta-data to dynamic content could come from a standards-based or proprietary tools. Security Controller could use any available mechanisms to derive this mapping and to make automatic updates to policy content if the mapping information changes. The system SHOULD allow for multiple, or sets of tags to be applied to a single network object.

Client-Facing policy interface must support endpoint groups for expressing a Security Policy. The following meta-data driven groups MAY be used for configuring security polices:

User-Group: This group identifies a set of users based on a tag or on static information. The tag to identify user is dynamically derived from systems such as Active Directory or LDAP. For

example, an organization may have different User-groups, such as HR-users, Finance-users, Engineering-users, to classify a set of users in each department.

Device-Group: This group identifies a set of devices based on a tag or on static information. The tag to identify device is dynamically derived from systems such as configuration management database (CMDB). For example, a Security Admin may want to classify all machines running a particular operating system into one group and machines running a different operating system into another group.

Application-Group: This group identifies a set of applications based on a tag or on static information. The tag to identify application is dynamically derived from systems such as CMDB. For example, a Security Admin may want to classify all applications running in the Legal department into one group and all applications running in the HR department into another group. In some cases, the application can semantically associated with a VM or a device. However, in other cases, the application may need to be associated with a set of identifiers (e.g., transport numbers, signature in the application packet payload) that identify the application in the corresponding packets. The mapping of application names/tags to signatures in the associated application packets should be defined and communicated to the NSF. The Client-Facing Interface shall support the communication of this information.

Location-Group: This group identifies a set of location tags. Tag may correspond 1:1 to location. The tag to identify location is either statically defined or dynamically derived from systems such as CMDB. For example, a Security Admin may want to classify all sites/locations in a geographic region as one group.

4.8. Requirement to express rich set of policy rules

The security policy rules can be as simple as specifying a match for the user or application specified through "Policy Endpoint Group" and take one of the "Policy Actions" or more complicated rules that specify how two different "Policy Endpoint Groups" interact with each other. The Client-Facing interface must support mechanisms to allow the following rule matches.

Policy Endpoint Groups: The rule must allow a way to match either a single or a member of a list of "Policy Endpoint Groups".

There must be a way to express a match between two "Policy Endpoint Groups" so that a policy can be effective for communication between two groups.

Direction: The rule must allow a way to express whether Security Admin wants to match the "Policy Endpoint Group" as the source or destination. The default should be to match both directions, if the direction rule is not specified in the policy.

Threats: The rule should allow the security administrator to express a match for threats that come either in the form of feeds (such as Botnet feeds, GeoIP feeds, URL feeds, or feeds from a SIEM) or speciality security appliances. Threats could be identified by Tags/names in policy rules. The tag is a label of one or more event types that may be detected by a threat detection system.

The threat could be from malware and this requires a way to match for virus signatures or file hashes.

4.9. Requirement to express rich set of policy actions

Security Admin must be able to configure a variety of actions within a security policy. Typically, security policy specifies a simple action of "deny" or "permit" if a particular condition is matched. Although this may be enough for most of the simple policies, the I2NSF Client-Facing interface must also provide a more comprehensive set of actions so that the interface can be used effectively across various security functions.

Policy action MUST be extensible so that additional policy action specifications can easily be added.

The following list of actions SHALL be supported:

Permit: This action means continue processing the next rule or allow the packet to pass if this is the last rule. This is often a default action.

Deny: This action means stop further packet processing and drop the packet.

Drop connection: This action means stop further packet processing, drop the packet, and drop connection (for example, by sending a TCP reset).

Log: This action means create a log entry whenever a rule is matched.

Authenticate connection: This action means that whenever a new connection is established it should be authenticated.

Quarantine/Redirect: This action may be relevant for event driven policy where certain events would activate a configured policy that quarantines or redirects certain packets or flows. The redirect action must specify whether the packet is to be tunneled and in that case specify the tunnel or encapsulation method and destination identifier.

Netflow: This action creates a Netflow record; Need to define Netflow server or local file and version of Netflow.

Count: This action counts the packets that meet the rule condition.

Encrypt: This action encrypts the packets on an identified flow. The flow could be over an IPSEC tunnel, or TLS session for instance.

Decrypt: This action decrypts the packets on an identified flow. The flow could be over an IPSEC tunnel, or TLS session for instance.

Throttle: This action defines shaping a flow or a group of flows that match the rule condition to a designated traffic profile.

Mark: This action defines traffic that matches the rule condition by a designated DSCP value and/or VLAN 802.1p Tag value.

Instantiate-NSF: This action instantiates an NSF with a predefined profile. An NSF can be any of the FW, LB, IPS, IDS, honeypot, or VPN, etc.

WAN-Accelerate: This action optimizes packet delivery using a set of predefined packet optimization methods.

Load-Balance: This action load balances connections based on predefined LB schemes or profiles.

The policy actions should support combination of terminating actions and non-terminating actions. For example, Syslog and then Permit; Count and then Redirect.

Policy actions SHALL support any L2, L3, L4-L7 policy actions.

4.10. Requirement to express policy in a generic model

Client-Facing interface SHALL provide a generic metadata model that defines once and then be used by appropriate model elements any times, regardless of where they are located in the class hierarchy, as necessary.

Client-Facing interface SHALL provide a generic context model that enables the context of an entity, and its surrounding environment, to be measured, calculated, and/or inferred.

Client-Facing interface SHALL provide a generic policy model that enables context-aware policy rules to be defined to change the configuration and monitoring of resources and services as context changes.

Client-Facing interface SHALL provide the ability to apply policy or multiple sets of policies to any given object. Policy application process SHALL allow for nesting capabilities of given policies or set of policies. For example, an object or any given set of objects could have application team applying certain set of policy rules, while network team would apply different set of their policy rules. Lastly, security team would have an ability to apply its set of policy rules, being the last policy to be evaluated against.

4.11. Requirement to detect and correct policy conflicts

Client-Facing interface SHALL be able to detect policy "conflicts", and SHALL specify methods on how to resolve these "conflicts"

For example: two clients issues conflicting set of security policies to be applied to the same Policy Endpoint Group.

4.12. Requirement for backward compatibility

It MUST be possible to add new capabilities to Client-Facing interface in a backward compatible fashion.

4.13. Requirement for Third-Party integration

The security policies in a network may require the use of specialty devices such as honeypots, behavioral analytics, or SIEM in the network, and may also involve threat feeds,virus signatures, and malicious file hashes as part of comprehensive security policies.

The Client-Facing interface must allow Security Admin to configure these threat sources and any other information to provide integration and fold this into policy management.

4.14. Requirement to collect telemetry data

One of the most important aspect of security is to have visibility into the networks. As threats become more sophisticated, Security Admin must be able to gather different types of telemetry data from various devices in the network. The collected data could simply be logged or sent to security analysis engines for behavioral analysis, policy violations, and for threat detection.

The Client-Facing interface MUST allow Security Admin to collect various kinds of data from NSFs. The data source could be syslog, flow records, policy violation records, and other available data.

Detailed Client-Facing interface telemetry data should be available between clients and Security Controller. Clients should be able to subscribe and receive these telemetry data.

client should be able to receive notifications when a policy is dynamically updated.

5. Operational Requirements for the Client-Facing Interface

5.1. API Versioning

Client-Facing interface must support a version number for each RESTful API. This is very important because the client application and the controller application may come from different vendors. Even if the vendor is same, it is hard to imagine that two different applications would be released in lock step.

Without API versioning, it is hard to debug and figure out issues if an application breaks. Although API versioning does not guarantee that applications will always work, it helps in debugging if the problem is caused by an API mismatch.

5.2. API Extensiblity

Abstraction and standardization of Client-Facing interface is of tremendous value to Security Admin as it gives them the flexibility of deploying any vendor’s NSF without need to redefine their policies if or when a NSF is changed.

If a vendor comes up with new feature or functionality that can’t be expressed through the currently defined Client-Facing interface, there must be a way to extend existing APIs or to create a new API that is relevant for that NSF vendor only.

5.3. APIs and Data Model Transport

The APIs for client interface must be derived from the YANG based data model. The YANG data model for client interface must capture all the requirements as defined in this document to express a security policy. The interface between a client and controller must be reliable to ensure robust policy enforcement. One such transport mechanism is RESTCONF that uses HTTP operations to provide necessary CRUD operations for YANG data objects, but any other mechanism can be used.

5.4. Notification

Client-Facing interface must allow Security Admin to collect various alarms and events from the NSF in the network. The events and alarms may be either related to security policy itself or related to NSF where the policy is implemented. The events and alarms could also be used as an input to the security policy for autonomous handling as specified in a given security policy.

5.5. Affinity

Client-Facing interface must allow Security Admin to pass any additional metadata that a user may want to provide for a security policy e.g. certain security policy needs to be implemented only on a very highly secure NSF with Trusted Platform Module (TPM) chip. A user may require Security Controller not to share the NSF with other tenant, this must be expressed through the interface API.

5.6. Test Interface

Client-Facing interface must allow Security Admin ability to test a security policy before it is enforced e.g. a user may want to verify whether the policy creates any potential conflicts with the existing policies or if there are even resources to enforce the policy. The test policy would provide such a capability without actually applying the policies.

6. Security Considerations

Client-Facing interface to Security controller must be protected to ensure that entire security posture is not compromised. This draft mandates that interface must have proper authentication and authorization with Role Based Access Controls to address multi-tenancy requirement. The draft does not specify a particular mechanism as different organization may have different needs based on their deployment. This has been discussed extensively in this draft.

Authentication and authorization alone may not be sufficient for Client-Facing interface; the interface API must be validated for proper inputs to guard against attacks. The type of checks and verification may be specific to each interface API, but a careful consideration must be made to ensure that Security Controller is not compromised.

7. IANA Considerations

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

8. Acknowledgements

The authors would like to thank Adrian Farrel, Linda Dunbar and Diego R.Lopez from IETF I2NSF WG for helpful discussions and advice.

The authors would also like to thank Kunal Modasiya, Prakash T. Sehsadri and Srinivas Nimmagadda from Juniper networks for helpful discussions.

9. Normative References

[I-D.ietf-i2nsf-problem-and-use-cases]
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-15 (work in progress), April 2017.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Nabil Bitar
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: nabil.bitar@nokia.com

Senad Palislamovic
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: senad.palislamovic@nokia.com

Liang Xia
Huawei
101 Software Avenue
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

I2NSF
Internet-Draft
Intended status: Informational
Expires: May 3, 2017

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

October 30, 2016

Framework for Interface to Network Security Functions
draft-ietf-i2nsf-framework-04

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 in 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 May 3, 2017.

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 NSFs that are externally provided. It also describes how I2NSF facilitates Software-Defined Networking (SDN) and Network Function Virtualization (NFV) control, while avoiding potential constraints that could limit the internal functionality and capabilities of NSFs.

The I2NSF use cases [I-D.ietf-i2nsf-problem-and-use-cases] 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.

[I-D.ietf-i2nsf-problem-and-use-cases] 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.

2.1. Acronyms

The following acronyms are used in this document:

BSS Business Support System

CDN Content Delivery Networks

ICN Information-Centric Networks

IDS Intrusion Detection System

IoT Internet of Things

IPS Intrusion Protection System

NSF Network Security Function

OSS Operation Support System

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

Consumer

Controller

Firewall

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 security controller; 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 security controller does not participate in NSF data plane activities.

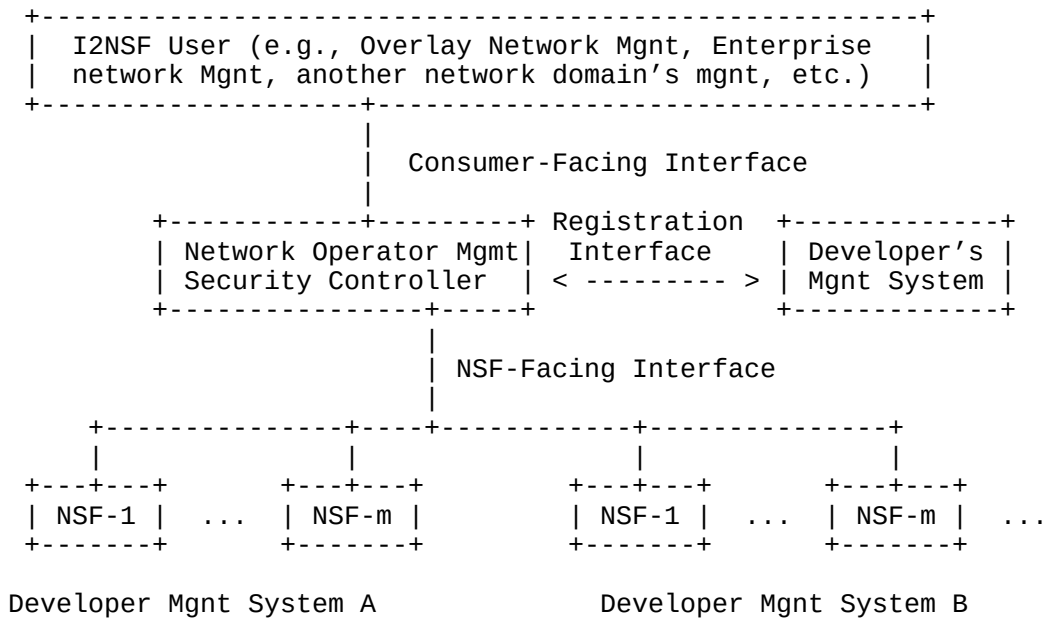


Figure 1: I2NSF Reference Model

When defining controller 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. Consumer-Facing Interface

The 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. In today's world, where everything is connected, preventing unwanted traffic has become a key challenge. More and more networks are implemented as a form of overlay network, with their paths or links among nodes being provided by other networks (a.k.a. underlay networks).

The overlay network's own security solutions cannot prevent various attacks from saturating the access links to the overlay network nodes, which may cause various components of one or more overlay nodes (e.g., CPU or link bandwidth) to become overloaded, and unable to handle their own legitimate traffic. An I2NSF system can be used by overlay networks to request certain flow-based security rules to be enforced by underlay networks. This operates in a similar manner to how traditional networks use firewalls or IPS devices to enforce traffic rules. The I2NSF system can reduce, or even eliminate, unwanted traffic, which prevents unwanted traffic from consuming critical node resources. The same approach can be used by enterprise networks to request their specific flow security policies to be enforced by the provider network that interconnects their users. 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. NSF-Facing Interface

The NSF-Facing Interface is used to specify and monitor flow-based security policies enforced by one or more NSFs. Note that the controller does not need to use all features of a given NSF, nor does it need to use all available NSFs. Hence, this abstraction enables the different features from the set of NSFs that make up able given I2NSF system to be treated as building blocks, so that developers are

free to use the security functions needed independent of vendor and technology.

Flow-based NSFs [I-D.ietf-i2nsf-problem-and-use-cases] inspect packets in the order that they are received. The Interface to flow-based NSFs can be grouped into the following types of Interface Groups:

1. NSF Operational and Administrative Interface: an Interface Group used by a controller to program the operational state of the NSF; this also includes administrative control functions. Since applications and controllers 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 a controller 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 (as dedcribed above). This Interface Group includes logging and query functions between the NSF and external systems. The functionality of this Interface Group may also be defined by other protocols, such as SYSLOG and DOTS.

3. Notification Interface: an Interface Group used by a controller to receive notification events (e.g., alarms) from NSFs. This requires the NSF to be registered. The controller may take an action based on the event; this SHOULD be specified by an I2NSF policy. This Interface Group does NOT change the operational state of the NSF.

This draft proposes that the flow-based paradigm is used 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.

3.3. 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 an interface for vendors to define the capabilities of their NSFs. This Interface Group is called the Registration Interface Group.

An NSF’s capabilities can either be pre-configured or retrieved dynamically through the Registration Interface Group. If a new function that is exposed to the consumer is added to an NSF, then

those capabilities SHOULD be notified to security controllers via the Registration Interface Group.

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 policies and applications of the attacked user 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. This can become especially serious when such a user has higher (or even administration) privileges granted by the provider (the direct NSF provider, the ISP or the underlay network operator).

- A usermay try to install malformed elements (policy or configuration), trying to directly take the control of a NSF or the whole provider platform, for example by exploiting 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 providing the functions (the operating system or the specific NSF implementations) to alter the behavior of the latter. This event has a high impact on all users accessing NSFs as the provider has the highest level of privilege on the software in execution.

- A user that has physical access to the provider platform can modify the behavior of the hardware/software components, or the components themselves. Furthermore, it 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 authentication between the user and the NSF environment and, what is more important, the attestation of the elements in the NSF environment by users could address these threats to an acceptable level of risk. Periodical attestation enables users to detect

alterations in the NSFs and their supporting infrastructure, and raises the degree of physical control necessary to perform an untraceable malicious modification of the environment.

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 not to 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 rules provisioning interfaces to NSFs, 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 rules provisioning interfaces to NSFs 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
- Be a limited lowest common denominator approach, where interfaces can only support a limited set of standardized functions, without allowing for developer-specific functions
- 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 I2NSF Controller

[TBD: should we add the Remote Attestation to this section?]

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

Upon successful authentication, a trusted connection between the user and the Security Controller (or an endpoint designated by it) SHALL 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 user and Security Controller, as described in [I-D.pastor-i2nsf-vnsf-attestation], with the only possible exception of the application of the lowest levels of assurance, in which case the user MUST be made aware of this circumstance.

6.2. Network Connecting the Security 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 Security 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 environment.

- Open environments, where some NSFs can be hosted in external administrative domains or reached via secure external network domains. This requires more restrictive security control to be placed over the I2NSF interface. The information over the I2NSF interfaces SHALL be exchanged used trusted channels as described in the previous section.

When running in an open environment, I2NSF needs to rely on interfaces to properly verify peer identities e.g. through an AAA framework. The implementation of identity management functions is out of scope for I2NSF.

6.3. Interface to vNSFs

Even though there is no difference between virtual network security functions (vNSF) and physical NSFs from the policy provisioning perspective, there are some unique characteristics in interfacing to the vNSFs:

- There could be multiple instantiations of one single NSF that has been distributed across a network. When different instantiations are visible to the Security 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).

- When multiple instantiations of one single NSF appear as one single entity to the Security Controller, the policy provisioning has to be sent to the NSF Manager, which in turn disseminates the polices to the corresponding instantiations of the NSF, as shown in Figure 2 below.

- Policies to one vNSF 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.

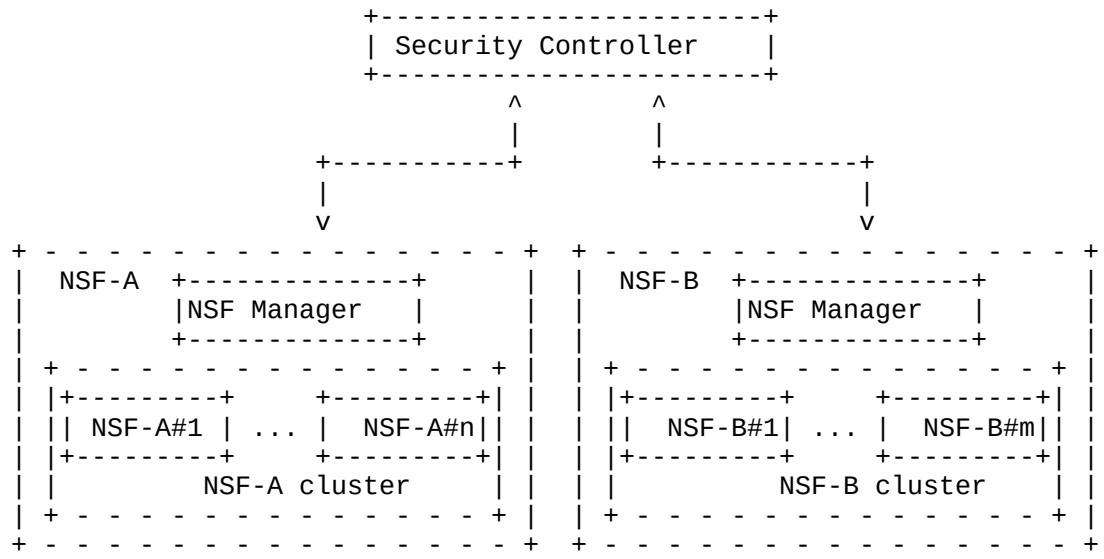


Figure 2: Cluster of NSF Instantiations Management

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 Event - Condition - Action (ECA) policy rulesets.

Event is used to determine whether the condition clause of the Policy Rule can be evaluated or not.

A Condition, when used in the context of policy rules for flow-based NSFs, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. A condition can be based on various combinations of the content (header/payload) and/or the context (session state, authentication state, etc) of the received packets.

Action can be simple permit/deny/rate-limiting, applying specify profile, or establishing specific secure tunnels, etc.

7.1. Customer-Facing Flow Security Policy Structure

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

Some customers may not have security skills. As such, they are not able to express requirements or security policies that are precise enough. These customers may instead express expectations or intent of the functionality desired by their security policies. Customers may also express guidelines such as which certain types of destinations are not allowed for certain groups. As a result, there could be different depths or layers of Service Layer policies. Here are some examples of more abstract security Policies that can be developed based on the I2NSF defined customer-facing interfaces:

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 Customer-Facing Interface may need multiple network functions at various locations to achieve the 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 simple client policies that can be modeled as closely as possible to the flow security policies to individual NSFs. The I2NSF simple client flow policies should have similar structure as the policies to NSFs, but with more of a client-oriented expression for the packet content, context, and other parts of an ECA policy rule. This enables the client to construct an ECA policy rule without having to know actual tags or addresses 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 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 something 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, mirroring, etc.

- 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

[I-D.bogdanovic-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.bogdanovic-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.bogdanovic-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.

The I2NSF "actions" should extend the actions specified by [I-D.bogdanovic-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 very important to have capability discovery or inquiry mechanism over the I2NSF Customer-Facing Interface for the clients to discover if the needed flow polices can be supported or not.

When an NSF cannot perform the desired provisioning (e.g., due to resource constraints), it MUST inform the controller.

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 (but includes security clauses such as isolation requirements, non-via nodes, etc.), could be extended as a basis for the negotiation procedure. Likewise, the companion Connectivity Provisioning Negotiation Protocol (CPNP) could be a candidate to proceed with the negotiation procedure.

The "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.

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 technological capacity increases, integration of platforms, and new threats. At their core:

- 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 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		Port
SCTP		syn
DCCP		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: Subject Capability Index

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: Object 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 Controller to enable/disable

Table 4: Function Capability Index

10. Manageability Considerations

Management of NSFs usually includes:

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

I2NSF only focuses on the policy rule provisioning part, i.e. the last bullet listed above.

11. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security services. 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.

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 Seetharama Rao

Durbha (Cablelabs), Ramki Krishnan (Dell), Anil Lohiya (Juniper Networks), Joe Parrott (BT), 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, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3060] Moore, B., Ellesson, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, DOI 10.17487/RFC3060, February 2001, <http://www.rfc-editor.org/info/rfc3060>.

[RFC3460] Moore, B., Ed., "Policy Core Information Model (PCIM) Extensions", RFC 3460, DOI 10.17487/RFC3460, January 2003, <http://www.rfc-editor.org/info/rfc3460>.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <http://www.rfc-editor.org/info/rfc5575>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <http://www.rfc-editor.org/info/rfc7297>.

14.2. Informative References

[I-D.bogdanovic-netmod-acl-model] Bogdanovic, D., Sreenivasa, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-bogdanovic-netmod-acl-model-02 (work in progress), October 2014.

[I-D.ietf-i2nsf-problem-and-use-cases] Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-02 (work in

progress), October 2016.

[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-02 (work in progress), October 2016.

[I-D.pastor-i2nsf-vnsf-attestation]
Pastor, A., Lopez, D., and A. Shaw, "Remote Attestation Procedures for Network Security Functions (NSFs) through the I2NSF Security Controller", draft-pastor-i2nsf-vnsf-attestation-03 (work in progress), July 2016.

[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.

[ITU-T-X1036]
"ITU-T Recommendation X.1036 - Framework for creation, storage, distribution and enforcement of policies for network security", November 2007.

[NW-2011] Burke, J., "The Pros and Cons of a Cloud-Based Firewall", November 2011.

[SC-MobileNetwork]
Haeffner, W. and N. Leymann, "Network Based Services in Mobile Network", July 2013.

[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
Fortinet
899 Kifer Road
Sunnyvale, CA 94086
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

I2NSF
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2017

S. Hares
L. Dunbar
Huawei
D. Lopez
Telefonica I+D
M. Zarny
Goldman Sachs
C. Jacquenet
France Telecom

October 5, 2016

I2NSF Problem Statement and Use cases
draft-ietf-i2nsf-problem-and-use-cases-02.txt

Abstract

This document describes the problem statement for Interface to Network Security Functions (I2NSF) as well as some companion use cases.

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 April 8, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document describes the problem statement for Interface to Network Security Functions (I2NSF) as well as some I2NSF use cases. A summary of the state of the art in the industry and IETF which is relevant to I2NSF work is documented in [I-D.hares-i2nsf-gap-analysis].

The growing challenges and complexity in maintaining a secure infrastructure, complying with regulatory requirements, and controlling costs are enticing enterprises into consuming network security functions hosted by service providers. The hosted security service is especially attractive to small and medium size enterprises who suffer from a lack of security experts to continuously monitor networks, acquire new skills and propose immediate mitigations to ever increasing sets of security attacks.

According to [Gartner-2013], the demand for hosted (or cloud-based) security services is growing. Small and medium-sized businesses (SMBs) are increasingly adopting cloud-based security services to replace on-premises security tools, while larger enterprises are deploying a mix of traditional and cloud-based security services.

To meet the demand, more and more service providers are providing hosted security solutions to deliver cost-effective managed security services to enterprise customers. The hosted security services are primarily targeted at enterprises (especially small/medium ones), but could also be provided to any kind of mass-market customer. As a result, the Network Security Functions (NSFs) are provided and consumed in a large variety of environments. Users of NSFs may consume network security services hosted by one or more providers, which may be their own enterprise, service providers, or a combination of both. This document also briefly describes the following use cases summarized by [I-D.pastor-i2nsf-merged-use-cases]:

- [I-D.pastor-i2nsf-access-usecases] (I2NSF-Access),
- [I-D.zarny-i2nsf-data-center-use-cases](I2NSF-DC), and
- [I-D.qi-i2nsf-access-network-usecase] (I2NSF-Mobile).

2. Terminology

ACL: Access Control List

B2B: Business-to-Business

Bespoke: Something made to fit a particular person, client or company.

Bespoke security management: Security management which is made to fit a particular customer.

DC: Data Center

FW: Firewall

IDS: Intrusion Detection System

IPS: Intrusion Protection System

I2NSF: interface to Network Security Functions.

NSF: Network Security Function. An NSF is a function that detects abnormal activity and blocks/mitigates the effect of such abnormal activity in order to preserve the availability of a network or a service. In addition, the NSF can help in supporting communication stream integrity and confidentiality.

Flow-based NSF: An NSF which inspects network flows according to a security policy. Flow-based security also means that packets are inspected in the order they are received, and without altering packets due to the inspection process (e.g., MAC rewrites, TTL decrement action, or NAT inspection or changes).

Virtual NSF: An NSF which is deployed as a distributed virtual resource.

VNFPool: Pool of Virtual Network Functions.

3. Problem Space

The following sub-section describes the problems and challenges facing customers and security service providers when some or all of the security functions are no longer physically hosted by the customer’s adminstrative domain.

Security service providers can be internal or external to the company. For example, an internal IT Security group within a large

enterprise could act as a security service provider for the enterprise. In contrast, an enterprise could outsource all security services to an external security service provider. In this document, the security service provider function whether it is internal or external, will be denoted as "service provider".

The "Customer-Provider" relationship may be between any two parties. The parties can be in different firms or different domains of the same firm. Contractual agreements may be required in such contexts to formally document the customer's security requirements and the provider's guarantees to fulfill those requirements. Such agreements may detail protection levels, escalation procedures, alarms reporting, etc. There is currently no standard mechanism to capture those requirements.

A service provider may be a customer of another service provider.

3.1. Challenges Facing Security Service Providers

3.1.1. Diverse Types of Security Functions

There are many types of NSFs. NSFs by different vendors can have different features and have different interfaces. NSFs can be deployed in multiple locations in a given network, and perhaps have different roles.

Below are a few examples of security functions and locations or contexts in which they are often deployed:

External Intrusion and Attack Protection: Examples of this function are firewall/ACL authentication, IPS, IDS, and endpoint protection.

Security Functions in a DMZ: Examples of this function are firewall/ACLs, IDS/IPS, authentication and authorization services, NAT, forwarding proxies, application, and AAA services. These functions may be physically on-premise in a server provider's network at the DMZ spots or located in a "virtual" DMZ.

Internal Security Analysis and Reporting: Examples of this function are security logs, event correlation, and forensic analysis.

Internal Data and Content Protection: Examples of this function are encryption, authorization, and public/private key management for internal database.

Given the diversity of security functions, the contexts in which these functions can be deployed, and the constant evolution of these

functions, standardizing all aspects of security functions is challenging, and most probably not feasible. Fortunately, it is not necessary to standardize all aspects. For example, from an I2NSF perspective, there is no need to standardize on how firewall filters are created or applied.

What is needed is a standardized interface to control and monitor the rule sets that NSFs use to treat packets traversing through. And standardizing interfaces will provide an impetuous for standardizing established security functions.

3.1.2. Diverse Interfaces to Control and Monitor NSFs

To provide effective and competitive solutions and services, Security Service Providers may need to utilize multiple security functions from various vendors to enforce the security policies desired by their customers.

Since no widely accepted industry standard security interface exists today, management of NSFs (device and policy provisioning, monitoring, etc.) tends to be bespoke security management offered by product vendors. As a result, automation of such services, if it exists at all, is also bespoke. Thus, even in the traditional way of deploying security features, there is a gap to coordinate among implementations from distinct vendors. This is the main reason why mono-vendor security functions are often deployed and enabled in a particular network segment.

A challenge for monitoring is that an NSF cannot monitor what it cannot view. Therefore, enabling a security function (e.g., firewall [I-D.ietf-opsawg-firewalls]) does not mean that a network is protected. As such, it is necessary to have a mechanism to monitor and provide execution status of NSFs to security and compliance management tools. There exist various network security monitoring vendor-specific interfaces for forensics and troubleshooting.

3.1.3. More Distributed NSFs and vNSFs

The security functions which are invoked to enforce a security policy can be located in different equipment and network locations.

The European Telecommunications Standards Institute (ETSI) Network Function Virtualization (NFV) initiative creates new management challenges for security policies to be enforced by distributed, virtual, and network security functions (vNSF).

A vNSF has higher risk of failure, migrating, and state changes as their hosting VMs are being created, moved, or decommissioned.

3.1.4. More Demand to Control NSFs Dynamically

In the advent of Software-Defined Networking (see [I-D.jeong-i2nsf-sdn-security-services]), more clients, applications or application controllers need to dynamically update their security policies that are enforced by NSFs. The Security Service Providers have to dynamically update their decision-making process (e.g., in terms of NSF resource allocation and invocation) upon receiving requests from their clients.

3.1.5. Demand for Multi-Tenancy to Control and Monitor NSFs

Service providers may need several operational units to control and monitor the NSFs, especially when NSFs become distributed and virtualized.

3.1.6. Lack of Characterization of NSFs and Capability Exchange

To offer effective security services, service providers need to activate various security functions in NSFs or vNSFs manufactured by multiple vendors. Even within one product category (e.g., firewall), security functions provided by different vendors can have different features and capabilities. For example, filters that can be designed and activated by a firewall may or may not support IPv6 depending on the firewall technology.

The service provider’s management system (or controller) needs a way to retrieve the capabilities of service functions by different vendors so that it could build an effective security solution. These service function capabilities can be documented in a static manner (e.g., a file) or via an interface which accesses a repository of security function capabilities which the NSF vendors dynamically update.

A dynamic capability registration is useful for automation because security functions may be subject to software and hardware updates. These updates may have implications on the policies enforced by the NSFs.

Today, there is no standard method for vendors to describe the capabilities of their security functions. Without a common technical framework to describe the capabilities of security functions, service providers cannot automate the process of selecting NSFs by different vendors to accommodate customer’s requirements.

3.1.7. Lack of Mechanism for NSFs to Utilize External Profiles

Many security functions depend on signature files or profiles to perform (e.g., IPS/IDS signatures, DOTS filters). Different policies might need different signatures or profiles. Today, the construction and use of black list databases can be a win-win strategy for all parties involved. There might be Open Source-provided signature/ profiles (e.g., by Snort or others) in the future.

There is a need to have a standard envelop (i.e., the format) to allow NSFs to use external profiles.

3.1.8. Lack of Mechanisms to Accept External Alerts to Trigger Automatic Rule and Configuration Changes

NSF can ask the I2NSF security controller to alter a specific rules and/or configurations. For example, a DDoS alert could trigger a change to the routing system to send traffic to a traffic scrubbing service to mitigate the DDoS.

The DDoS protection has the following two parts: a) the configuration of signaling of open threats and b) DDoS mitigation. DOTS controller manages the signaling part of DDoS. I2NSF controller(s) would manage the changing to the affected policies (e.g., forwarding and routing, filtering, etc.). By monitoring the network alerts from DDoS, I2NSF can feed an alerts analytics engine that could recognize attacks and the I2NSF can thus enforce the appropriate policies.

DDoS mitigation is enhanced if the provider’s network security controller can monitor, analyze, and investigate the abnormal events and provide information to the client or change the network configuration automatically.

[I-D.zhou-i2nsf-capability-interface-monitoring] provides details on how monitoring aspects of the flow-based Network Security Functions (NSFs) can use the I2NSF interfaces to receive traffic reports and enforce policy.

3.1.9. Lack of Mechanism for Dynamic Key Distribution to NSFs

There is a need for a controller to distribute various keys to distributed NSFs. To distribute various keys, the keys must be created and managed. While there are many key management methods and key derivation functions (KDF), there is a lack of standard interface to provision and manage keys.

The keys may be used for message authentication and integrity in order to protect data flows. In addition, keys may be used to secure

the protocol and messages in the core routing infrastructure ([RFC4948])

As of now there is not much focus on an abstraction for keying information that describes the interface between protocols, operators, and automated key management.

The ability to utilize keys when routing protocols send or receive messages will be enhanced by having an abstract key table maintained by a security service. Conceptually, there must be an interface defined for routing/signaling protocols to make requests for automated key management when it is being used, to notify the protocols when keys become available in the key table.

An abstract key service will work under the following conditions:

1. I2NSF needs to design the key table abstraction, the interface between key management protocols and routing/other protocols, and possibly security protocols at other layers.

2. For each routing/other protocol, I2NSF needs to define the mapping between how the protocol represents key material and the protocol-independent key table abstraction. (If protocols share common mechanisms for authentication (e.g., TCP Authentication Option), then the same mapping may be reused.)

3. Automated Key management must support both symmetric keys and group keys via the service provided by items 1 and 2.

3.2. Challenges Facing Customers

When customers invoke hosted security services, their security policies may be enforced by a collection of security functions hosted in different domains. Customers may not have the security skills to express sufficiently precise requirements or security policies. Usually, these customers express the expectations of their security requirements or the intent of their security policies. These expectations can be considered customer level security expectations. Customers may also desire to express guidelines for security management. Examples of such guidelines include:

- Which critical communications are to be preserved during critical events (DOTS),

- Which hosts are to continue service even during severe security attacks (DOTS),

- Reporting of attacks to CERT (MILE),

o Managing network connectivity of systems out of compliance (SACM),

3.2.1. NSFs from Heterogeneous Administrative Domains

Many medium and large enterprises have deployed various on-premises security functions which they want to continue to deploy. These enterprises want to combine local security functions with remote hosted security functions to achieve more efficient and immediate counter-measures to both Internet-originated attacks and enterprise network-originated attacks.

Some enterprises may only need the hosted security services for their remote branch offices where minimal security infrastructures/ capabilities exist. The security solution will consist of deploying NSFs on customer networks and on service provider networks.

3.2.2. Today’s Control Requests are Vendor Specific

Customers may consume NSFs by multiple service providers. Customers need to express their security requirements, guidelines, and expectations to the service providers. In turn, the service providers must translate this customer information into customer security policies and associated configuration tasks for the set of security functions in their network. Without a standard technical characterization or a standard interface, the service provider faces many challenges:

No standard technical characterization and/or APIs : Even for the most common security services there is no standard technical characterization or APIs. Most security services are accessible only through disparate, proprietary interfaces (e.g., portals or APIs) in whatever format vendors choose to offer. The service provider must have the customer’s input to manage these widely varying interfaces.

No standard interface: Without standard interfaces it is complex for customers to update security policies or integrate the security functions in their enterprise with the security services provided by the security service providers. This complexity is induced by the diversity of the configuration models, policy models, and supported management interfaces. Without a standard interface, new innovative security products find a large barrier to entry into the market.

Managing by scripts de-jour: The current practices rely upon the use of scripts that generate other scripts which automatically run to upload or download configuration changes, log information and other things. These scripts have to be adjusted each time an

implementation from a different vendor technology is enabled on a provider side.

Lack of immediate feedback: Customers may also require a mechanism to easily update/modify their security requirements with immediate effect in the underlying involved NSFs.

Lack of explicit invocation request: While security agreements are in place, security functions may be solicited without requiring an explicit invocation means. Nevertheless, some explicit invocation means may be required to interact with a service function.

To see how standard interfaces could help achieve faster implementation time cycles, let us consider a customer who would like to dynamically allow an encrypted flow with specific port, src/dst addresses or protocol type through the firewall/IPS to enable an encrypted video conferencing call only during the time of the call. With no commonly accepted interface in place, the customer would have to learn about the particular provider's firewall/IPS interface and send the request in the provider's required format.

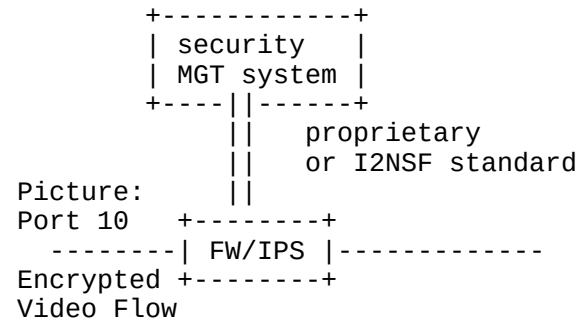


Figure 1: Example of non-standard vs. standard interface

In contrast, if a firewall/IPS interface standard exists, the customer would be able to send the request, without having to do the extensive preliminary legwork. A standard interface also helps service providers since they could now offer the same firewall/IPS interface to represent firewall/IPS services for utilizing products from many vendors. The result is that the service provider has now abstracted the firewall/IPS services. The standard interface also helps the firewall/IPS vendors to focus on their core security functions or extended features rather than the standard building blocks of a management interface.

3.2.3. Difficulty to Monitor the Execution of Desired Policies

How a policy is translated into technology-specific actions is hidden from the customers. However, customers still need ways to monitor the delivered security service that results from the execution of their desired security requirements, guidelines and expectations.

Today, there is no standard way for customers to get security service assurance of their specified security policies properly enforced by the security functions in the provider domain. The customer also lacks the ability to perform "what-if" scenarios to assess the efficiency of the delivered security service.

3.3. Difficulty to Validate Policies across Multiple Domains

One key aspect of a hosted security service with security functions located at different premises is the ability to express, monitor and verify security policies that combine several distributed security functions. It is crucial to an effective service to be able to take these actions via a standard interface. This standard interface becomes more crucial to the hosted security service when NSFs are instantiated in Virtual Machines which are sometimes widely distributed in the network and sometimes are combined together in one device to perform a set of tasks for delivering a service.

Without standard interfaces and security policy data models, the enforcement of a customer-driven security policy remains challenging because of the inherent complexity created by combining the invocation of several vendor-specific security functions into a multi-vendor, heterogeneous environment. Each vendor-specific function may require specific configuration procedures and operational tasks.

Ensuring the consistent enforcement of the policies at various domains is also challenging. Standard data models are likely to contribute to addressing that issue.

3.4. Lack of Standard Interface to Inject Feedback to NSF

Today, many security functions, such as IPS, IDS, DDoS and Antivirus, depend heavily on the associated profiles. They can perform more effective protection if they have the up-to-date profiles. As more sophisticated threats arise, enterprises, vendors, and service providers have to rely on each other to achieve optimal protection. Cyper Threat Alliance (CA, http://cyberthreatalliance.org/) is one of those initiatives that aim at combining efforts conducted by multiple organizations.

Today there is no standard interface to exchange security profiles between organizations.

3.5. Lack of Standard Interface for Capability Negotiation

There could be situations when the selected NSFs cannot perform the policies requested by the Security Controller, due to resource constraints. The customer and security service provider should negotiate the appropriate resource constraints before the security service begins. However, unexpected events somethings happen and the NSF may exhaust those negotiated resources. At this point, the NSF should inform the security controller that the alloted resources have been exhausted. To support the automatic control in the SDN-era, it is necessary to have a set of messages for proper notification (and a response to that notification) between the Security Controller and the NSFs.

4. Use Cases

Standard interfaces for monitoring and controlling the behavior of NSFs are essential building blocks for Security Service Providers and enterprises to automate the use of different NSFs from multiple vendors by their security management entities. I2NSF may be invoked by any (authorized) client. Examples of authorized clients are upstream applications (controllers), orchestration systems, and security portals.

4.1. Basic Framework

Users request security services through specific clients (e.g., a customer application, the NSP BSS/OSS or management platform) and the appropriate NSP network entity will invoke the (v)NSFs according to the user service request. This network entity is denoted as the security controller in this document. The interaction between the entities discussed above (client, security controller, NSF) is shown in Figure 2:

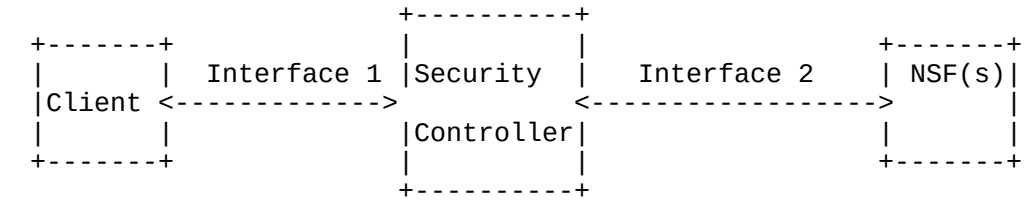


Figure 2: Interaction between Entities

Interface 1 is used for receiving security requirements from client and translating them into commands that NSFs can understand and execute. The security controller also passes back NSF security reports (e.g., statistics) to the client which the control has gathered from NSFs. Interface 2 is used for interacting with NSFs according to commands, and collect status information about NSFs.

Client devices or applications can require the security controller to add, delete or update rules in the security service function for their specific traffic.

When users want to get the executing status of a security service, they can request NSF status from the client. The security controller will collect NSF information through Interface 2, consolidate them, and give feedback to client through Interface 1. This interface can be used to collect not only individual service information, but also aggregated data suitable for tasks like infrastructure security assessment.

Customers may require validating NSF availability, provenance, and correct execution. This validation process, especially relevant for vNSFs, includes at least:

Integrity of the NSF: by ensuring that the NSF is not compromised;

Isolation: by ensuring the execution of the NSF is self-contained for privacy requirements in multi-tenancy scenarios.

Provenance of NSF: Customers may need to be provided with strict guarantees about the origin of the NSF, its status (e.g. available, idle, down, and others), and feedback mechanisms so that a customer may be able to check that a given NSF or set of NSFs properly conform to the the customer’s requirements and subsequent configuration tasks.

In order to achieve this, the security controller may collect security measurements and share them with an independent and trusted third party (via interface 1) in order to allow for attestation of NSF functions using the third party added information.

4.2. Access Networks

This scenario describes use cases for users (e.g. enterprise user, network administrator, and residential user) that request and manage security services hosted in the network service provider (NSP) infrastructure. Given that NSP customers are essentially users of their access networks, the scenario is essentially associated with their characteristics, as well as with the use of vNSFs.

The Virtual CPE described in [NFVUC] use cases #5 and #7 requires a model of access virtualization that includes mobile and residential access where the operator may offload security services from the customer local environment (e.g., device or terminal) to its own infrastructure.

These use cases define the interaction between the operator and the vNSFs through automated interfaces, typically by means of B2B communications.

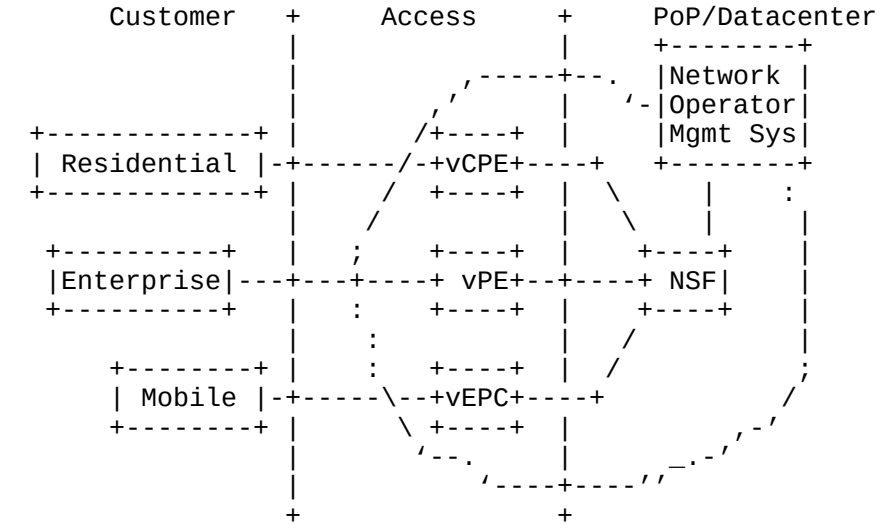


Figure 3: NSF and actors

Different access clients may have different service requests:

Residential: service requests for parental control, content management, and threat management.

Parental control requests may include identity based filters for web content or usage. Content management may include identifying and blocking malicious activities from web contents, mail, or files downloaded. Threat management may include identifying and blocking botnets or malware.

Enterprise: service requests for enterprise flow security policies and managed security services

Flow security policies include access (or isolation) to data from various internal groups, access (or isolation) from varous web sites or social media applications, and encryption on data transferred between corporates sites (main office, enterprise branch offices, and remote campuses). Managed security services may include detection and mitigation of external and internal threats. External threats can include application or phishing attacks, malware, botnet, DDoS, and others. Internal threats (aka lateral threats) can include detecting programs moving from one enterprise site to another without permission.

Service Provider: Service requests for policies that protect service providers networks against various threats (including DDoS, botnets and malware). Such policies are meant to securely and reliably deliver contents (e.g., data, voice, video) to various customers, including residential, mobile and corporate customers. These security policies are also enforced to guarantee isolation between multiple tenants, regardless of the nature of the corresponding connectivity services.

Mobile: service requests from interfaces which monitor and ensure user quality of experience, content management, parental controls, and external threat management.

Content management for the mobile device includes identifying and blocking malicious activities from web contents, mail, files. Threat management for infrastructure includes detecting and removing malicious programs such as Botnet, DDoS, and Malware.

Some access customers may not care about which NSFs are utilized to achieve the services they requested. In this case, provider network orchestration systems can internally select the NSFs (or vNSFs) to enforce the policies requested by the clients. Other access customers, especially some enterprise customers, may want to get their dedicated NSFs (most likely vNSFs) for direct control purposes. In this case, here are the steps to associate vNSFs to specific customers:

vNSF Deployment: The deployment process consists in instantiating a NSF on a Virtualization Infrastructure (NFVI), within the NSP administrative domain(s) or with other external domain(s). This is a required step before a customer can subscribe to a security service supported in the vNSF.

vNSF Customer Provisioning: Once a vNSF is deployed, any customer can subscribe to it. The provisioning lifecycle includes the following:

* Customer enrollment and cancellation of the subscription to a vNSF;

* Configuration of the vNSF, based on specific configurations, or derived from common security policies defined by the NSP.

* Retrieve and list the vNSF functionalities, extracted from a manifest or a descriptor. The NSP management systems can demand this information to offer detailed information through the commercial channels to the customer.

4.3. Cloud Data Center Scenario

In a data center, network security mechanisms such as firewalls may need to be dynamically added or removed for a number of reasons. These changes may be explicitly requested by the user, or triggered by a pre-agreed upon Service Level Agreement (SLA) between the user and the provider of the service. For example, the service provider may be required to add more firewall capacity within a set timeframe whenever the bandwidth utilization hits a certain threshold for a specified period. This capacity expansion could result in adding new instances of firewalls on existing machines or provisioning a completely new firewall instance in a different machine.

The on-demand, dynamic nature of security service delivery essentially encourages that the network security "devices" be in software or virtual form factors, rather than in a physical appliance form. This requirement is a provider-side concern. Users of the firewall service are agnostic (as they should) as to whether or not the firewall service is run on a VM or any other form factor. Indeed, they may not even be aware that their traffic traverses firewalls.

Furthermore, new firewall instances need to be placed in the "right zone" (domain). The issue applies not only to multi-tenant environments where getting the tenant in the right domain is of paramount importance, but also in environments owned and operated by a single organization with its own service segregation policies. For example, an enterprise may mandate that firewalls serving Internet traffic and Business-to-Business (B2B) traffic be separated. Another example is that IPS/IDS services for investment banking and non-banking traffic may be separated for regulatory reasons.

4.3.1. On-Demand Virtual Firewall Deployment

A service provider-operated cloud data center could serve tens of thousands of clients. Clients’ compute servers are typically hosted on virtual machines (VMs), which could be deployed across different

server racks located in different parts of the data center. It is often not technically and/or financially feasible to deploy dedicated physical firewalls to suit each client's security policy requirements, which can be numerous. What is needed is the ability to dynamically deploy virtual firewalls for each client's set of servers based on established security policies and underlying network topologies.

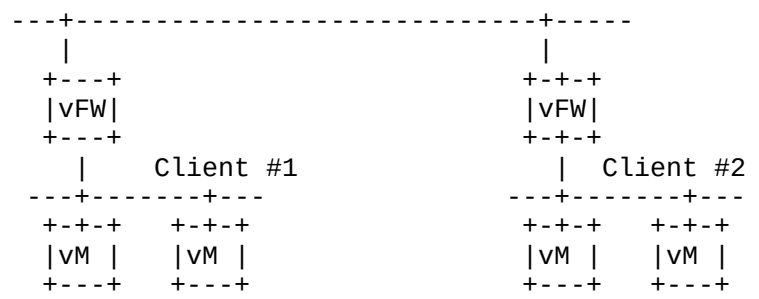


Figure 4: NSF in Data Centers

4.3.2. Firewall Policy Deployment Automation

Firewall rule setting is often a time consuming, complex and error-prone process even within a single organization/enterprise framework. It becomes far more complex in provider-owned cloud networks that serve myriads of customers.

Firewall rules today are highly tied with ports and addresses that identify traffic. This makes it very difficult for clients of cloud data centers to construct rules for their own traffic as the clients only see the virtual networks and the virtual addresses. The customer-visible virtual networks and addresses may be different from the actual packets traversing the FWs.

Even though most vendors support similar firewall features, the actual rule configuration keywords are different from vendors to vendors, making it difficult for automation. Automation works best when it can leverage a common set of standards that will work across NSFs by multiple vendors. Without automation, it is virtually impossible for clients to dynamically specify their desired rules for their traffic.

4.3.3. Client-Specific Security Policy in Cloud VPNs

Clients of service provider-operated cloud data centers need not only to secure Virtual Private Networks (VPNs) but also virtual security functions that apply the clients' security policies. The security policies may govern communication within the clients' own virtual networks as well as communication with external networks. For example, VPN service providers may need to provide firewall and other security services to their VPN clients. Today, it is generally not possible for clients to dynamically view (let alone change) what, where and how security policies are implemented on their provider-operated clouds. Indeed, no standards-based framework exists to allow clients to retrieve/manage security policies in a consistent manner across different providers.

4.3.4. Internal Network Monitoring

There are many types of internal traffic monitors that may be managed by a security controller. This includes a new class of services referred to as Data Loss Prevention (DLP), or Reputation Protection Services (RPS). Depending on the class of event, alerts may go to internal administrators, or external services.

4.4. I2NSF Preventing Distributed DoS in Overlay Networks

In the internet where everything is connected, preventing unwanted traffic that may cause Denial of Service (DoS) attack or distributed DoS (DDoS) has become a challenge. One place where DDoS can be challenging to prevent or mitigate is in overlay networks. Many networks such as Internet of Things (IoT) networks, Information-Centric Networks (ICN), Content Delivery Networks (CDN), and cloud networks use overlay networks within their paths (or links). The underlay networks that support overlay networks can be attacked by DDoS, thereby saturating access links or links within the network. DoS or DDoS attacks on the access links may also cause the overlay nodes' CPUs or links to be saturated by DoS or DDoS traffic which will prevent these links from being used by legitimate overlay traffic. Overlay security solutions do not address underlay security threats so there is a need for a distributed solution to prevent DDoS attacks from spreading throughout overlay and underlay networks. Such solution may for example rely upon the dynamic, highly-reactive, enforcement of security filtering policies network-wise.

Similar to traditional networks placing a firewall or Intrusion Prevention System (IPS) on the wire to enforce traffic rules, the interface to network security functions (I2NSF) can be used by overlay networks to request underlay networks enforce certain flow-based security rules. Using this mechanism, the overlay network can

coordinate with the underlay network to remove unwanted traffic including DoS and DDoS in the underlay network.

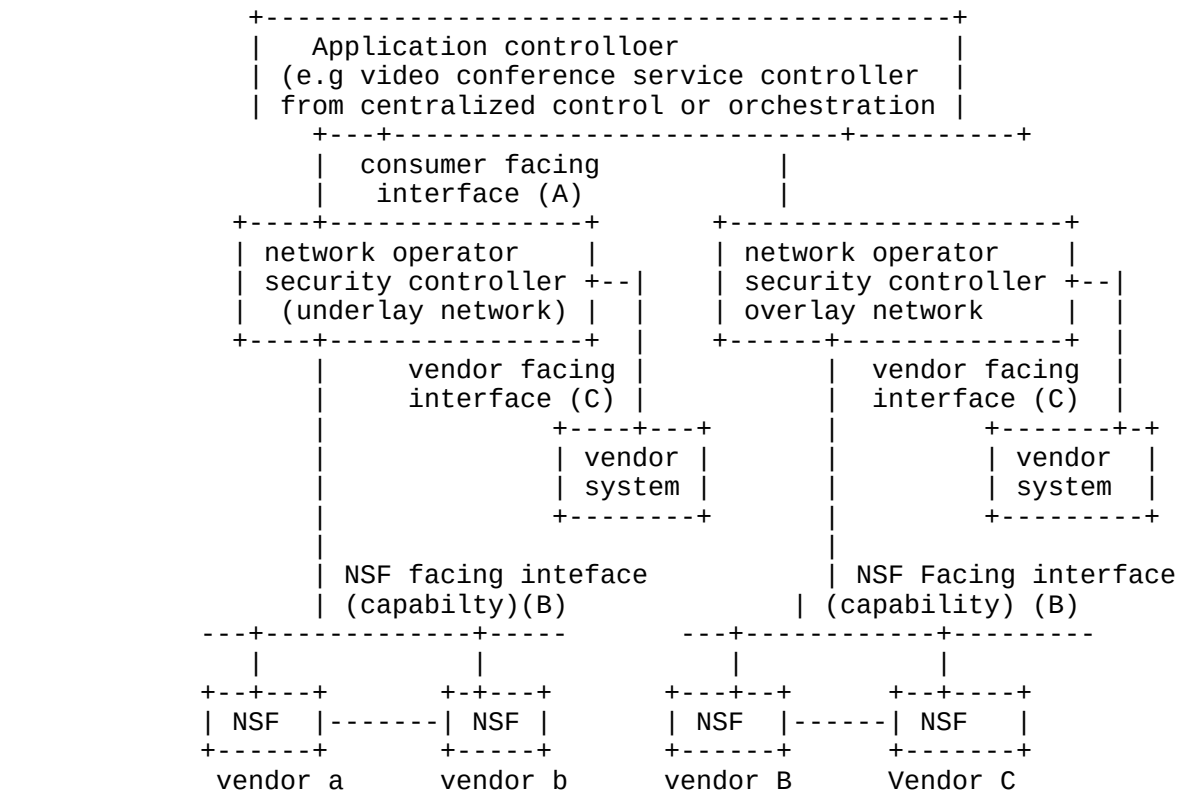


Figure 5: I2NSF Preventing DDoS Attacks in Overlay Networks.

5. Management Considerations

Management of NSFs usually include the following:

- o Lifecycle managment and resource management of NSFs,
- o Device configuration, such as address configuration, device internal attributes configuration, etc.;
- o Signaling, and
- o Policy rule provisioning.

I2NSF will only focus on the policy provisioning part of NSF management.

6. IANA Considerations

No IANA considerations exist for this document.

7. Security Considerations

Having a secure access to control and monitor NSFs is crucial for hosted security services. An I2NSF security controller raises new security threats. It needs to be resilient to attacks and quickly recover from attacks. Therefore, proper secure communication channels have to be carefully specified for carrying controlling and monitoring traffic between the NSFs and their management entity (or entities).

In addition, the Flow security policies specified by customers can conflict with providers' internal security policies which may allow unauthorized traffic or unauthorized changes to flow polices (e.g. customers changing flow policies that do not belong to them). Therefore, it is crucial to have proper AAA to authorize access to the network and access to the I2NSF management stream.

8. Contributors

I2NSF is a group effort. The following people actively contributed to the initial use case text: Xiaojun Zhuang (China Mobile), Sumandra Majee (F5), Ed Lopez (Fortinet), and Robert Moskowitz (Huawei).

9. Contributing Authors

I2NSF has had a number of contributing authors. The following are contributing authors:

- Antonio Pastur (Telefonica I+D),
- Mohamed Boucadair (France Telecom),
- Michael Georgiades (Prime Tel),
- Minpeng Qi (China Mobile),
- Shaibal Chakrabarty (US Ignite),
- Nic Leymann (Deutsche Telekom),
- Rakesh Kumar (Juniper),
- Anil Lohiya (Juniper),

- David Qi (Bloomberg), and
- Xiaobo Long.

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>.

10.2. Informative References

[Gartner-2013]
Messmer, E., "Gartner: Cloud-based security as a service set to take off", October 2013.

[I-D.hares-i2nsf-gap-analysis]
Hares, S., Zhang, D., Moskowitz, R., and H. Rafiee, "Analysis of Existing work for I2NSF", draft-hares-i2nsf-gap-analysis-01 (work in progress), December 2015.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-06 (work in progress), December 2015.

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

[I-D.jeong-i2nsf-sdn-security-services]
Jeong, J., Kim, H., and P. Jung-Soo, "Requirements for Security Services based on Software-Defined Networking", draft-jeong-i2nsf-sdn-security-services-01 (work in progress), March 2015.

[I-D.lopez-i2nsf-packet]
Ed, E., "Packet-Based Paradigm For Interfaces To NSFs", draft-lopez-i2nsf-packet-00 (work in progress), March 2015.

[I-D.pastor-i2nsf-access-usecases]
Pastor, A. and D. Lopez, "Access Use Cases for an Open OAM Interface to Virtualized Security Services", draft-pastor-i2nsf-access-usecases-00 (work in progress), October 2014.

[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.

[I-D.qi-i2nsf-access-network-usecase]
Wang, K. and X. Zhuang, "Integrated Security with Access Network Use Case", draft-qi-i2nsf-access-network-usecase-02 (work in progress), March 2015.

[I-D.zarny-i2nsf-data-center-use-cases]
Zarny, M., Leymann, N., and L. Dunbar, "I2NSF Data Center Use Cases", draft-zarny-i2nsf-data-center-use-cases-00 (work in progress), October 2014.

[I-D.zhou-i2nsf-capability-interface-monitoring]
Zhou, C., Xia, L., Boucadair, M., and J. Xiong, "The Capability Interface for Monitoring Network Security Functions (NSF) in I2NSF", draft-zhou-i2nsf-capability-interface-monitoring-00 (work in progress), October 2015.

[RFC4948] Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, DOI 10.17487/RFC4948, August 2007, <http://www.rfc-editor.org/info/rfc4948>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Linda Dunbar
Huawei
5340 Legacy Drive, Suite 175
Plano, TX 75024
USA

Phone: +1-734-604-0332
Email: ldunbar@huawei.com

Diego R. Lopex
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.com

Myo Zarny
Goldman Sachs
30 Hudson Street
Jersey City, NJ 07302
USA

Email: myo.zarny@gs.com

Christian Jacquenet
France Telecom
Rennes, 35000
France

Email: Christian.jacquenet@orange.com

I2NSF
Internet-Draft
Intended status: Informational
Expires: April 25, 2017

S. Hares
J. Strassner
Huawei
D. Lopez
Telefonica I+D
L. Xia
Huawei
H. Birkholz
Fraunhofer SIT
October 23, 2016

Interface to Network Security Functions (I2NSF) Terminology
draft-ietf-i2nsf-terminology-02.txt

Abstract

This document defines a set of terms that are used for the Interface to Network Security Functions (I2NSF) effort.

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 April 25, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document defines the terminology for the Interface to Network Security Functions (I2NSF) effort. This section provides some background on I2NSF; a detailed problem statement can be found in [I-D.ietf-i2nsf-problem-and-use-cases]. Motivation and comparison to previous work can be found in [I-D.ietf-i2nsf-gap-analysis].

Enterprises are now considering using network security functions (NSFs) hosted by service providers due to the growing challenges and complexity in maintaining an up-to-date secure infrastructure that complies with regulatory requirements, while controlling costs. The hosted security service is especially attractive to small- and medium-size enterprises who suffer from a lack of security experts to continuously monitor, acquire new skills and propose immediate mitigations to ever increasing sets of security attacks. Small- and medium-sized businesses (SMBs) are increasingly adopting cloud-based security services to replace on-premises security tools, while larger enterprises are deploying a mix of traditional (hosted) and cloud-based security services.

To meet the demand, more and more service providers are providing hosted security solutions to deliver cost-effective managed security services to enterprise customers. The hosted security services are primarily targeted at enterprises, but could also be provided to mass-market customers as well. NSFs are provided and consumed in increasingly diverse environments. Users of NSFs may consume network security services hosted by one or more providers, which may be their own enterprise, service providers, or a combination of both.

It is out of scope in this document to define an exhaustive list of terms that are used in the security field; the reader is referred to other applicable documents, such as [RFC4949].

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 [RFC2119] significance.

3. Terminology

AAA: Authentication, Authorization, and Accounting. See individual definitions.

Abstraction: The definition of the salient characteristics and behavior of an object that distinguish it from all other types of objects. It manages complexity by exposing common properties between objects and processes while hiding detail that is not relevant.

Access Control: Protection of system resources against unauthorized access; a process by which use of system resources is regulated according to a security policy, and is permitted by only authorized entities (users, programs, processes, or other systems) according to that policy [RFC4949].

Accounting: The act of collecting information on resource usage for the purpose of trend analysis, auditing, billing, or cost allocation ([RFC2975] [RFC3539]).

ACL (Access Control List): This is a mechanism that implements access control for a system resource by enumerating the system entities that are permitted to access the resource and stating, either implicitly or explicitly, the access modes granted to each entity [RFC4949]. A YANG description is defined in [I-D.ietf-netmod-acl-model].

Action: Defines what is to be done when a set of Conditions are met (See I2NSF Action). (from [I-D.ietf-supa-generic-policy-info-model]).

Assertion: Defined by the ITU in [X.1252] as "a statement made by an entity without accompanying evidence of its validity". In the context of I2NSF, an assertion MAY include metadata about all or part of the assertion (e.g., context of the assertion, or about timestamp indicating the point in time the assertion was created). The validity of an assertion cannot be verified. (from [I-D.ietf-sacm-terminology]).

Authentication: Defined in [RFC4949] as "the process of verifying a claim that a system entity or system resource has a certain attribute value." (from [I-D.ietf-sacm-terminology]).

Authorization: Defined in [RFC4949] as "an approval that is granted to a system entity to access a system resource." (from [I-D.ietf-sacm-terminology]).

B2B: Business-to-Business.

Bespoke: Something made to fit a particular person, customer, or company.

Bespoke security management: Security management systems that are make to fit a particular customer.

Boolean Clause: A logical statement that evaluates to either TRUE or FALSE. Also called Boolean Expression.

Capability: Defines a set of features that are available from a managed entity (see also I2NSF Capability). Examples of "managed entities" are NSFs and Controllers, where NSF Capabilities and Controller Capabilities define functionality of an NSF and about Controller, respectively. These functions may, but do not have to, be used. All Capabilities are announced through the Registration Interface.

Client: See Consumer. [Editor’s note: placeholder for gradually replacing Client with Consumer, since Client is too vague and has other connotations (e.g., client-server)].

Client-Facing Interface: See Consumer-Facing Interface. See also: Interface, NSF-Facing Interface.

Component: An encapsulation of software that communicates using Interfaces. A Component may be implemented by hardware and/or software, and be represented using a set of classes. In general, a Component encapsulates a set of data structures and a set of algorithms that implement the function(s) that it provides.

Consumer: A Consumer is a Role that is assigned to an I2NSF Component that represents the needs of a user of I2NSF services. A consumer can send/receive information to/from another I2NSF Component (e.g., for defining and monitoring security policies for the Consumer’s specific flows through an I2NSF administrative domain). See also: Producer, Role.

Consumer-Facing Interface: An Interface dedicated to communication with Consumers of NSF Data and Services. This is typically defined per I2NSF administrative domain. See also: Interface, NSF-Facing Interface.

Condition: A set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to make a decision. A Condition, when used in the context of a Policy Rule, is used to determine whether or not the set of Actions in that Policy Rule can be executed or not. Examples of an I2NSF Condition include matching attributes of a packet or flow, and comparing the internal state of a NSF to a desired state. (from [I-D.ietf-supa-generic-policy-info-model]).

Constraint: A Constraint is a limitation or restriction. Constraints may be associated with any type of object (e.g., Events, Conditions, and Actions in Policy Rules).

Constraint Programming: A type of programming that uses constraints to define relations between variables in order to find a feasible (and not necessarily optimal) solution.

Context: The Context of an Entity is a collection of measured and/or inferred knowledge that describe the state and the environment in which an Entity exists or has existed. (from http://www.ietf.org/mail-archive/web/i2nsf/current/msg00762.html).

Controller: A Controller is a management Component that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. This definition is based on that in [I-D.ietf-sacm-terminology].

Control Plane: In the context of I2NSF, the Control Plane is an architectural Component that provides common control functions to all I2NSF Components, including some or all of the following: authentication, authorization, accounting, auditing, and Capability discovery and negotiation. The Control Plane orchestrates the operation of the Data Plane according to guidance and/or input from the Management Plane. I2NSF Components with Interfaces to the Control Plane have knowledge of the Capabilities of other I2NSF Components within a particular I2NSF administrative domain. This definition is based on that in [I-D.ietf-sacm-terminology]. See also: Data Plane, Management Plane.

Customer: A business role of an entity that is involved in the definition and/or consumption of services, and the possible negotiation of a contract to use services from a Provider.

DC: Data Center

Data Model: 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 (typically one or more of these). Note the difference between a data **model** and a data **structure**. [I-D.ietf-supa-generic-policy-info-model].

Data Plane: In the context of I2NSF, the Data Plane is an architectural Component that provides operational functions to enable an I2NSF Component to provide and consume packets and flows. See also: Control Plane, Management Plane.

Data Structure: A low-level building block that is used in programming to implement an algorithm. A data model typically contains multiple types of data structures; however, a data structure does not contain a data model. Note the difference between a data **model** and a data **structure**.

Event: An important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. Examples of an I2NSF Event include time and user actions (e.g. logon, logoff, and actions that violate an ACL). An Event, when used in the context of a Policy Rule, is used to determine whether the Condition clause of an imperative Policy Rule can be evaluated or not (from [I-D.ietf-supa-generic-policy-info-model]).

ECA: Event - Condition - Action (a type of Policy Rule).

Firewall (FW): A function that restricts data communication traffic to and from one of the connected networks (the one said to be ’inside’ the firewall), and thus protects that network’s system resources against threats from the other network (the one that is said to be ’outside’ the firewall) [RFC4949]. [I-D.ietf-opsawg-firewalls]

Flow: A set of information (e.g., packets) that are related in a fundamental manner (e.g., sent from the same source and sent to the same destination). A common example is a sequence of packets. It is the opposite of packet-based, which treats each packet discretely (e.g., each packet is assessed individually to determine the action(s) to be taken).

Flow-based NSF: A NSF that inspects network flows according to a set of policies intended for enforcing security properties. Flow-based security also means that packets are inspected in the order they are received, and without modification to the packet due to the inspection process.

I2NSF Agent: A software Component in a device that implements an NSF. It receives provisioning information and requests for operational data (e.g., monitoring data) from an I2NSF Consumer. It is also responsible for enforcing the policies that it receives from an I2NSF Consumer.

I2NSF Action: An I2NSF Action is a special type of Action that is used to control and monitor aspects of flow-based Network Security Functions. 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 Action, when used in the context of a I2NSF Policy Rule, may be executed when both the Event and the Condition clauses of its owning I2NSF Policy Rule evaluate to true. The execution of this Action may be influenced by applicable metadata. (from [I-D.ietf-supa-generic-policy-info-model]).

I2NSF Capability: A set of features that are available from an NSF Server or an NSF Controller. While both are Capabilities, the former defines functions that are available from an NSF, whereas the latter defines functions that are available from a security Controller or other Management Entity. This definition is based on that in [I-D.ietf-sacm-terminology].

I2NSF Client: See I2NSF Consumer.

I2NSF Component: A Component that provides one or more I2NSF Services. I2NSF Components are managed and communicate with other I2NSF Components using I2NSF Interfaces.

I2NSF Consumer: A software Component that uses the I2NSF framework to read, write, and/or change provisioning and operational aspects of the NSFs that it attaches to.

I2NSF Consumer Interface: An Interface dedicated to requesting and using I2NSF Services. For example, this Interface could be used to request a set of Flow Security policies from an I2NSF Controller or from one or more individual NSFs. The difference is that the former uses more abstract Condition matching (e.g., based on tenant or customer ID), whereas the latter uses more low-level Condition matching (e.g., based on flow state or fields in a flow or packet). See also: Interface, I2NSF Provider Interface, Client-Facing Interface, NSF-Facing Interface.

I2NSF Management System: I2NSF Consumers operate within the scope of a network management system, which serves as a collection and distribution point for I2NSF security provisioning.

I2NSF Policy: A set of Policy Rules that are used to manage and control the changing or maintaining of the state of an instance of an NSF.

I2NSF Policy Rule: A Policy Rule that is adapted for I2NSF usage. The I2NSF Policy Rule is assumed to be in ECA form (i.e., an imperative structure). Other types of programming paradigms (e.g., declarative and functional) are currently out of scope. An example of an I2NSF Policy Rule is, in pseudo-code:

```
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**.

I2NSF Provider Interface: An Interface dedicated to providing I2NSF Services. For example, this could provide Anti-Virus, (D)DoS, or IPS Services. See also: Interface, I2NSF Provider Interface, Client-Facing Interface, NSF-Facing Interface.

I2NSF Registry: A registry that contains I2NSF capability information, which can be controlled by the I2NSF Management System. See also: Registry.

I2NSF Service: A set of functions, provided by an I2NSF Consumer, which are used by zero or more I2NSF Producers. Exemplary I2NSF Services include Anti-Virus, Authentication, Authorization, (D)DoS, Firewall, and IPS Services. See also: Interface, I2NSF Provider Interface, Client-Facing Interface, NSF-Facing Interface.

IDS: Intrusion Detection System (see below).

IPS: Intrusion Protection System (see below).

Information Model: 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 [I-D.ietf-supa-generic-policy-info-model].

Interface: A set of operations one object knows it can invoke on, and expose to, another object. It is a subset of all operations that a given object implements. The same object may have multiple types of interfaces to serve different purposes. An example of multiple interfaces can be seen by considering the interfaces include a firewall uses; these include:

* multiple interfaces for data packets to traverse through,
* an interface for a controller to impose policy, or retrieve the results of execution of a policy rule.

See also: Consumer Interface, I2NSF Interface, Provider Interface

Interface Group: A set of Interfaces that are related in purpose and which share the same communication mechanisms.

Intrusion Detection System (IDS): A system that detects network intrusions via a variety of filters, monitors, and/or probes. An IDS may be stateful or stateless.

Intrusion Protection System (IPS): A system that protects against network intrusions. An IPS may be stateful or stateless.

Management Plane: In the context of I2NSF, the Management Plane is an architectural Component that provides common functions to define the behavior of I2NSF Components. The primary use of the Management Plane is to transport behavioral commands, and supply OAM data, for making decisions that affect behavior. Examples include modifying the configuration of an I2NSF Component and transporting OAM data. See also: Control Plane, Data Plane.

Metadata: Data that provides information about other data. Examples include IETF network management protocols (e.g. NETCONF, RESTCONF, IPFIX) or IETF routing interfaces (I2RS). The I2NSF security interface may utilize Metadata to describe and/or prescribe characteristics and behavior of the YANG data models.

Middlebox: Any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host [RFC3234].

Network Security Function (NSF): Software that provides a set of security-related services. Examples include detecting unwanted activity and blocking or mitigating the effect of such unwanted activity in order to fulfil service requirements. The NSF can also help in supporting communication stream integrity and confidentiality.

NSF-Facing Interface: An Interface dedicated to communication with a set of NSFs. This is typically defined per I2NSF administrative domain. See also: Interface, Consumer-Facing Interface.

OAM: Operation, Administrative, and Management.

OCL (Object Constraint Language): A constraint programming language that is used to specify constraints (e.g., in UML) (from http://www.ietf.org/mail-archive/web/i2nsf/current/msg00762.html)

Policy Rule: A set of rules that are used to manage and control the changing or maintaining of the state of one or more managed objects. Often this is shortened to Rule or Policy (see I2NSF policy rule). (from [I-D.ietf-supa-generic-policy-info-model]).

Profile: A structured representation of information that uses a pre-defined set of capabilities of an object, typically in a specific context. Zero or more Capabilities may be changed at runtime. This may be used to simplify how this object interacts with other objects in its environment.

Producer: A Producer is a Role that is assigned to an I2NSF Component that can send information and/or commands to another I2NSF Component. See also: Consumer, Role.

Registry: A logically centralized location containing data of a particular type; it may optionally contain metadata, relationships, and other aspects of the registered data in order to use those data effectively. An I2NSF registry is used to contain capability information that can be controlled by the controller.

Registration Interface: An interface dedicated to requesting, receiving, editing, and deleting information in a Registry.

Role: An abstraction of a Component that models context-specific views and responsibilities of an object as separate Role objects. Role objects can optionally be attached to, and removed from, the object that the Role object describes at runtime. This provides three important benefits. First, it enables different behavior to be supported by the same Component for different contexts. Second, it enables the behavior of a Component to be adjusted dynamically (i.e., at runtime, in response to changes in context) by using one or more Roles to define the behavior desired for each context. Third, it decouples the Roles of a Component from the Applications use that Component.

Service Interface: An Interface dedicated to enabling Policy Rules to be managed. This is also called the I2NSF Consumer Interface.

Service Provider Security Controller: TBD (Editorial: Place holder for a split between controller and security controller definitions.)

Tenant: A group of users that share common access privileges to the same software. An I2NSF tenant may be physical or virtual, and may run on a variety of systems or servers.

Vendor-Facing Interface: An Interface dedicated to registering and vendor-specific NSFs and Capabilities. It is also used to invoke vendor-specific functionality. This is also called the NSF-Facing Interface.

3. IANA Considerations

No IANA considerations exist for this document.

4. Security Considerations

This is a terminology document with no security considerations.

5. Contributors

The following people contributed to creating this document, and are listed in alphabetical order:

Henk Birkholz

6. References

6.1. Informative References

[I-D.ietf-i2nsf-gap-analysis]
Hares, S., Moskowitz, R., and Zhang, D., "Analysis of Existing work for I2NSF", draft-ietf-i2nsf-gap-analysis-02 (work in progress), July 2016.

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-02 (work in progress), October 2016.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Sreenivasa, K., Huang, L., Blair, D., "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-09 (work in progress), October 2016.

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

[I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., Cam-Wignet, N., "Secure Automation and Continuous Monitoring (SACM) Terminology", draft-ietf-sacm-terminology-11, September 2016

[I-D.ietf-supa-generic-policy-info-model]
Strassner, J., Halpern, J., and J. Coleman, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-01 (work in progress), July 2016.

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

[RFC2975] Aboba, B., Arkko, J., and D. Harrington, "Introduction to Accounting Management", RFC 2975, DOI 10.17487/RFC2975, October 2000, <http://www.rfc-editor.org/info/rfc2975>.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <http://www.rfc-editor.org/info/rfc3539>.

[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>.

[X.1252] ITU-T, "Baseline identity management terms and definitions", Recommendation ITU-T X.1252, April 2510

Authors' Addresses

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

John Strassner
Huawei Technologies
Santa Clara, CA USA 95050
Email: john.sc.strassner@huawei.com

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain
Email: diego.r.lopez@telefonica.com

Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing , Jiangsu 210012
China
Email: Frank.Xialiang@huawei.com

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany
Email: henk.birkholz@sit.fraunhofer.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2017

J. Jeong
H. Kim
Sungkyunkwan University
J. Park
ETRI
T. Ahn
S. Lee
Korea Telecom

July 5, 2016

Software-Defined Networking Based Security Services using Interface to Network Security Functions
draft-jeong-i2nsf-sdn-security-services-05

Abstract

This document describes a framework, objectives, requirements, and use cases for security services based on Software-Defined Networking (SDN) using a common Interface to Network Security Functions (I2NSF). It first proposes the framework of SDN-based security services in the I2NSF framework. It then explains three use cases, such as a centralized firewall system, centralized DDoS-attack mitigation system, and centralized VoIP/VoLTE security system.

Table of Contents

1. Introduction

Software-Defined Networking (SDN) is a set of techniques that enables users to directly program, orchestrate, control and manage network resources through software (e.g., SDN applications). It relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller uses interfaces to arbitrate the control of network resources in a logically centralized manner. It also manages and configures the distributed network resources, and provides the abstracted view of the network resources to the SDN applications. The SDN applications can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via the interfaces [RFC7149][ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

Due to the increase of sophisticated network attacks, the legacy security services become difficult to cope with such network attacks in an autonomous manner. SDN has been introduced to make networks more controllable and manageable, and this SDN technology will be promising to autonomously deal with such network attacks in a prompt manner.

This document describes a framework, objectives and requirements to support the protection of network resources through SDN-based security services using a common interface to Network Security Functions (NSF) [i2nsf-framework]. It uses an interface to NSF (I2NSF) for such SDN-based security services that are performed in virtual machines through network functions virtualization [ETSI-NFV].

This document addresses the challenges of the exisiting systems for security services. As feasible solutions to handle these challenges, this document proposes three use cases of the security services, such as a centralized firewall system, centralized DDoS-attack mitigation system, and centralized VoIP/VoLTE security system.

For the centralized firewall system, this document raises limitations in the legacy firewalls in terms of flexibility and administration costs. Since in many cases, access control management for firewall is manually performed, it is difficult to add the access control policy rules corresponding to new network attacks in a prompt and autonomous manner. Thus, this situation requires expensive administration costs. This document introduces a use case of SDN-based firewall system to overcome these limitations.

For the centralized DDoS-attack mitigation system, this document raises limitations in the legacy DDoS-attack mitigation techniques in terms of flexibility and administration costs. Since in many cases, network configuration for the mitigation is manually performed, it is

difficult to dynamically configure network devices to limit and control suspicious network traffic for DDoS attacks. This document introduces a use case of SDN-based DDoS-attack mitigation system to provide an autonomous and prompt configuration for suspicious network traffic.

For the centralized VoIP/VoLTE security system, this documents raises challenges in the legacy VoIP/VoLTE security system in terms of provisioning time, the granularity of security, cost, and the establishment of policy. This document shows a use case of SDN-based VoIP/VoLTE security system to resolve these challenges along in the I2NSF framework.

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 [RFC7149], [ITU-T.Y.3300], [ONF-OpenFlow], [ONF-SDN-Architecture], [ITU-T.X.1252], and [ITU-T.X.800]. In addition, the following terms are defined below:

- Software-Defined Networking: A set of techniques that enables to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

- Access Control: A procedure used to determine if an entity should be granted access to resources, facilities, services, or information based on pre-established rules and specific rights or authority associated with the requesting party [ITU-T.X.1252].

- Access Control Policy: The set of rules that define the conditions under which access may take place [ITU-T.X.800].

- Access Control Policy Rules: Security policy rules concerning the provision of the access control service [ITU-T.X.800].

- Network Resources: Network devices that can perform packet forwarding in a network system. The network resources include network switch, router, gateway, WiFi access points, and similar devices.

- Firewall: A firewall that is a device or service at the junction of two network segments that inspects every packet that attempts to cross the boundary. It also rejects any packet that does not satisfy certain criteria for disallowed port numbers or IP addresses.

- Centralized Firewall System: A centralized firewall that can establish and distribute access control policy rules into network resources for efficient firewall management. These rules can be managed dynamically by a centralized server for firewall. SDN can work as a network-based firewall system through a standard interface between firewall applications and network resources.

- Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation. These rules can be managed dynamically by a centralized server for DDoS-attack mitigation. SDN can work as a network-based mitigation system through a standard interface between DDoS-attack mitigation applications and network resources.

- Centralized VoIP/VoLTE Security System: A centralized security system that handles the security issues related to VoIP and VoLTE services. SDN can work as a network-based security system through a standard interface between VoIP/VoLTE security applications and network resources.

4. Overview

This section describes the referenced architecture to support SDN-based security services, such as centralized firewall system and centralized DDoS-attack mitigation system. Also, it describes a framework for SDN-based security services using I2NSF.

As shown in Figure 1, network security functions (NSFs) as security services (e.g., firewall, DDoS-attack mitigation, VoIP/VoLTE, web filter, and deep packet inspection) run on the top of SDN controller [ITU-T.Y.3300] [ONF-SDN-Architecture]. When an administrator enforces security policies for such security services through an application interface, SDN controller generates the corresponding access control policy rules to meet such security policies in an autonomous and prompt manner. According to the generated access control policy rules, the network resources such as switches take an action to mitigate network attacks, for example, dropping packets with suspicious patterns.

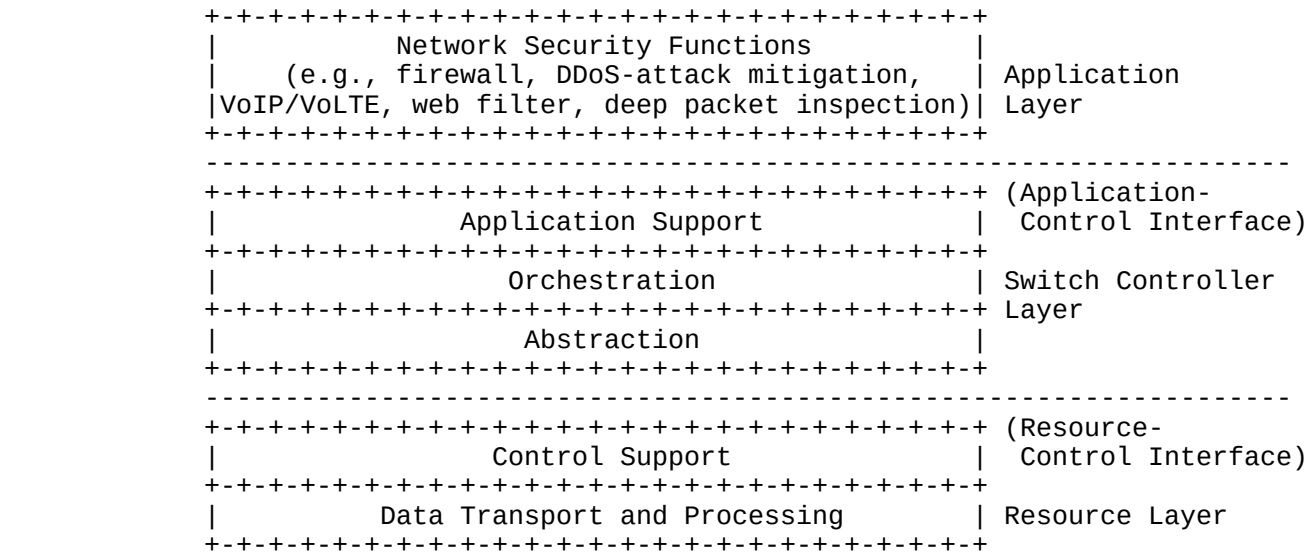


Figure 1: High-level Architecture for SDN-based Security Services

Figure 2 shows a framework to support SDN-based security services using I2NSF [i2nsf-framework]. As shown in Figure 2, I2NSF client can use security services by delivering their high-level security policies to security controller via client facing interface. Security controller asks NSFs to perform function-level security services via NSF facing interface. The NSFs run on top of virtual machines through Network Functions Virtualization (NFV) [ETSI-NFV]. NSFs ask switch controller to perform their required security services on switches under the supervision of switch controller. In addition, security controller uses registration interface to communicate with developer’s management system for registering (or deregistering) the developer’s NSFs into (or from) the NFV system using the I2NSF framework.

NSF facing interface between security controller and NSFs can be implemented by Network Configuration Protocol (NETCONF) [RFC6241] with a data modeling language called YANG [RFC6020] that describes function-level security services. A data model in [i2nsf-cap-interface-yang] can be used for the I2NSF capability interface, which is NSF facing interface.

The proposed framework of SDN-based security services can be combined to a security management architecture in [i2nsf-sec-mgnt-arch] for handling high-level security policies as well as low-level security policies.

Also, the proposed framework can enforce low-level security policies in NSFs by using a service function chaining (SFC) enabled I2NSF architecture in [i2nsf-sfc-enabled-arch].

5. Objectives

- Prompt reaction to new network attacks: SDN-based security services allow private networks to defend themselves against new sophisticated network attacks.

- Automatic defense from network attacks: SDN-based security services identify the category of network attack (e.g., malware and DDoS attacks) and take counteraction for the defense without the intervention of network administrators.

- Network-load-aware resource allocation: SDN-based security services measure the overhead of resources for security services and dynamically select resources considering load balance for the maximum network performance.

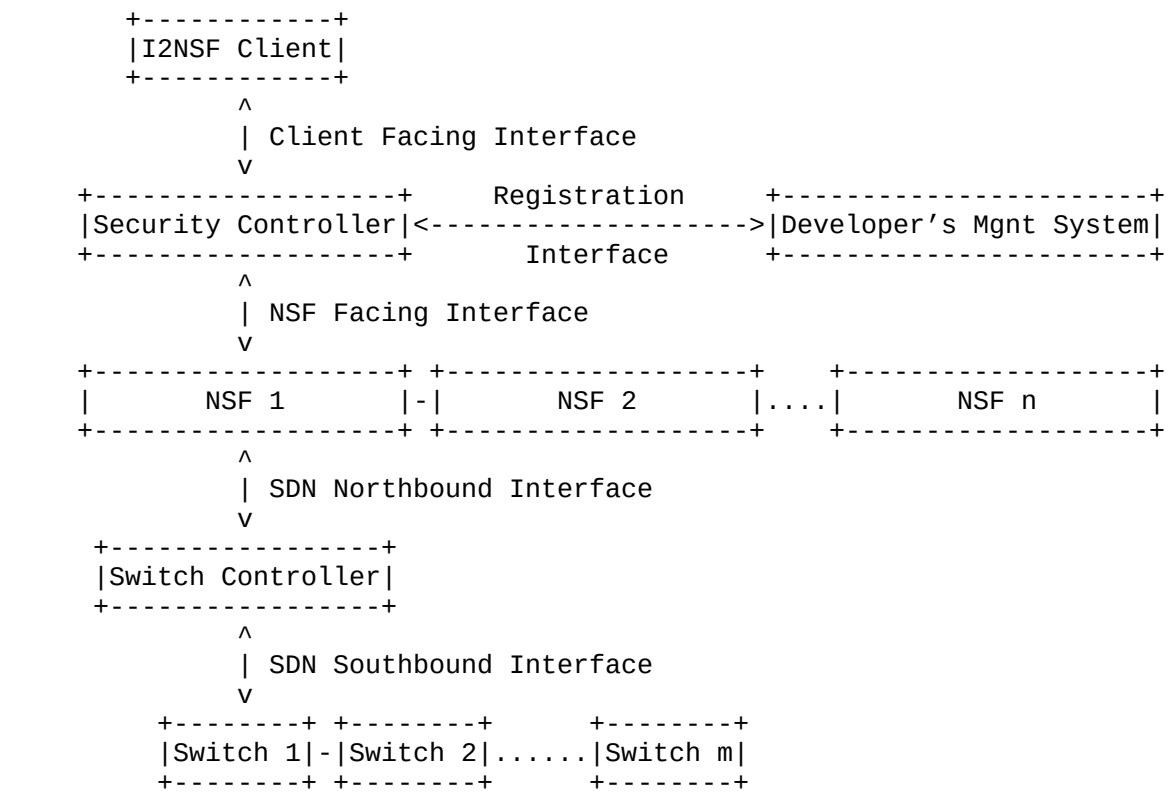


Figure 2: A Framework for SDN-based Security Services using I2NSF

6. Requirements

SDN-based security services provide dynamic and flexible network resource management to mitigate network attacks, such as malware and DDoS attacks. In order to support this capability, the requirements for SDN-based security services are described as follows:

- SDN-based security services are required to support the programmability of network resources to mitigate network attacks.
- SDN-based security services are required to support the orchestration of network resources and SDN applications to mitigate network attacks.
- SDN-based security services are required to provide an application interface allowing the management of access control policies in an autonomous and prompt manner.

- SDN-based security services are required to provide a resource-control interface for the control of network resources to mitigate network attacks.

- SDN-based security services are required to provide the logically centralized control of network resources to mitigate network attacks.

- SDN-based security services are required to support the seamless services to mitigate network attacks.

- SDN-based security services are required to provide the dynamic control of network resources to mitigate network attacks.

7. Use Cases

This section introduces three use cases for security services based on SDN: (i) centralized firewall system, (ii) centralized DDoS-attack mitigation system, and (iii) centralized VoIP/VoLTE security system.

7.1. Centralized Firewall System

For the centralized firewall system, a centralized network firewall can manage each network resource and firewall rules can be managed flexibly by a centralized server for firewall (called Firewall). The centralized network firewall controls each switch for the network resource management and the firewall rules can be added or deleted dynamically.

The procedure of firewall operations in the centralized firewall system is as follows:

1. Switch forwards an unknown flow’s packet to Switch Controller.

2. Switch Controller forwards the unknown flow’s packet to an appropriate security service application, such as Firewall.

3. Firewall analyzes the headers and contents of the packet.

4. If Firewall regards the packet as a malware’s packet with a suspicious pattern, it reports the malware’s packet to Switch Controller.

5. Switch Controller installs new rules (e.g., drop packets with the suspicious pattern) into switches.

6. The malware’s packets are dropped by switches.

For the above centralized firewall system, the existing SDN protocols can be used through standard interfaces between the firewall application and switches [RFC7149][ITU-T.Y.3300][ONF-OpenFlow] [ONF-SDN-Architecture].

Legacy firewalls have some challenges such as the expensive cost, performance, management of access control, establishment of policy, and packet-based access mechanism. The proposed framework can resolve these challenges through the above centralized firewall system based on SDN as follows:

- Cost: The cost of adding firewalls to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add firewall on each network resource. To solve this, each network resource can be managed centrally such that a single firewall is manipulated by a centralized server.

- Performance: The performance of firewalls is often slower than the link speed of network interfaces. Every network resource for firewall needs to check firewall rules according to network conditions. Firewalls can be adaptively deployed among network switches, depending on network conditions in the framework.

- The management of access control: Since there may be hundreds of network resources in an administered network, the dynamic management of access control for security services like firewall is a challenge. In the framework, firewall rules can be dynamically added for new malware.

- The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for firewall within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

- Packet-based access mechanism: Packet-based access mechanism is not enough for firewall in practice since the basic unit of access control is usually users or applications. Therefore, application level rules can be defined and added to the firewall system through the centralized server.

7.2. Centralized DDoS-attack Mitigation System

For the centralized DDoS-attack mitigation system, a centralized DDoS-attack mitigation can manage each network resource and manipulate rules to each switch through a centralized server for DDoS-attack mitigation (called DDoS-attack Mitigator). The centralized DDoS-attack mitigation system defends servers against

DDoS attacks outside private network, that is, from public network.

Servers are categorized into stateless servers (e.g., DNS servers) and stateful servers (e.g., web servers). For DDoS-attack mitigation, traffic flows in switches are dynamically configured by traffic flow forwarding path management according to the category of servers [AVANT-GUARD]. Such a managenent should consider the load balance among the switches for the defense against DDoS attacks.

The procedure of DDoS-attack mitigation operations in the centralized DDoS-attack mitigation system is as follows:

1. Switch periodically reports an inter-arrival pattern of a flow's packets to Switch Controller.

2. Switch Controller forwards the flow's inter-arrival pattern to an appropriate security service application, such as DDoS-attack Mitigator.

3. DDoS-attack Mitigator analyzes the reported pattern for the flow.

4. If DDoS-attack Mitigator regards the pattern as a DDoS attack, it computes a packet dropping probability corresponding to suspiciousness level and reports this DDoS-attack flow to Switch Controller.

5. Switch Controller installs new rules into switches (e.g., forward packets with the suspicious inter-arrival pattern with a dropping probability).

6. The suspicious flow's packets are randomly dropped by switches with the dropping probability.

For the above centralized DDoS-attack mitigation system, the existing SDN protocols can be used through standard interfaces between the DDoS-attack mitigator application and switches [RFC7149] [ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

The centralized DDoS-attack mitigation system has challenges similar to the centralized firewall system. The proposed framework can resolve these challenges through the above centralized DDoS-attack mitigation system based on SDN as follows:

- Cost: The cost of adding DDoS-attack mitigators to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add DDoS-attack mitigator on each network resource. To solve this, each network resource can be managed centrally such that a single DDoS-attack mitigator is

manipulated by a centralized server.

- Performance: The performance of DDoS-attack mitigators is often slower than the link speed of network interfaces. The checking of DDoS attacks may reduce the performance of the network interfaces. DDoS-attack mitigators can be adaptively deployed among network switches, depending on network conditions in the framework.

- The management of network resources: Since there may be hundreds of network resources in an administered network, the dynamic management of network resources for performance (e.g., load balancing) is a challenge for DDoS-attack mitigation. In the framework, as dynamic network resource management, traffic flow forwarding path management can handle the load balancing of network switches [AVANT-GUARD]. With this management, the current and near-future workload can be spread among the network switches for DDoS-attack mitigation. In addition, DDoS-attack mitigation rules can be dynamically added for new DDoS attacks.

- The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for new DDoS-attacks (e.g., DNS reflection attack) within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

7.3. Centralized VoIP/VoLTE Security System

For the centralized VoIP/VoLTE security system, a centralized 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.

The procedure of VoIP/VoLTE security operations in the centralized VoIP/VoLTE security system is as follows:

1. A switch forwards an unknown call flow’s signal packet (e.g., SIP packet) to Switch Controller. Also, if the packet belongs to a matched flow’s packet related to SIP (called matched SIP packet), Switch forwards the packet to Switch Controller so that the packet can be checked by an NSF for VoIP (i.e., VoIP IPS) via Switch Controller, which monitors the behavior of its SIP call.

2. Switch Controller forwards the unknown flow’s packet or the matched SIP packet to an appropriate security service function,

such as VoIP IPS.

3. VoIP IPS analyzes the headers and contents of the signal packet, such as IP address, calling number, and session description [RFC4566].

4. If VoIP IPS regards the packet as a spoofed packet by hackers or a scanning packet searching for VoIP/VoLTE devices, it requests the Switch Controller to block that packet and the subsequent packets that have the same call-id.

5. Switch Controller installs new rules (e.g., drop packets) into switches.

6. The illegal packets are dropped by switches.

For the above centralized VoIP/VoLTE security system, the existing SDN protocols can be used through standard interfaces between the VoIP IPS application and switches [RFC7149][ITU-T.Y.3300] [ONF-OpenFlow][ONF-SDN-Architecture].

Legacy hardware based VoIP IPSes have some challenges, such as provisioning time, the granularity of security, expensive cost, and the establishment of policy. The proposed framework can resolve these challenges through the above centralized VoIP/VoLTE security system based on SDN as follows:

- Provisioning: The provisioning time of setting up a legacy VoIP IPS to network is substantial because it takes from some hours to some days. By managing the network resources centrally, VoIP IPS can provide more agility in provisioning both virtual and physical network resources from a central location.

- The granularity of security: The security rules of a legacy VoIP IPS are compounded considering the granularity of security. The proposed framework can provide more granular security by centralizing security control into a switch controller. The VoIP IPS can effectively manage security rules throughout the network.

- Cost: The cost of adding VoIP IPS to network resources, such as routers, gateways, and switches is substantial due to the reason that we need to add VoIP IPS on each network resource. To solve this, each network resource can be managed centrally such that a single VoIP IPS is manipulated by a centralized server.

- The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for VoIP IPS within a specific

organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

So far this document has described the procedure and impact of the three use cases for security services. To support these use cases in the proposed framework, a data model described in [i2nsf-cap-interface-yang] can be used as NSF facing interface along with NETCONF [RFC6241].

8. Security Considerations

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

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

9. Acknowledgements

This document was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [10041244, Smart TV 2.0 Software Platform] and by MSIP/IITP [R0166-15-1041, Standard Development of Network Security based SDN].

This document has greatly benefited from inputs by Jinyong Kim, Daeyoung Hyun, Mahdi Daghmehchi-Firoozjaei, and Geumhwan Cho.

10. References

10.1. Normative References

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

[i2nsf-framework] Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-01, June 2016.

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

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

10.2. Informative References

[i2nsf-cap-interface-yang] Jeong, J., Kim, J., Hyun, D., Park, J., and T. Ahn, "YANG Data Model of Interface to Network Security Functions Capability Interface", draft-jeong-i2nsf-capability-interface-yang-00, July 2016.

[i2nsf-sec-mgnt-arch] 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-01, July 2016.

[i2nsf-sfc-enabled-arch] Hyun, S., Woo, S., Yeo, Y., Jeong, J., and J. Park, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-sfc-enabled-i2nsf-00, July 2016.

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

[ITU-T.Y.3300] Recommendation ITU-T Y.3300, "Framework of Software-Defined Networking", June 2014.

[ONF-OpenFlow] ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.

[ONF-SDN-Architecture] ONF, "SDN Architecture", June 2014.

[ITU-T.X.1252] Recommendation ITU-T X.1252, "Baseline Identity Management Terms and Definitions", April 2010.

[ITU-T.X.800] Recommendation ITU-T X.800, "Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

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

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

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

Appendix A. Changes from draft-jeong-i2nsf-sdn-security-services-04

The following changes were made from draft-jeong-i2nsf-sdn-security-services-04:

- According to the change of terminology in the I2NSF framework, the names of the components and interfaces are updated as follows: Application Controller -> I2NSF Client, Security Function (SF) -> Network Security Function (NSF), Vendor System -> Developer's Management System, Service Layer Interface -> Client Facing Interface, Capability Layer Interface -> NSF Facing Interface.

- Three use cases described in this document can use a data model corresponding to the information model for the I2NSF capability interface.

- The proposed framework of SDN-based security services can be combined to a security management architecture for handling security policies.

- The proposed framework can enforce low-level security policies in NSFs by using a service function chaining (SFC) enabled I2NSF architecture.

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

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

Phone: +82 31 299 4324
Fax: +82 31 290 7996
EMail: hyoung@skku.edu
URI: http://seclab.skku.edu/people/hyoungshick-kim/

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

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

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

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

Se-Hui Lee
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8162
EMail: sehuilee@kt.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

H. Kim
H. Ko
D. Daghmehchi
J. Jeong
Sungkyunkwan University
T. Ahn
Korea Telecom
October 31, 2016

I2NSF Data Model of Consumer-Facing Interface for Security Management draft-kim-i2nsf-consumer-facing-interface-dm-00

Abstract

This document describes a 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 NSF instances 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 May 4, 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-management], which is based on I2NSF framework [i2nsf-framework]. 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-management] 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-management]. 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.

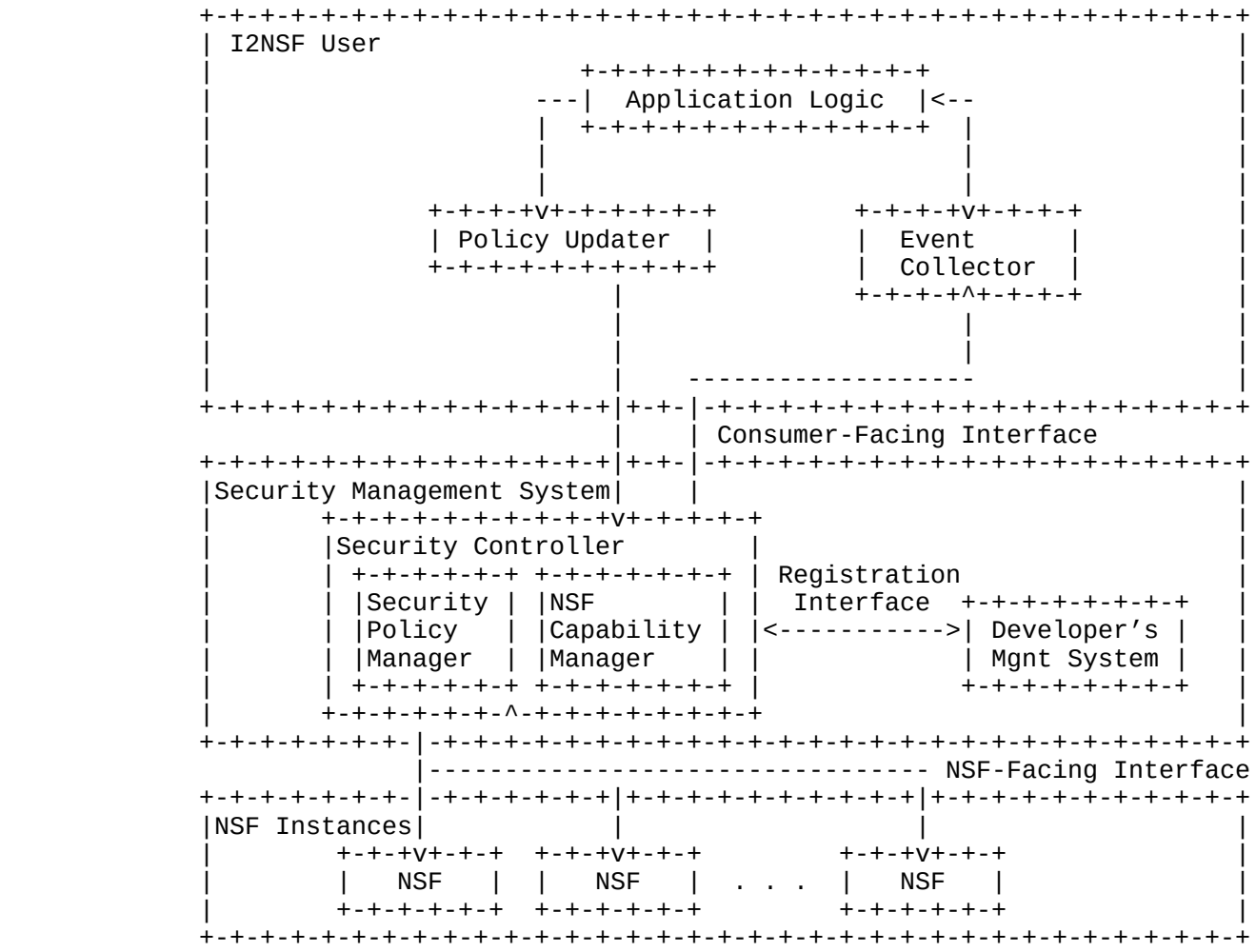


Figure 1: Security Management Architecture in I2NSF

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 Model for VoIP-VoLTE Security Service

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 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 services. Figure 2 shows a part of this data model.

```
+--: (policy)
   +--rw policy-lifecycle  *(policy-lifecycel-id)
   |  +--rw expiration-event
   |  |  +--rw enabled boolean
   |  |  +--rw event-id  uint 16
   |  +--rw expiration-time
   |     +--rw enabled   boolean
   |     +--rw time      data-and-time
   +--rw policy-rule   *[policy-rule-id]
   |  +--rw policy-name  string
   |  +--rw policy-rule-id   uint 16
   |  +--rw service
   |  |  +--voip-handling  boolean
   |  |  +--volet-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
      +--rw (action-type)?
         +--: (ingress-action)
         |  +--rw permit?  boolean
         |  +--rw mirror?  boolean
         |  +--rw log?     boolean
         +--: (engress-type)
            +--rw redirection?   boolean
```

Figure 2: 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.

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 Sanghak Oh, Eunsoo Kim, Soyoung Kim, 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-management] 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.

Authors' Addresses

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

Phone: +82 31 299 4324
Fax: +82 31 290 7996
EMail: hyoung@skku.edu
URI: http://seclab.skku.edu/people/hyoungshick-kim/

Hoon Ko
Department of Computer Science and Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82-31-299-4104
EMail: skoh21@skku.edu

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

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

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

October 31, 2016

I2NSF Network Security Functions Facing Interface YANG Data Model
draft-kim-i2nsf-nsf-facing-interface-data-model-00

Abstract

This document defines a YANG data model corresponding to the information model for Interface to Network Security Functions (I2NSF) NSF facing interface. It describes a data model for three security capabilities (i.e., network security functions), such as network security control, content security control, and attack mitigation control, as defined in the information model for the I2NSF NSF facing 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 May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for security services with the information model of Interface to Network Security Functions (I2NSF) NSF facing interface. 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 rule*  [rule-id]
|      +--rw rule-name  string
|      +--rw rule-id  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
|      |  |  +--rw packet-security-mac-condition* [pkt-sec-cond-mac-id]
|      |  |  |  +--rw pkt-sec-cond-mac-id uint 8
|      |  |  |  +--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*[pkt-sec-cond-ipv4-id]
|       |  |  |  +--rw pkt-sec-cond-ipv4-id 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-protocol string
|       |  |  |  +--rw pkt-sec-cond-ipv4-dscp string
|       |  |  |  +--rw pkt-sec-cond-ipv4-ecn string
|       |  |  |  +--rw pkt-sec-cond-ipv4-length string
|       |  |  |  +--rw pkt-sec-cond-ipv4-ttl
|       |  |  +--rw packet-security-ipv6-condition*[pkt-sec-cond-ipv6-id]
|       |  |  |  +--rw pkt-sec-cond-ipv6-id uint 8
|       |  |  |  +--rw pkt-sec-cond-ipv6-src inet:ipv6-address
|       |  |  |  +--rw pkt-sec-cond-ipv6-dest inet:ipv6-address
|       |  |  |  +--rw pkt-sec-cond-ipv6-dscp string
|       |  |  |  +--rw pkt-sec-cond-ipv6-ecn string
|       |  |  |  +--rw pkt-sec-cond-ipv6-flow-label string
|       |  |  |  +--rw pkt-sec-cond-ipv6-payload-length string
|       |  |  |  +--rw pkt-sec-cond-ipv6-next-header string
|       |  |  |  +--rw pkt-sec-cond-ipv6-hop-limit string
|       |  |  +--rw packet-security-tcp-condition* [pkt-sec-cond-tcp-id]
|       |  |  |  +--rw pkt-sec-cond-tcp-id uint 8
|       |  |  |  +--rw pkt-sec-cond-tcp-src-port inet:port-number
|       |  |  |  +--rw pkt-sec-cond-tcp-dest-port inet:port-number
|       |  |  |  +--rw pkt-sec-cond-tcp-seq-num string
|       |  |  |  +--rw pkt-sec-cond-tcp-falgs string
|       |  |  +--rw packet-security-udp-condition* [pkt-sec-cond-udp-id]
|       |  |     +--rw pkt-sec-cond-udp-id uint 8
|       |  |     +--rw pkt-sec-cond-udp-src-port inet:port-number
|       |  |     +--rw pkt-sec-cond-udp-dest-port inet:port-number
|       |  |     +--rw pkt-sec-cond-udp-length string
|       |  +--rw packet-payload-security-condition* [pkt-payload-id]
|       |  |  +--rw pkt-payload-id uint 8
|       |  +--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)?
|            |    +--: (permit)
|            |    |  +--rw permit boolean
|            |    +--: (deny)
|            |    |  +--rw deny boolan
|            |    +--: (mirror)
|            |       +--rw mirror boolean
|            +--: (egress-action)
|            |  +--rw (egress-action-type)?
|            |    +--: (invoke-signaling)
|            |    |  +--rw invoke-signaling boolean
|            |    +--: (tunnel-encapsulation)
|            |    |  +--rw tunnel-encapsulation boolean
|            |    +--: (forwarding)
|            |       +--rw forwarding  boolean
|            +--: (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
|                  |        +--rw voip-volte-rule*  [voip-volte-rule-id]
|                  |          +--rw voip-volte-rule-id  uint 8
|                  |          +--rw event
|                  |          |  +--rw called-voip  boolean
|                  |          |  +--rw called-volte  boolean
|                  |          +--rw condition
|                  |          |  +--rw sip-header*  [sip-header-uri]
|                  |          |  |  +--rw sip-header-uri string
|                  |          |  |  +--rw sip-header-method string
|                  |          |  |  +--rw expire-time yang:date-and-time
|                  |          |  |  +--rw sip-header-user-agent uint32
|                  |          |  +--rw cell-region?* [cell-id-region]
|                  |          |     +--rw cell-id-region uint 32
|                  |          +--rw action
|                  |             +--rw (action-type)?
|                  |                +--: (ingress-action)
|                  |                |  +--rw (ingress-action-type)?
|                  |                |    +--: (permit)
|                  |                |    |  +--rw permit boolean
|                  |                |    +--: (deny)
|                  |                |    |  +--rw deny boolean
|                  |                |    +--: (mirror)
|                  |                |       +--rw mirror boolean
|                  |                +--: (egress-action)
|                  |                   +--: (egress-action-type)?
|                  |                     +--: (redirection)
|                  |                        +--rw redirection? boolean
|                  +--: (attack-mitigation-control)
|                     +--rw (attack-mitigation-control-type)?
|                        +--: (ddos-attack)
|                        |  +--rw (ddos-attack-type)?
|                        |     +--: (network-layer-ddos-attack)
|                        |     |  +--rw network-layer-ddos-attack-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)?
|                |    +--: (permit)
|                |    |  +--rw permit boolean
|                |    +--: (deny)
|                |    |  +--rw deny 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@2016-10-31.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 "2016-10-31"{
    description "Initial revision";
    reference
      "draft-xia-i2nsf-capability-interface-im-06
       draft-jeong-i2nsf-capability-interface-yang-03.txt";
  }

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

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

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

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

            list packet-security-mac-condition {
              key pkt-sec-cond-mac-id;
              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 executed or not.";

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

              leaf pkt-sec-cond-mac-dest {
                type inet:port-number;
                mandatory true;
                description
                  "This is a mandatory uint 32 attribute, and defines
                   the MAC destination address (6 octets long).";
              }

              leaf pkt-sec-cond-mac-src {
                type inet:port-number;
                mandatory true;
                description
                  "This is a mandatory uint 32 attribute, and defines
                   the MAC source address (6 octets long).";
              }
```

```
            leaf pkt-sec-cond-mac-8021q {
              type string;
              mandatory true;
              description
                "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.";
            }

            leaf pkt-sec-cond-mac-ether-type {
              type string;
              mandatory true;
              description
                "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.";
            }

            leaf pkt-sec-cond-mac-tci {
              type string;
              mandatory true;
              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).";
            }
          }

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

            leaf pkt-sec-cond-ipv4-id {
              type uint8;
              mandatory true;
              description
                "The ID of the pkt-sec-cond-ipv4.
                 This is key for pkt-sec-cond-ipv4-list.
                 This must be unique.";
            }
```

```
                leaf pkt-sec-cond-ipv4-src {
                  type inet:ipv4-address;
                  mandatory true;
                  description
                    "This is a mandatory inet:ipv4-address attribute,
                     and defines the IPv4 Source Address (32 bits).";
                }

                leaf pkt-sec-cond-ipv4-dest {
                  type inet:ipv4-address;
                  mandatory true;
                  description
                    "This is a mandatory inet:ipv4-address attribute,
                     and defines the IPv4 Destination Address
                     (32 bits).";
                }

                leaf pkt-sec-cond-ipv4-protocol {
                  type string;
                  mandatory true;
                  description
                    "This is a mandatory string attribute, and defines
                     he protocol used in the data portion of the IP
                     datagram (8 bits).";
                }

                leaf pkt-sec-cond-ipv4-dscp {
                  type string;
                  mandatory true;
                  description
                    "This is a mandatory string attribute, and defines
                     the Differentiated Services Code Point field
                     (6 bits).";
                }

                leaf pkt-sec-cond-ipv4-ecn {
                  type string;
                  mandatory true;
                  description
                    "This is an optional string attribute, and defines
                     the Explicit Congestion Notification field
                     (2 bits).";
                }

                leaf pkt-sec-cond-ipv4-length {
                  type string;
                  mandatory true;
                  description
```

```
                    "This is a mandatory string attribute, and defines
                     the total length of the packet (including header
                     and data) in bytes (16 bits).";
                  }

                  leaf pkt-sec-cond-ipv4-ttl {
                    type string;
                    mandatory true;
                    description
                      "This is a mandatory string attribute, and defines
                       the Time To Live in seconds (8 bits).";
                  }
                }

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

                  leaf pkt-sec-cond-ipv6-id {
                    type uint8;
                    mandatory true;
                    description
                      "The ID of the pkt-sec-cond-ipv6.
                       This is key for pkt-sec-cond-ipv6-list.
                       This must be unique.";
                  }

                  leaf pkt-sec-cond-ipv6-src {
                    type inet:ipv6-address;
                    mandatory true;
                    description
                      "This is a mandatory inet:ipv6-address attribute,
                       and defines the IPv6 Source Address (128 bits).";
                  }

                  leaf pkt-sec-cond-ipv6-dest {
                    type inet:ipv6-address;
                    mandatory true;
                    description
                      "This is a mandatory inet:ipv6-address attribute,
                       and defines the IPv6 Destination Address
                       (128 bits).";
                  }
```

```
                leaf pkt-sec-cond-ipv6-dscp {
                  type string;
                  mandatory true;
                  description
                    "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.";
                }

                leaf pkt-sec-cond-ipv6-ecn {
                  type string;
                  mandatory true;
                  description
                   "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.";
                }

                leaf pkt-sec-cond-ipv6-flow-label {
                  type string;
                  mandatory true;
                  description
                    "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).";
                }

                leaf pkt-sec-cond-ipv6-payload-length {
                  type string;
                  mandatory true;
                  description
                    "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).";
                }

                leaf pkt-sec-cond-ipv6-next-header {
                  type string;
                  mandatory true;
                  description
                    "This is a mandatory string attribute, and defines
                     the type of the next header (e.g., which extension
```

```
                    header to use) (8 bits).";
                }

                leaf pkt-sec-cond-ipv6-hop-limit {
                  type string;
                  mandatory true;
                  description
                    "This is a mandatory string attribute, and defines
                     the maximum number of hops that this packet can
                     traverse (8 bits).";
                }
              }

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

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

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

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

                leaf pkt-sec-cond-tcp-seq-num {
```

```
            type string;
            mandatory true;
            description
              "This is a mandatory string attribute, and defines
               the sequence number (32 bits).";
          }

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

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

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

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

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

```
                leaf pkt-sec-cond-udp-length {
                  type string;
                  mandatory true;
                  description
                    "This is a mandatory string attribute, and defines
                     the length in bytes of the UDP header and data
                     (16 bits).";
                }
              }
            }
              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.";
                }
              }
              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 permit {
                  description
                    "Permit case.";
                  leaf permit {
                    type boolean;
                    mandatory true;
                    description
                      "Packet flow is permitted.";
                  }
                }
                case deny {
                  description
```

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

```
            case forwarding {
              description
                "forwarding case.";
              leaf forwarding {
                type boolean;
                mandatory true;
                description
                  "TBD.";
              }
            }
          }
        }
        case 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 permit {
                    description
                      "Permit case.";
                    leaf permit {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is permitted.";
                    }
                  }
                  case deny {
                    description
                      "Deny case.";
                    leaf deny {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is denied.";
                    }
                  }
                  case mirror {
                    description
                      "Mirror case.";
                    leaf mirror {
                      type boolean;
                      mandatory true;
                      description
                        "Packet flow is mirrored.";
                    }
                  }
               }
            }
            case egress-action {
               description
                  "The engress actions consist of
                   mirror and etc.";
               choice egress-action-type {
                   description
                      "Engress-action-type: redirection,
                       and etc.";
                   case redirection {
                      description
                         "Redirection case.";
                      leaf redirection {
                         type boolean;
                         mandatory true;
```

```
                                description "TBD.";
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
```

```
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., Li, K., Zhang, D., Lopez, E., BOUTHORS, N., and L. Fang, "Information Model of Interface to Network

Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-06 (work in progress), June 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-06 (work in progress), July 2016.

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

[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.

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

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 18, 2019

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei
J. Jeong
Sungkyunkwan University

July 17, 2018

Information Model for Consumer-Facing Interface to Security Controller draft-kumar-i2nsf-client-facing-interface-im-07

Abstract

This document defines an information model for Consumer-Facing interface to Security Controller based on the requirements identified in [I-D.ietf-i2nsf-client-facing-interface-req]. The information model defines various managed objects and relationship among these objects needed to build the interface. The information model is organized based on the "Event-Condition-Event" (ECA) policy model defined by a capability information model for Interface to Network Security Functions (I2NSF) [I-D.ietf-i2nsf-capability].

Status of This Memo

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

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

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

This Internet-Draft will expire on January 18, 2019.

Table of Contents

1. Introduction

Interface to Network Security Functions (I2NSF) defines a Consumer-Facing Interface to deliver high-level security policies to Security Controller [RFC8192][RFC8329] for securiy enforcement in Network Security Functions (NSFs).

The Consumer-Facing Interface would be built using a set of objects, with each object capturing a unique set of information from Security Admin (i.e., I2NSF User [RFC8329]) needed to express a Security Policy. An object may have relationship with various other objects to express a complete set of requirement. An information model captures the managed objects and relationship among these objects. The information model proposed in this document is in accordance with interface requirements as defined in [I-D.ietf-i2nsf-client-facing-interface-req].

An NSF Capability model is proposed in [I-D.ietf-i2nsf-capability] as the basic model for both the NSF-Facing interface and Consumer-Facing Interface security policy model of this document. The information model proposed in this document is structured in accordance with the "Event-Condition-Event" (ECA) policy model.

[RFC3444] explains differences between an information and data model. This document use the guidelines in [RFC3444] to define an information model for Consumer-Facing Interface in this document. Figure 1 shows a high-level abstraction of Consumer-Facing Interface. A data model, which represents an implementation of the proposed information model in a specific data representation language, will be defined in a separate document.

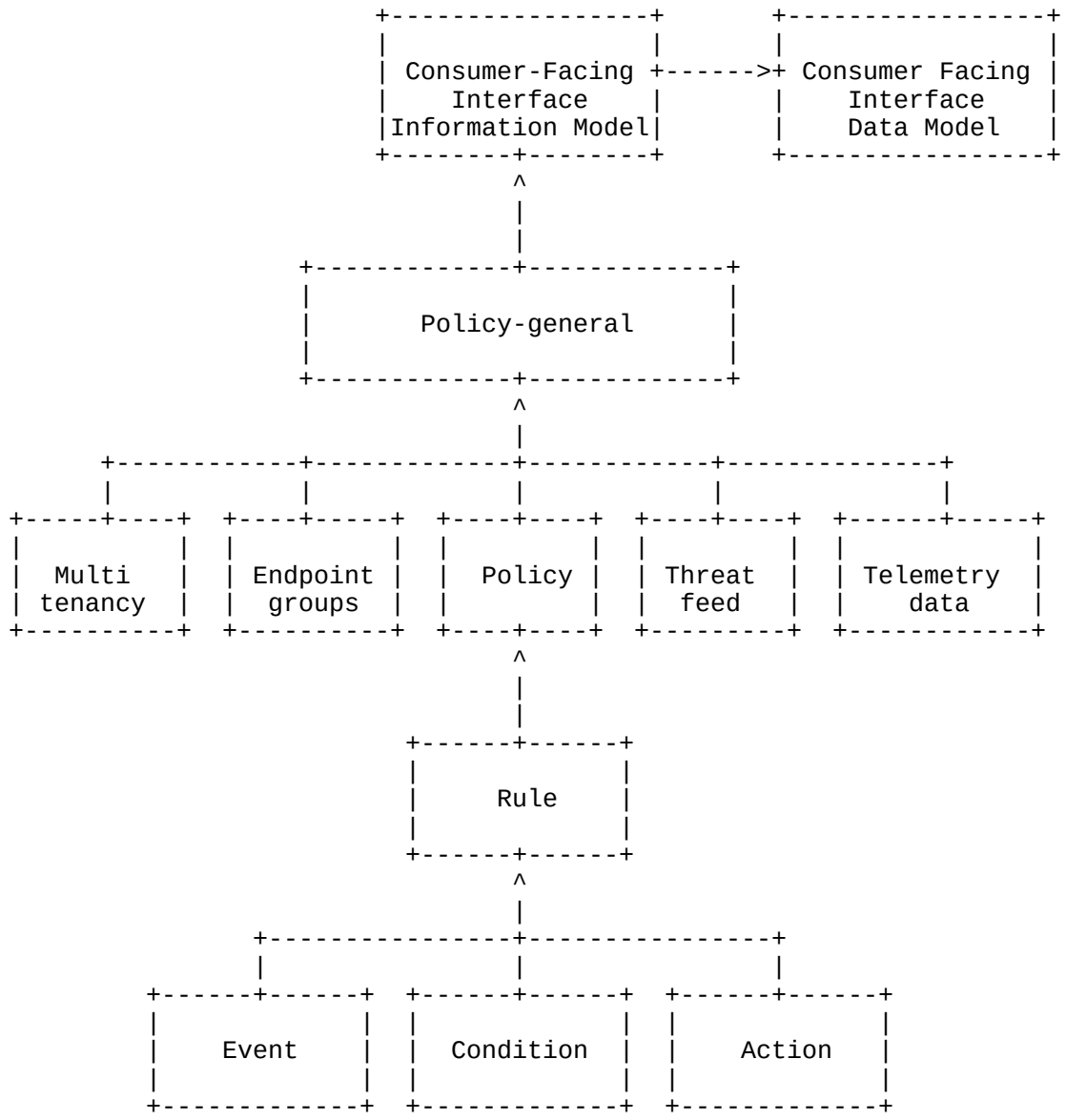


Figure 1: Diagram for High-level Abstraction of Consumer-Facing Interface

2. Conventions used in the Document

BSS: Business Support System

CLI: Command Line Interface

CMDB: Configuration Management Database

Controller: Security Controller or Management System

CRUD: Create, Retrieve, Update, Delete

FW: Firewall

GUI: Graphical User Interface

IDS: Intrusion Detection System

IPS: Intrusion Prevention System

LDAP: Lightweight Directory Access Protocol

NSF: Network Security Function, defined by [I-D.ietf-i2nsf-terminology]

OSS: Operations Support System

RBAC: Role-Based Access Control

SIEM: Security Information and Event Management

URL: Universal Resource Locator

vNSF: NSF being instantiated on Virtual Machines

3. Information Model for Policy

A Policy object represents a mechanism to express a Security Policy by Security Admin (i.e., I2NSF User) using Consumer-Facing Interface toward Security Controller; the policy would be enforced on an NSF. The Policy object SHALL have following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Multi-Tenancy: The multi-tenant environment information in which the policy is applied. The Rules in the Policy can refer

to sub-objects (e.g., domain, tenant, role, and user) of it. It can be either a reference to a Multi-Tenancy object defined in another place, or a concrete object. See details in Section 4.

End-Group: This field contains a list of logical entities in the business environment where a Security Policy is to be applied. It can be referenced by the Condition objects in a Rule, e.g., Source, Destination, Match, etc. It can be either a reference to an End-Group object defined in other place, or a concrete object. See details in Section 5.

Threat-Feed: This field represents threat feed such as Botnet servers, GeoIP, and Malware signature. This information can be referenced by the Rule Action object directly to execute the threat mitigation. See details in Section 6.

Telemetry-Data: This field represents the telemetry collection related information that the Rule Action object can refer to about how to collect the interested telemetry information, for example, what type of telemetry to collect, where the telemetry source is, where to send the telemetry information. See details in Section 7.

Rules: This field contains a list of rules. If the rule does not have a user-defined precedence, then any conflict must be manually resolved.

Owner: This field defines the owner of this policy. Only the owner is authorized to modify the contents of the policy.

A policy is a container of Rules. In order to express a Rule, a Rule must have complete information such as where and when a policy needs to be applied. This is done by defining a set of managed objects and relationship among them. A Policy Rule may be related segmentation, threat mitigation or telemetry data collection from an NSF in the network, which will be specified as the sub-model of the policy model in the subsequent sections.

The rule object SHALL have the following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event: This field includes the information to determine whether the Rule Condition can be evaluated or not. See details in Section 3.1.

Condition: This field contains all the checking conditions to apply to the objective traffic. See details in Section 3.2.

Action: This field identifies the action taken when a rule is matched. There is always an implicit action to drop traffic if no rule is matched for a traffic type. See details in Section 3.3.

Precedence: This field identifies the precedence assigned to this rule by Security Admin. This is helpful in conflict resolution when two or more rules match a given traffic class.

3.1. Event Sub-Model

The Event Object contains information related to scheduling a Rule. The Rule could be activated based on a time calendar or security event including threat level changes.

Event object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event-Type: This field identifies whether the event of triggering policy enforcement is "ADMIN-ENFORCED", "TIME-ENFORCED" or "EVENT-ENFORCED".

Time-Information: This field contains a time calendar such as "BEGIN-TIME" and "END-TIME" for one time enforcement or recurring time calendar for periodic enforcement.

Event-Map-Group: This field contains security events or threat map in order to determine when a policy needs to be activated. This is a reference to Event-Map-Group defined later.

3.1.1. Event-Map-Group

This object represents an event map containing security events and threat levels used for dynamic policy enforcement. The Event-Map-Group object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Security-Events: This contains a list of security events used for purpose for Security Policy definition.

Threat-Map: This contains a list of threat levels used for purpose for Security Policy definition.

3.2. Condition Sub-Model

This object represents Conditions that Security Admin wants to apply the checking on the traffic in order to determine whether the set of actions in the Rule can be executed or not.

The Condition object SHALL have following information:

Source: This field identifies the source of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry-Source for telemetry collection policy.

Destination: This field identifies the destination of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry- Destination for telemetry collection policy.

Match: This field identifies the match criteria used to evaluate whether the specified action needs to be taken or not. This could be either a Policy-Endpoint-Group identifying an Application set or a set of traffic rules.

Match-Direction: This field identifies whether the match criteria is to be evaluated for both directions or only one direction of the traffic with a default of allowing the other direction for stateful match conditions. This is

optional and by default a rule should apply to both directions.

Exception: This field identifies the exception consideration when a rule is evaluated for a given communication. This could be a reference to "Policy-Endpoint-Group" object or set of traffic matching criteria.

The condition object is made of condition clauses. Each condition clause consists of three tuples; variable, operator and value.

The variable and value can be source and destination IP address, for example, and they have logical operator in between to check whether they match the condition criteria set by a security admin. For Example: If condition A AND B is true: THEN execute actions ENDIF where A denotes a destination address, and B denotes a blacklisted IP address. The operator AND is the logical AND operation.

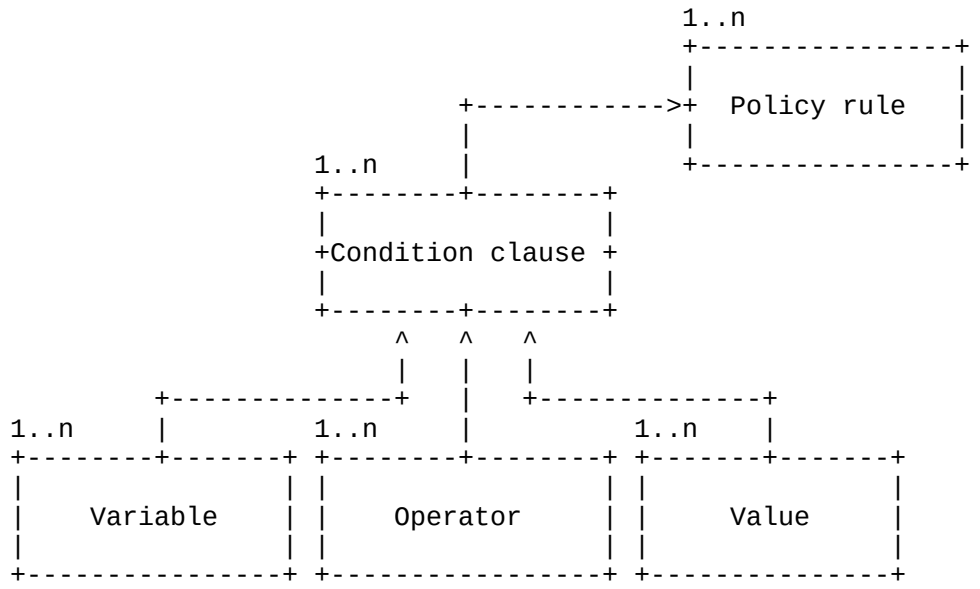


Figure 2: Condition Clause Diagram

The semantics used in a condition clause is also used in the clauses in the Event-submodel and Action sub-model.

3.3. Action Sub-Model

This object represents actions that Security Admin wants to perform based on certain traffic class.

The Action object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Primary-Action: This field identifies the action when a rule is matched by an NSF. The action could be one of "PERMIT", "DENY", "DROP-CONNECTION", "AUTHENTICATE-CONNECTION", "MIRROR", "REDIRECT", "NETFLOW", "COUNT", "ENCRYPT", "DECRYPT", "THROTTLE", "MARK", or "INSTANTIATE-NSF".

Secondary-Action: Security Admin can also specify additional actions if a rule is matched. This could be one of "LOG", "SYSLOG", or "SESSION-LOG".

4. Information Model for Multi-Tenancy

Multi-tenancy is an important aspect of any application that enables multiple administrative domains in order to manage application resources. An Enterprise organization may have multiple tenants or departments such as Human Resources (HR), Finance, and Legal, with each tenant having a need to manage their own Security Policies. In a Service Provider, a tenant could represent a Customer that wants to manage its own Security Policies.

There are multiple managed objects that constitute multi-tenancy aspects. This section lists these objects and any relationship among these objects. Below diagram shows an example of multi-tenancy in an Enterprise domain.

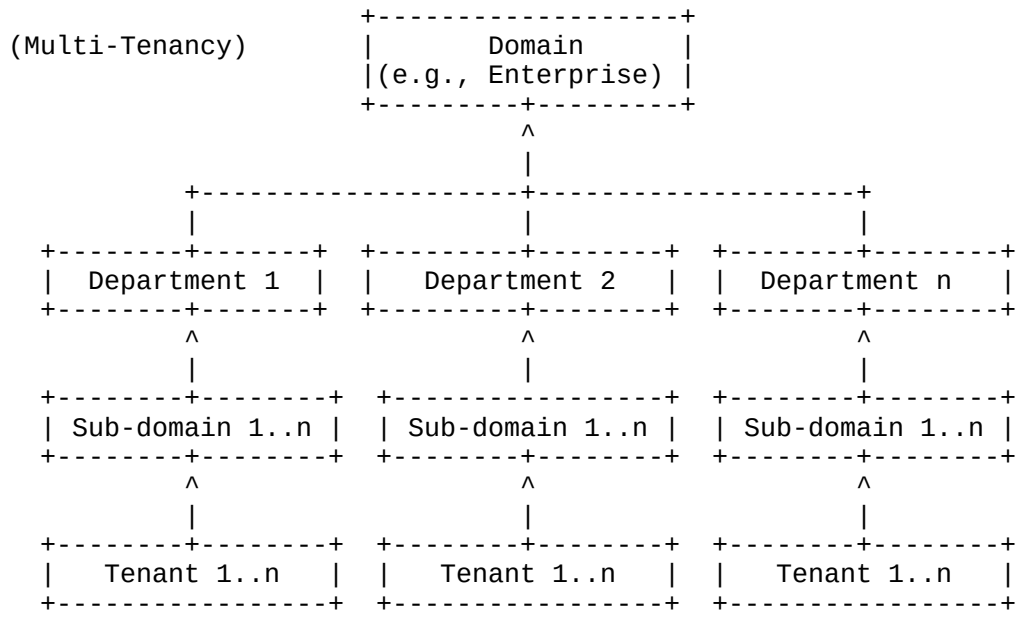


Figure 3: Multi-tenancy Diagram

4.1. Policy-Domain

This object defines a boundary for the purpose of policy management within a Security Controller. This may vary based on how the Security Controller is deployed and hosted. For example, if an Enterprise hosts a Security Controller in their network; the domain in this case could just be the one that represents that Enterprise. But if a Cloud Service Provider hosts managed services, then a domain could represent a single customer of that Provider. Multi-tenancy model should be able to work in all such environments.

The Policy-Domain object SHALL have following information:

Name: Name of the organization or customer representing this domain.

Address: Address of the organization or customer.

Contact: Contact information of the organization or customer.

Date: Date when this account was created or last modified.

Authentication-Method: Authentication method to be used for this domain. It should be a reference to a 'Policy-Management-Authentication-Method' object.

4.2. Policy-Tenant

This object defines an entity within an organization. The entity could be a department or business unit within an Enterprise organization that would like to manage its own Policies due to regulatory compliance or business reasons.

The Policy-Tenant object SHALL have following information:

Name: Name of the Department or Division within an organization.

Date: Date when this account was created or last modified.

Domain: This field identifies the domain to which this tenant belongs. This should be a reference to a Policy-Domain object.

4.3. Policy-Role

This object defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. It provides a convenient way to assign policy users to a job function or a set of permissions within the organization.

The Policy-Role object SHALL have the following information:

Name: This field identifies the name of the role.

Date: Date when this role was created or last modified.

Access-Profile: This field identifies the access profile for the role. The profile grants or denies the permissions to access Endpoint Groups for the purpose of policy management or may restrict certain operations related to policy managements.

4.4. Policy-User

This object represents a unique identity within an organization. The identity authenticates with Security Controller using credentials such as a password or token in order to perform policy management. A user may be an individual, system, or application requiring access to Security Controller.

The Policy-User object SHALL have the following information:

Name: Name of a user.

Date: Date when this user was created or last modified.

Password: User password for basic authentication.

Email: E-mail address of the user.

Scope-Type: This field identifies whether the user has domain-wide or tenant-wide privileges.

Scope-Reference: This field should be a reference to either a Policy-Domain or a Policy-Tenant object.

Role: This field should be a reference to a Policy-Role object that defines the specific permissions.

4.5. Policy Management Authentication Method

This object represents authentication schemes supported by Security Controller.

This Policy-Management-Authentication-Method object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Authentication-Method: This field identifies the authentication methods. It could be a password-based, token-based, certificate-based or single sign-on authentication.

Mutual-Authentication: This field indicates whether mutual authentication is mandatory or not.

Token-Server: This field stores the information about server that validates the token submitted as credentials.

Certificate-Server: This field stores the information about server that validates certificates submitted as credentials.

Single Sign-on-Server: This field stores the information about server that validates user credentials.

5. Information Model for Policy Endpoint Groups

The Policy Endpoint Group is a very important part of building User-construct based policies. Security Admin would create and use these objects to represent a logical entity in their business environment, where a Security Policy is to be applied.

There are multiple managed objects that constitute a Policy Endpoint Group. This section lists these objects and relationship among these objects.

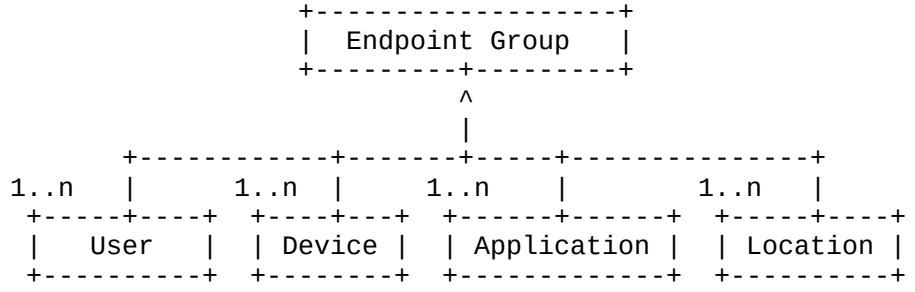


Figure 4: Endpoint Group Diagram

5.1. Tag-Source

This object represents information source for tag. The tag in a group must be mapped to its corresponding contents to enforce a Security Policy.

Tag-Source object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Tag-Type: This field identifies the Endpoint Group type. It can be a User-Group, App-Group, Device-Group or Location-Group.

Tag-Source-Server: This field identifies information related to the source of the tag such as IP address and UDP/TCP port information.

Tag-Source-Application: This filed identifies the protocol, e.g., LDAP, Active Directory, or CMDB used to communicate with a server.

Tag-Source-Credentials: This field identifies the credential information needed to access the server.

5.2. User-Group

This object represents a user group based on either tag or other information.

The User-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the user group is based on User-tag, User-name or IP-address.

Metadata-Server: This field should be a reference to a Metadata-Source object.

Group-Member: This field is a list of User-tag, User-names or IP addresses based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.3. Device-Group

This object represents a device group based on either tag or other information.

The Device-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the device group is based on Device-tag or Device-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Device-tag, Device-name or IP address based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.4. Application-Group

This object represents an application group based on either tag or other information.

The Application-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the application group is based on App-tag or App-name, or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Application-tag Application-name or IP address and port information based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

5.5. Location-Group

This object represents a location group based on either tag or other information.

The 'Location-Group' object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the location group is based on Location-tag, Location-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Location-tag, Location-name or IP addresses based on Group-Type.

Risk Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

6. Information Model for Threat Prevention

The threat prevention plays an important part in the overall security posture by reducing the attack surfaces. This information could come in the form of threat feeds such as Botnet and GeoIP feeds usually from a third party or external service.

There are multiple managed objects that constitute this category. This section lists these objects and relationship among these objects.

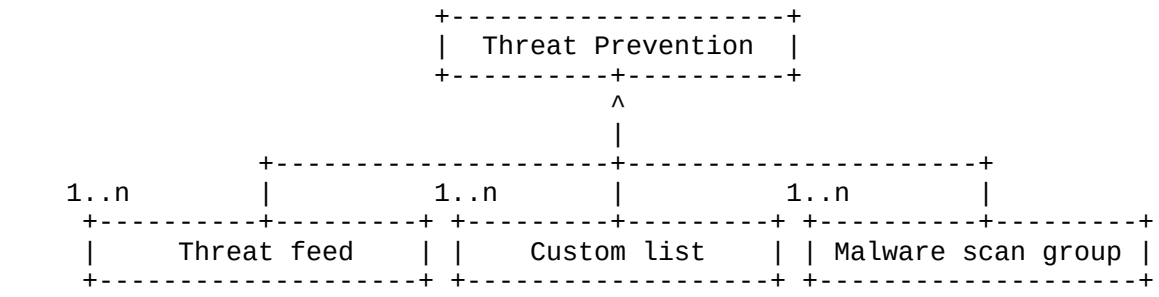


Figure 5: Threat Prevention Diagram

6.1. Threat-Feed

This object represents a threat feed such as Botnet servers and GeoIP.

The Threat-Feed object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Feed-Type: This field identifies whether a feed type is IP address-based or URL-based.

Feed-Server: This field identifies the information about the feed provider, it may be an external service or local server.

Feed-Priority: This field represents the feed priority level to resolve conflict if there are multiple feed sources; the valid range may be 0 to 9.

6.2. Custom-List

This object represents a custom list created for the purpose of defining exception to threat feeds. An organization may want to allow a certain exception to threat feeds obtained from a third party

The Custom-List object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

List-Type: This field identifies whether the list type is IP address-based or URL-based.

List-Property: This field identifies the attributes of the list property, e.g., Blacklist or Whitelist.

List-Content: This field contains contents such as IP addresses or URL names.

6.3. Malware-Scan-Group

This object represents information needed to detect malware. This information could come from a local server or uploaded periodically from a third party.

The Malware-Scan-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Signature-Server: This field contains information about the server from where signatures can be downloaded periodically as updates become available.

File-Types: This field contains a list of file types needed to be scanned for the virus.

Malware-Signatures: This field contains a list of malware signatures or hash values.

7. Information Model for Telemetry Data

Telemetry provides System Admin with the visibility of the network activities which can be tapped for further security analytics, e.g., detecting potential vulnerabilities, malicious activities, etc.

7.1. Telemetry-Data

This object contains information collected for telemetry.

The Telemetry-Data object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Log-Data: This field identifies whether Log data need to be collected.

Syslog-Data This field identifies whether Syslog data need to be collected.

SNMP-Data: This field identifies whether SNMP traps and alarm data need to be collected.

sFlow-Record: This field identifies whether sFlow records need to be collected.

NetFlow-Record: This field identifies whether NetFlow record need to be collected.

NSF-Stats: This field identifies whether statistics need to be collected from an NSF.

7.2. Telemetry-Source

This object contains information related to telemetry source. The source would be an NSF in the network.

The Telemetry-Source object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Source-Type: This field contains the type of the NSF telemetry source: "NETWORK-NSF", "FIREWALL-NSF", "IDS-NSF", "IPS-NSF", "PROXY-NSF or "OTHER-NSF".

NSF-Source: This field contains information such as IP address and protocol (UDP or TCP) port number of the NSF providing telemetry data.

NSF-Credentials: This field contains a username and a password used to authenticate the NSF.

Collection-Interval: This field contains time in milliseconds between each data collection. For example, a value of 5,000 means data is streamed to collector every 5 seconds. Value of 0 means data streaming is event-based.

Collection-Method: This field contains a method of collection whether it is PUSH-based or PULL-based.

Heartbeat-Interval: This field contains time in seconds when the source must send telemetry heartbeat.

QoS-Marking: This field contains a DSCP value source marked on its generated telemetry packets.

7.3. Telemetry-Destination

This object contains information related to telemetry destination. The destination is usually a collector which is either a part of Security Controller or external system such as SIEM.

The Telemetry-Destination object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Collector-Source: This field contains the information such as IP address and protocol (UDP or TCP) port number for the collector’s destination.

Collector-Credentials: This field contains a username and a password provided by the collector.

Data-Encoding: This field contains the telemetry data encoding, which could in the form of a schema.

Data-Transport: This field contains streaming telemetry data protocols: whether it is gRPC, protocol buffer over UDP, etc.

8. Role-Based Acess Control (RBAC)

Role-Based Access Control (RBAC) provides a powerful and centralized control within a network. It is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC, such as role-permissions, user-role and role-role relationships, make it simple to perform user assignments.

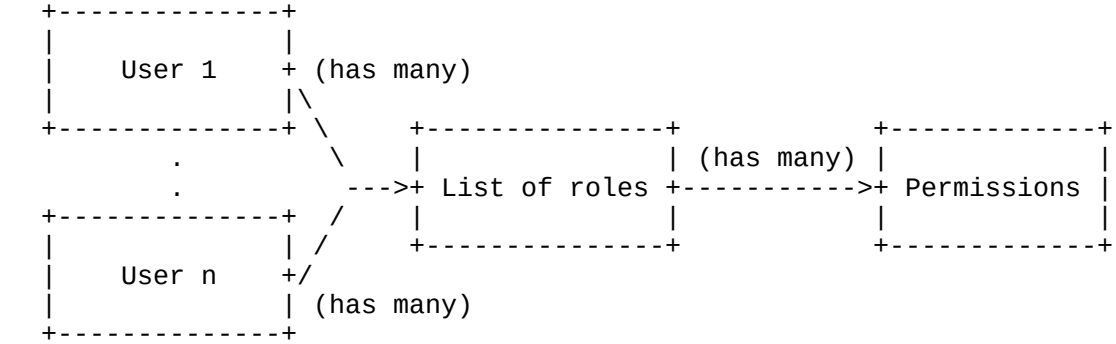


Figure 6: RBAC Diagram

As shown in Figure 6, a role represents a collection of permissions (e.g., accessing a file server or other particular resources). A role may be assigned to one or multiple users. Both roles and permissions can be organized in a hirarchy. A role may consists of other roles and permissions.

Following are the steps required to build RBAC.

1. Defining roles and permissions.

2. Establishing relations among roles and permissions.

3. Defining users.

4. Associating rules with roles and permissions.

5. assigning roles to users.

9. Security Considerations

An information model provides a mechanism to protect Consumer-Facing Interface between System Admin (i.e., I2NSF User) and Security Controller. One of the specified mechanism must be used to protect an Enterprise network, data and all resources from external attacks. This information model mandates that the interface must have proper

authentication and authorization with Role-Based Access Controls to address the multi-tenancy requirement. The document does not mandate that a particular mechanism should be used because a different organization may have different needs based on their deployment.

10. IANA Considerations

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

11. Acknowledgments

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

12. Contributors

This document is the work of I2NSF working group, greatly benefiting from inputs and suggestions by Kunal Modasiya, Prakash T. Sehsadri and Srinivas Nimmagadda from Juniper Networks. The authors sincerely appreciate their contributions.

The following are contributing authors of this document, who are considered co-authors:

- Eunsoo Kim (Sungkyunkwan University)
- Seungjin Lee (Sungkyunkwan University)
- Hyoungshick Kim (Sungkyunkwan University)

13. Informative References

[I-D.ietf-i2nsf-capability]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-02 (work in progress), July 2018.

[I-D.ietf-i2nsf-client-facing-interface-req]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-05 (work in progress), May 2018.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-06 (work in progress), July 2018.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

Appendix A. Changes from draft-kumar-i2nsf-client-facing-interface-im-06

The following changes have been made from draft-kumar-i2nsf-client-facing-interface-im-06:

- In Section 1, Figure 1 is added to show a diagram for a high-level abstraction of Consumer-Facing Interface.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rakeshkumarcloud@gmail.com

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Nabil Bitar
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: nabil.bitar@nokia.com

Senad Palislamovic
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: senad.palislamovic@nokia.com

Liang Xia
Huawei
101 Software Avenue
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

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

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

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:

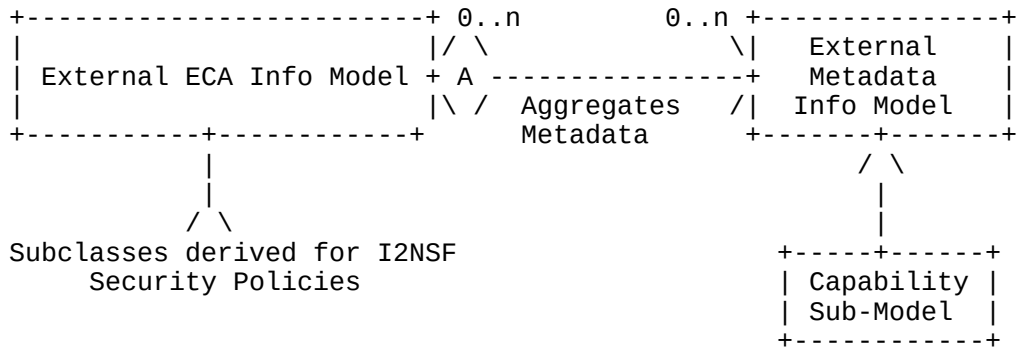


Figure 1. The Overall I2NSF Information Model Design

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:

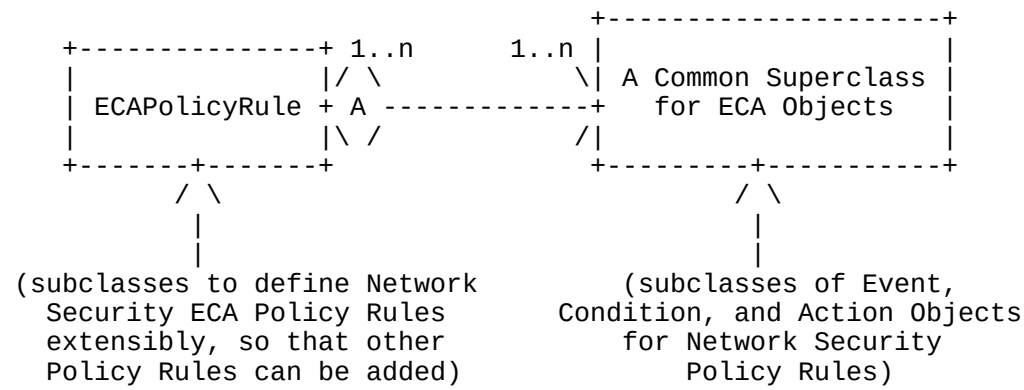


Figure 2. Network Security Information Sub-Model Overview

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.

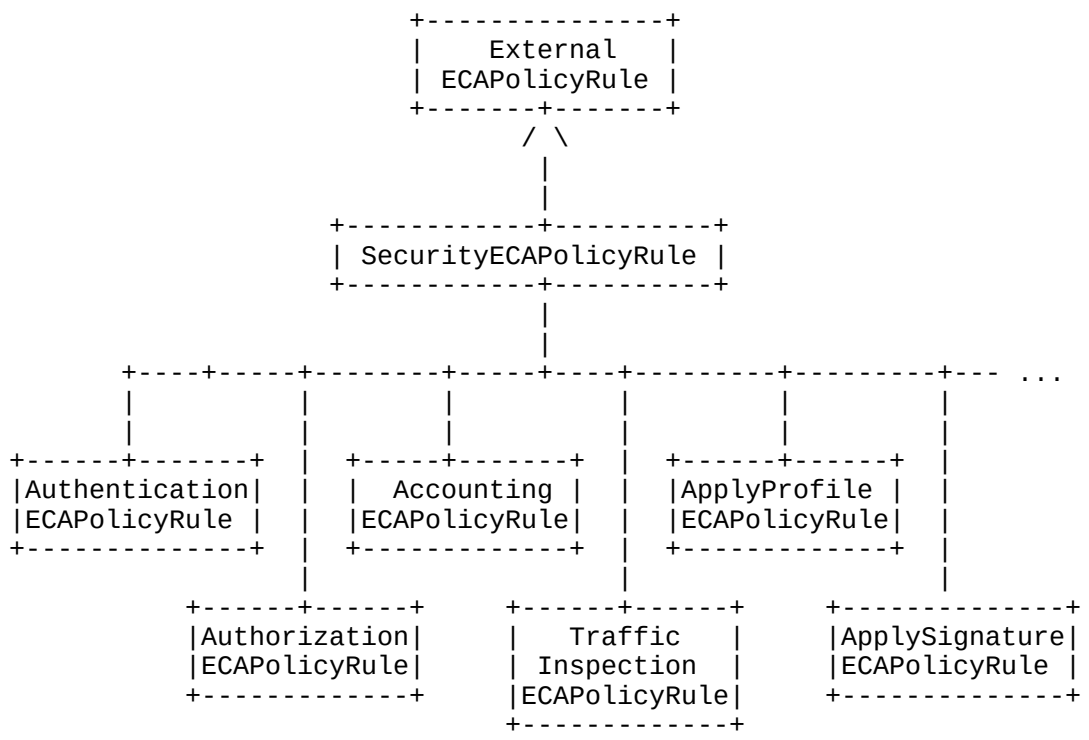


Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

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.

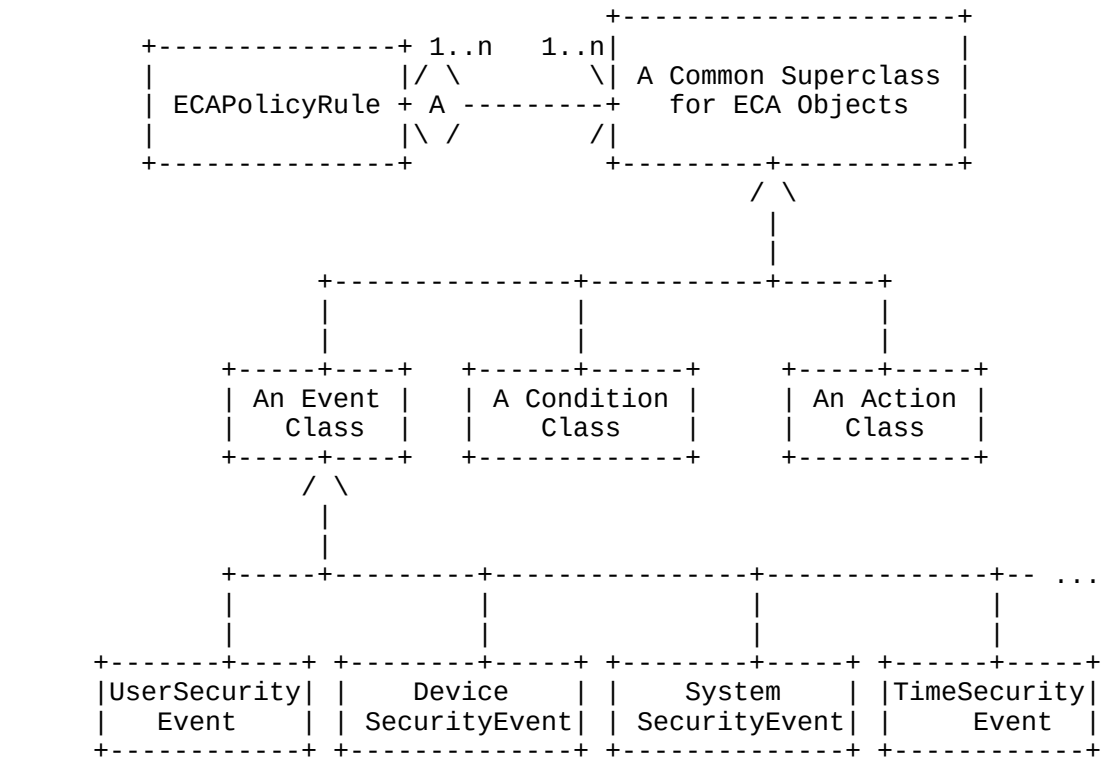


Figure 4. Network Security Info Sub-Model Event Class Extensions

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

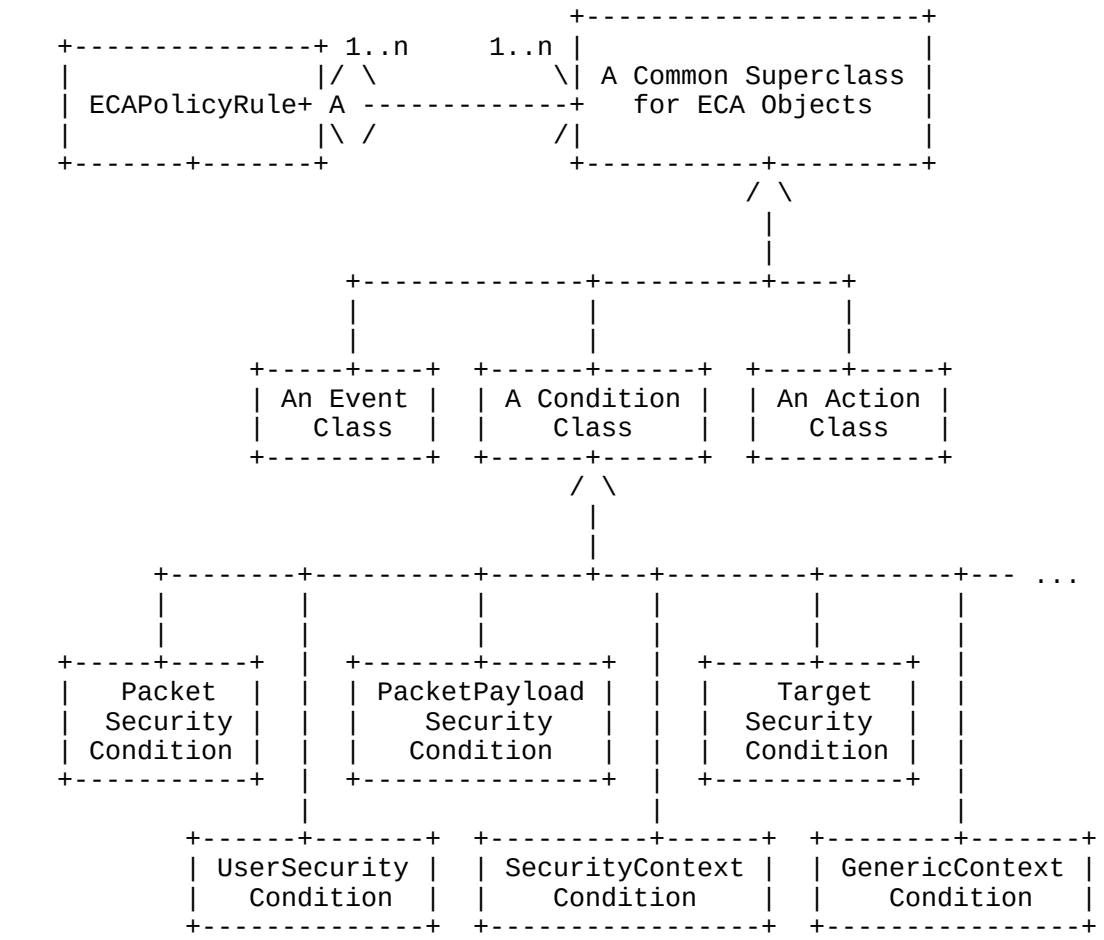


Figure 5. Network Security Info Sub-Model Condition Class Extensions

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.

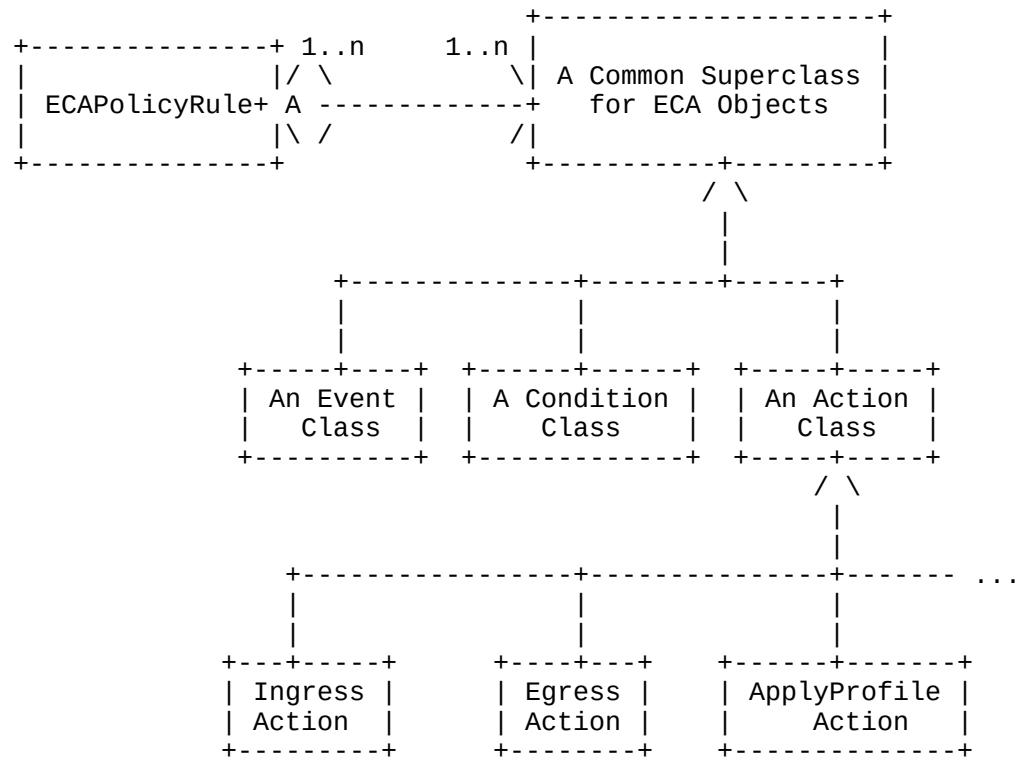


Figure 6. Network Security Info Sub-Model Action Extensions

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.

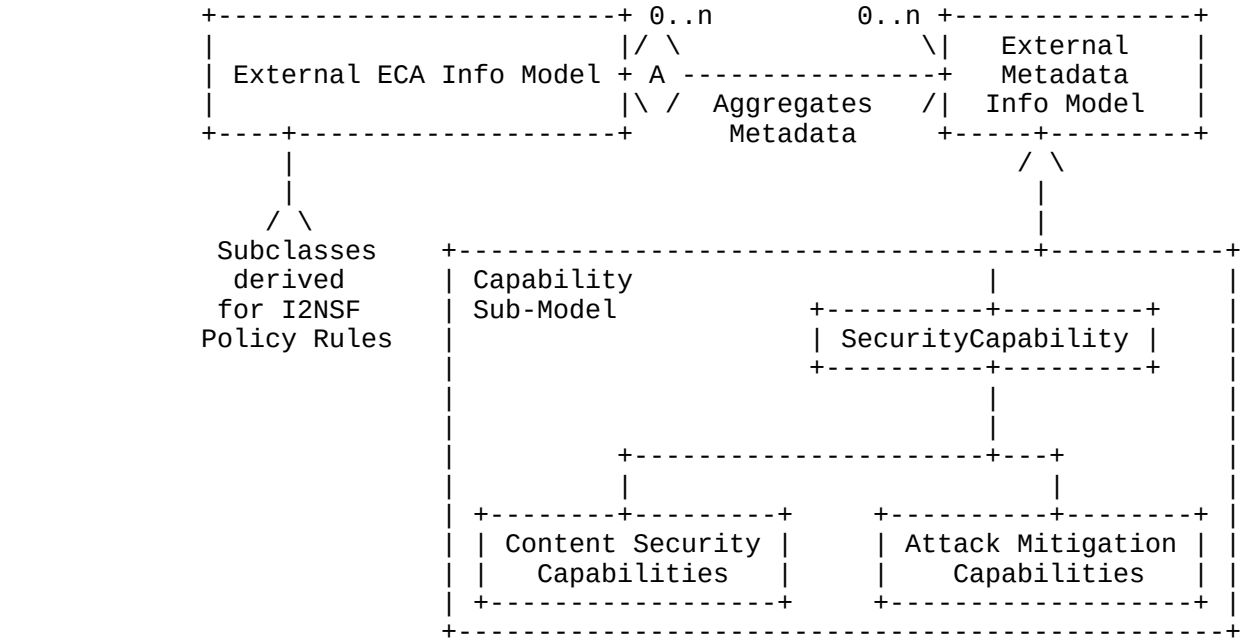


Figure 7. I2NSF Security Capability High-Level Model

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.

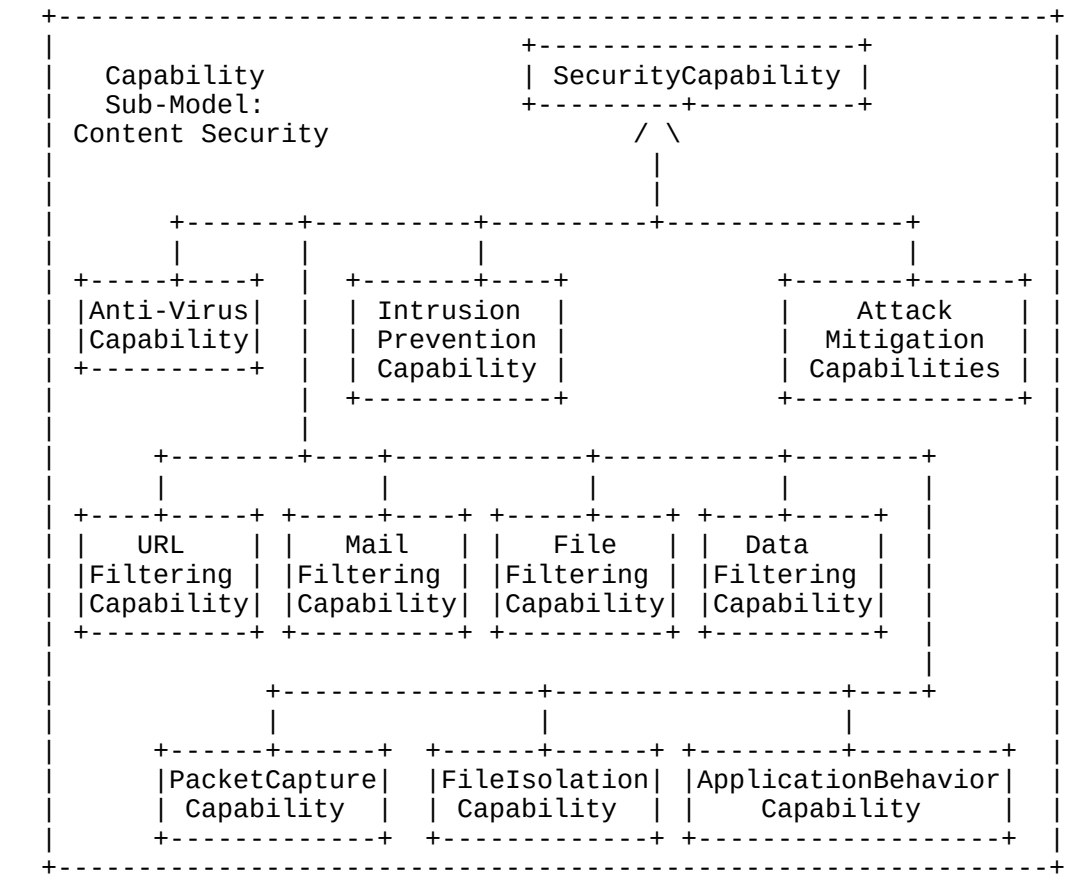


Figure 8. Network Security Capability Information Model

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.

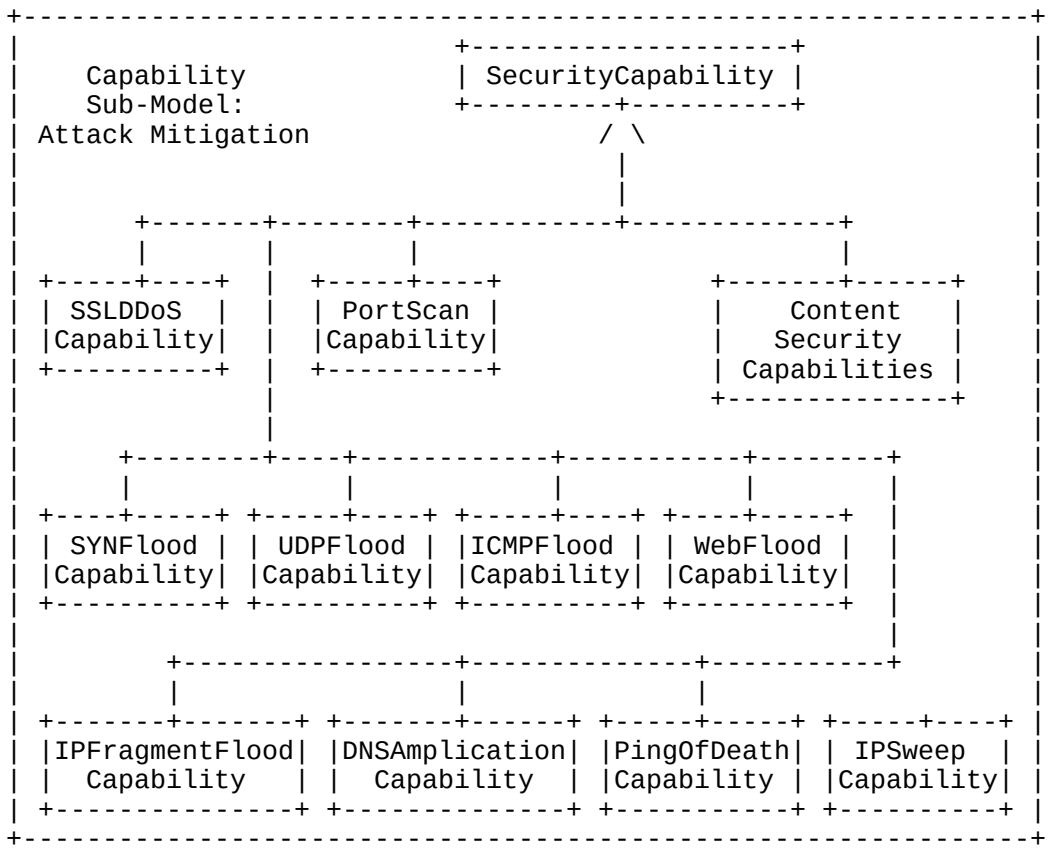


Figure 9. Attack Mitigation Capability Information Model

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.

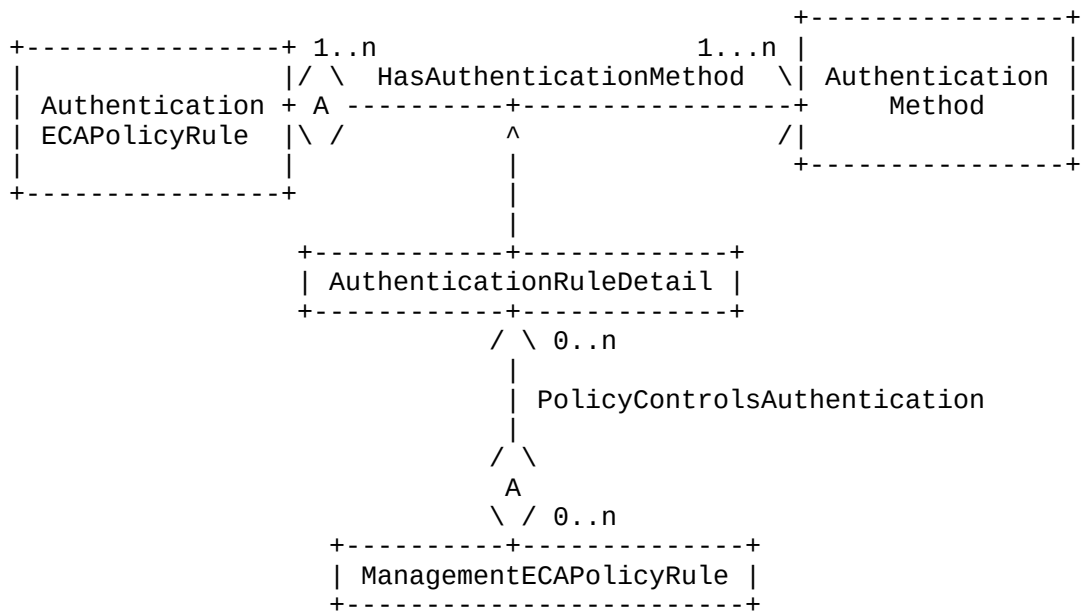


Figure 10. Modeling Authentication Mechanisms

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:

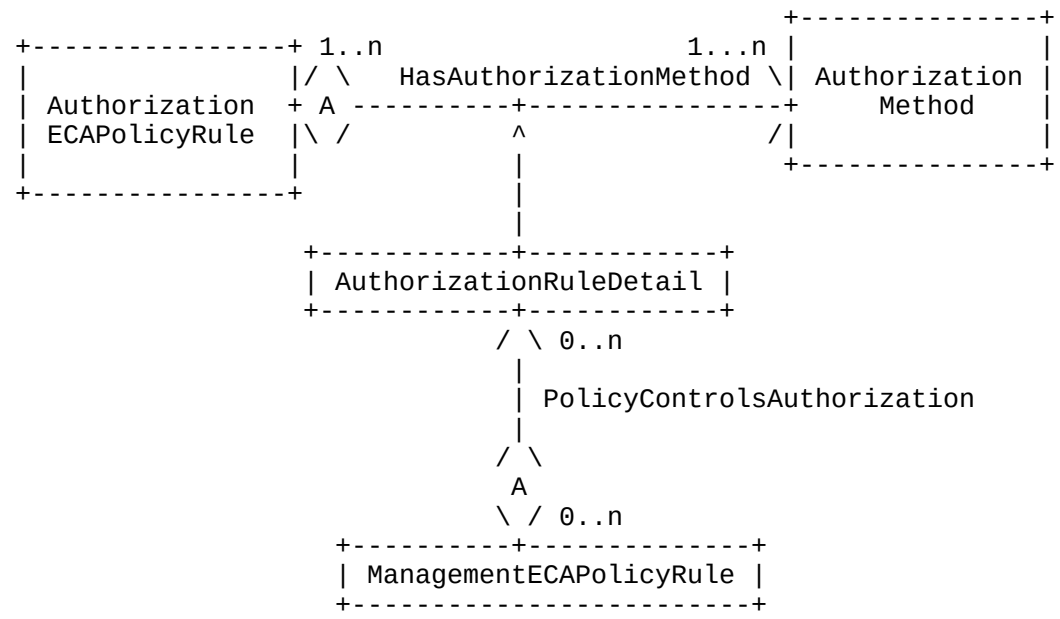


Figure 11. Modeling Authorization Mechanisms

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:

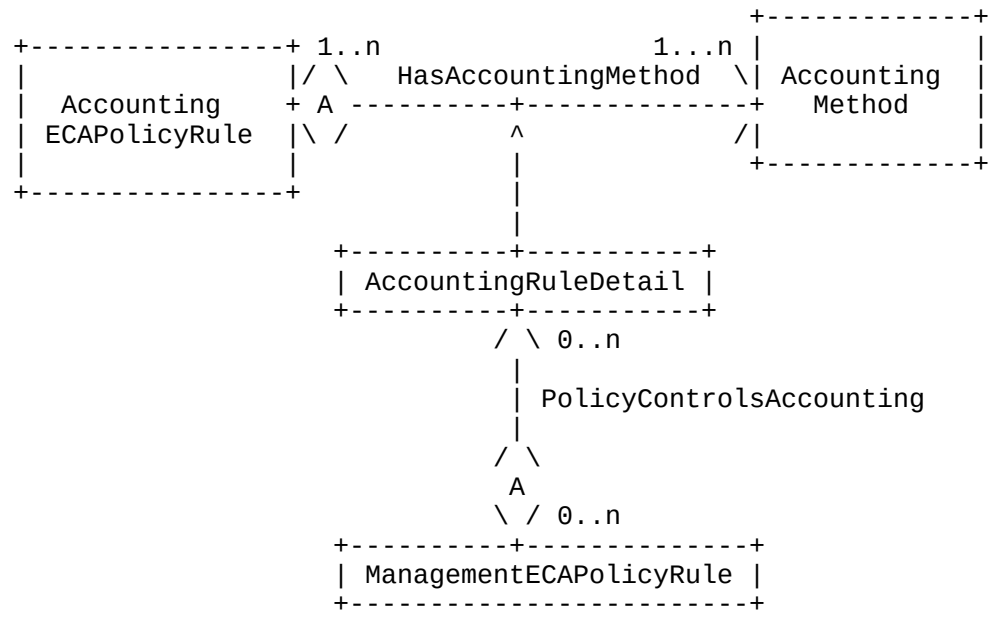


Figure 12. Modeling Accounting Mechanisms

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:

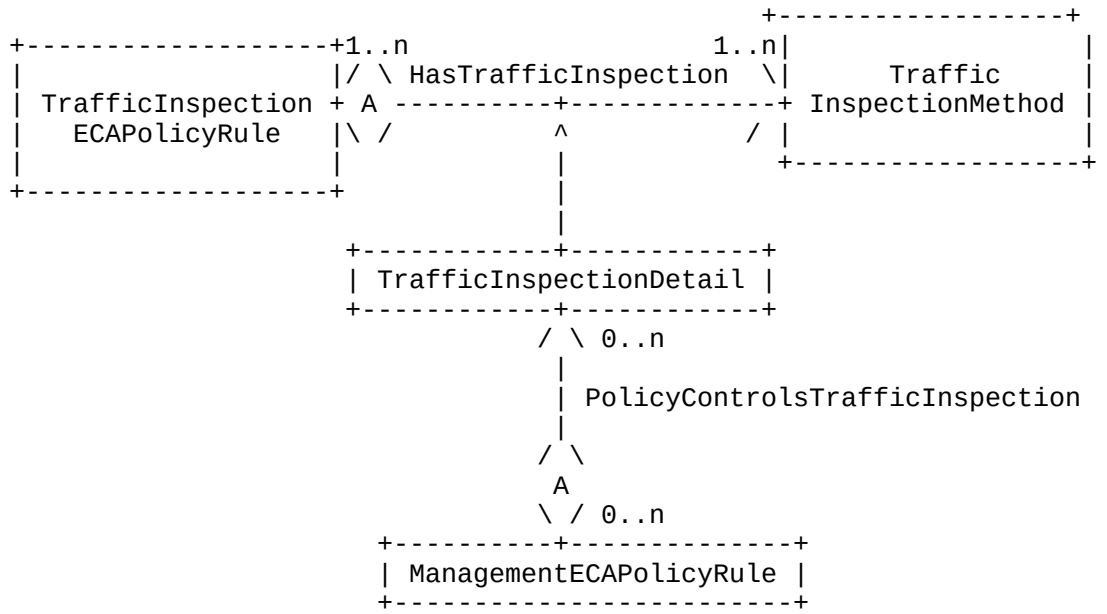


Figure 13. Modeling Traffic Inspection Mechanisms

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:

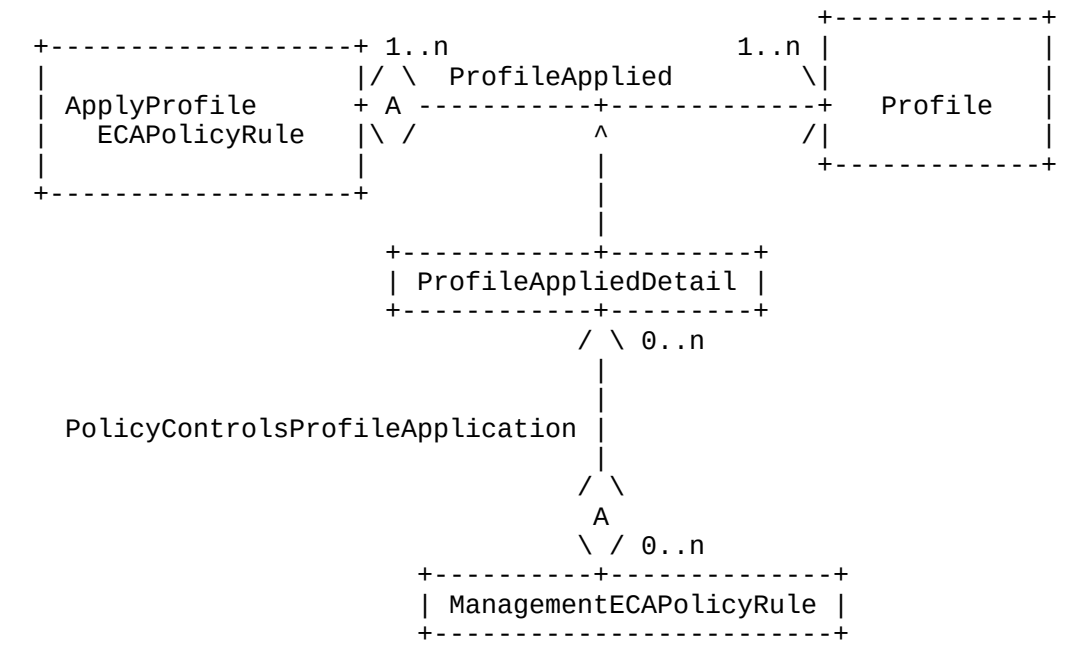


Figure 14. Modeling Profile ApplicationMechanisms

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.

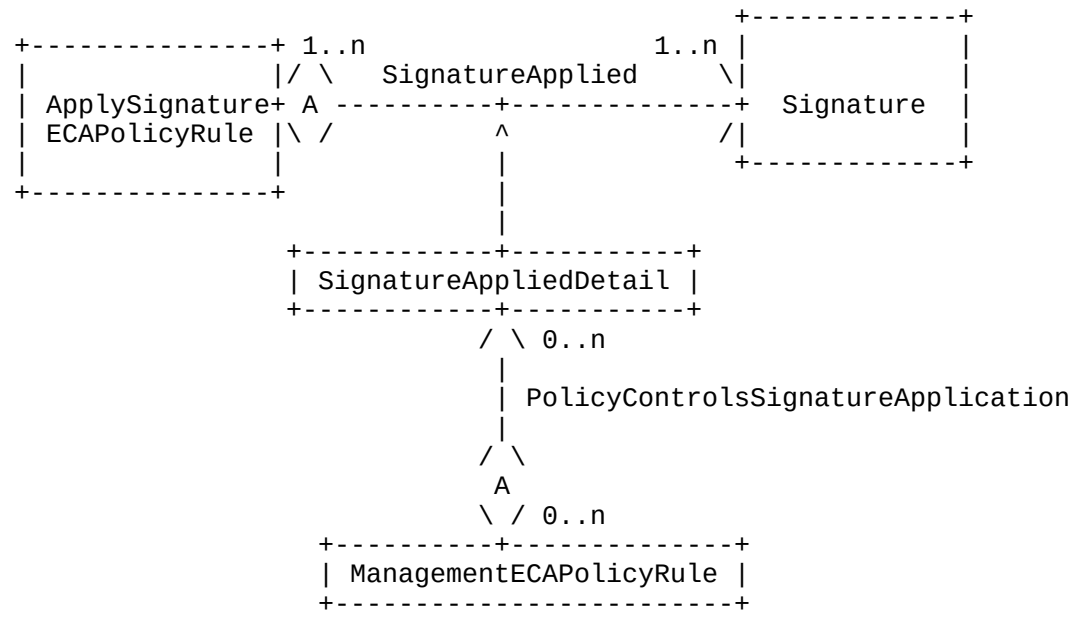


Figure 15. Modeling Sginature Application Mechanisms

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.

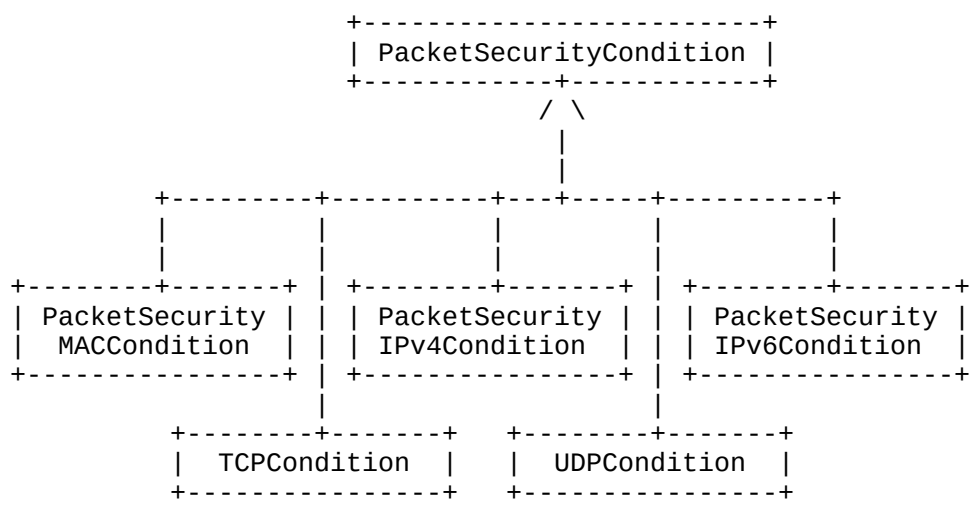


Figure 16. Network Security Info Sub-Model PacketSecurityCondition Class Extensions

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.

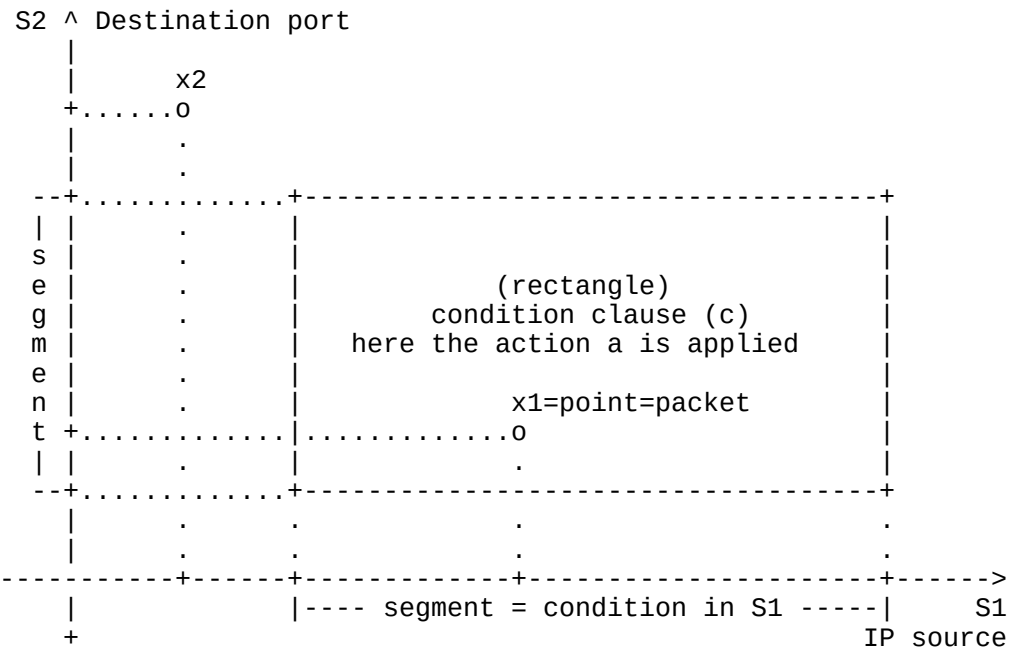


Figure 17: Geometric representation of a rule r=(c,a) that matches x1, but does not match x2.

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: April 27, 2019

L. Xia
D. Zhang
Huawei
Y. Wu
Aliababa Group
R. Kumar
A. Lohiya
Juniper Networks
H. Birkholz
Fraunhofer SIT

October 24, 2018

An Information Model for the Monitoring of Network Security Functions (NSF)
draft-zhang-i2nsf-info-model-monitoring-07

Abstract

The Network Security Functions (NSF) NSF-facing interface exists between the Service Provider's management system (or Security Controller) and the NSF to enforce security policy provisioning and network security status monitoring. This document focuses on the monitoring part and defines the corresponding 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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on April 27, 2019.

Table of Contents

1. Introduction

According to [I-D.ietf-i2nsf-terminology], the interface provided by an NSF (e.g., FW, IPS, Anti-DDOS, or Anti-Virus function) to administrative entities (e.g., NMS, security controller) to enable remote management (i.e. configuring and monitoring) is referred to as an "I2NSF NSF-Facing Interface". Monitoring procedures intent to acquire vital types of data at rest with respect to NSF, e.g. alarms, records, or counters, via data in motion, e.g. queries, notifications, or events. The monitoring of NSF plays an important role in the overall security framework, if done in a timely and comprehensive way. The monitoring information generated by an NSF can very well be an early indication of anomalous behavior or malicious activity, such as denial of service attacks.

This draft defines a comprehensive NSF monitoring information model that provides visibility into NSF. This document will not go into the design details of NSF-Facing Interfaces. Instead, it is focused on specifying the information and illustrates the methods that enable 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 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 NSF as listed below:

- The security administrator can configure a policy that is triggered on a specific event occurring in the NSF or the network. If a security controller detects the specified event, it configures additional security functions as defined by policies.

- The events triggered by NSF as a result of security policy violation can be used by SIEM to detect any suspicious activity in a larger correlation context.

- The events and activity logs from NSF can be used to build advanced analytics, such as behavior and predictive models to improve security posture in large deplyoments.

- 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 the root cause analysis of an operational issue - therefore improve debugging.

- 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 and therefore observable information. This enables security admins to assess the state of the network topology in a timely fashion. It is not possible to block all the internal and external threats based on static security posture. A more practical approach is supported by enabling dynamic security measures - for which continuous 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 may 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.

5.1. Information Types and Acquisition Methods

In this document most information types defined, benefit from high visibility with respect to value changes, e.g. alarms or records. In contrast, values that change monotonically in a continuous way do not

benefit from this high visibility. On the contrary, emitting each change would result in a useless amount of value updates. Hence, values, such as counter, are best acquired in periodic intervals.

The mechanism provided by YANG Push [I-D.ietf-netconf-yang-push] and YANG Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] address exactly these set of requirements. YANG also enables semantically well-structured information, as well as subscriptions to datatores or event streams - on-change or periodically.

In consequence, this information model is intended to support data models used in solicited or unsolicited event streams that potentially are facilitated by subscription mechanism. A subset of information elements defined in the information model address this domain of application.

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.

7.1. System Alarm

Characteristics:

- acquisition_method: subscription
- emission_type: on-change
- dampening_type: no-dampening

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

Characteristics:

- acquisition_method: subscription

- emission_type: on-change
- dampening_type: on_repetition

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

Characteristics:

- acquisition_method: subscription
- emission_type: on-change

- dampening_type: on_repetition

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

Characteristics:

- acquisition_method: subscription or query
- emission_type: periodical
- dampening_type: none

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

Characteristics:

- acquisition_method: subscription
- emission_type: on-change
- dampening_type: none

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
- 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.
- attack_rate: The PPS of attack traffic
- attack_speed: the bps of attack traffic
- rule_id: The ID of the rule being triggered
- rule_name: The name of the rule being triggered
- profile: Security profile that traffic matches.

7.5.2. Session Table Event

The following information should be included in a Session Table Event:

- event_name: 'SESSION_USAGE_HIGH'
- current: The number of concurrent sessions
- max: The maximum number of sessions that the session table can support
- threshold: The threshold triggering the event
- message: 'The number of session table exceeded the threshold'

7.5.3. Virus Event

The following information should be included in a Virus Event:

- event_Name: 'SEC_EVENT_VIRUS'

- virus_type: Type of the virus, e.g., trojan, worm, macro Virus type
- virus_name
- dst_ip: The destination IP address of the packet where the virus is found
- src_ip: The source IP address of the packet where the virus is found
- 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

Characteristics:

- acquisition_method: subscription
- emission_type: on-change
- dampening_type: on_repetition

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

Characteristics:

- acquisition_method: subscription or query
- emission_type: periodical
- dampening_type: none

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

- o 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

[I-D.ietf-core-yang-cbor]
Veillette, M., Pelov, A., Somaraju, A., Turner, R., and A. Minaburo, "CBOR Encoding of Data Modeled with YANG", draft-ietf-core-yang-cbor-07 (work in progress), September 2018.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-17 (work in progress), September 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-20 (work in progress), October 2018.

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

[RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <https://www.rfc-editor.org/info/rfc3877>.

[RFC4765] Debar, H., Curry, D., and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)", RFC 4765, DOI 10.17487/RFC4765, March 2007, <https://www.rfc-editor.org/info/rfc4765>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7011>.

10.2. 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", draft-ietf-i2nsf-framework-10 (work in progress), November 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-06 (work in progress), July 2018.

[I-D.xia-i2nsf-capability-interface-im]
Xia, L., Strassner, J., Li, K., Zhang, D., Lopez, E., Bouthors, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-06 (work in progress), July 2016.

[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, <https://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