

I2NSF
Internet-Draft
Intended status: Experimental
Expires: November 5, 2017

R. Marin-Lopez
G. Lopez-Millan
University of Murcia
May 4, 2017

Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-abad-i2nsf-sdn-ipsec-flow-protection-03

Abstract

This document describes the use case of providing IPsec-based flow protection by means of a Software-Defined Network (SDN) controller (aka. Security Controller) and establishes the requirements to support this service. It considers two main well-known scenarios in IPsec: (i) gateway-to-gateway and (ii) host-to-host. This document describes a mechanism based on the SDN paradigm to support the distribution and monitoring of IPsec information from a SDN controller to one or several flow-based Network Security Function (NSF). The NSFs implement IPsec to protect data traffic between network resources with IPsec.

The document focuses in the NSF Facing Interface by providing models for Configuration and State data model required to allow the Security Controller to configure the IPsec databases (SPD, SAD, PAD) and IKE to establish security associations with a reduced intervention of the network administrator. NOTE: State data model will be developed as part of this work but it is still TBD.

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 November 5, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. Terminology	4
4. Objectives	6
5. SDN-based IPsec management description	6
5.1. Case 1: IKE/IPsec in the NSF	6
5.1.1. Interface Requirements for Case 1	7
5.2. Case 2: IPsec (no IKE) in the NSF	7
5.2.1. Interface Requirements for Case 2	8
5.3. Case 1 vs Case 2	8
6. YANG configuration data models	9
6.1. Security Policy Database (SPD) Model	10
6.2. Security Association Database (SAD) Model	11
6.3. Peer Authorization Database (PAD) Model	13
6.4. Internet Key Exchange (IKE) Model	14
7. Use cases examples	16
7.1. Host-to-Host or Gateway-to-gateway under the same controller	16
7.2. Host-to-Host or Gateway-to-gateway under different Security controllers	18
8. Implementation notes	20
9. Security Considerations	21
10. Acknowledgements	21
11. References	21
11.1. Normative References	21
11.2. Informative References	22
Appendix A. Appendix A: YANG model IPsec Configuration data . .	25
Authors' Addresses	45

1. Introduction

Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software. SDN paradigm relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications. The SDN application can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via this interface [RFC7149][ITU-T.Y.3300][ONF-SDN-Architecture][ONF-OpenFlow].

Typically, traditional IPsec VPN concentrators and, in general, entities (i.e. hosts or security gateways) supporting IKE/IPsec, must be configured directly by the administrator. This makes the IPsec security association (SA) management difficult and generates a lack of flexibility, specially if the number of security policies and SAs to handle is high. With the growth of SDN-based scenarios where network resources are deployed in an autonomous manner, a mechanism to manage IPsec SAs according to the SDN architecture becomes more relevant. Thus, the SDN-based service described in this document will autonomously deal with IPsec SAs management.

An example of usage can be the notion of Software Defined WAN (SD-WAN), SDN extension providing a software abstraction to create secure network overlays over traditional WAN and branch networks. SD-WAN is based on IPsec as underlying security protocol and aims to provide flexible, automated, fast deployment and on-demand security network services.

IPsec architecture [RFC4301] defines a clear separation between the processing to provide security services to IP packets and the key management procedures to establish the IPsec security associations. In this document, we define a service where the key management procedures can be carried by an external entity: the Security Controller.

First, this document exposes the requirements to support the protection of data flows using IPsec [RFC4301]. We have considered two general cases:

- 1) The Network Security Function (NSF) implements the Internet Key Exchange (IKE) protocol and the IPsec databases: the Security Policy Database (SPD), the Security Association Database (SAD) and the Peer Authorization Database (PAD). The Security

Controller is in charge of provisioning the NSF with the required information to IKE, the SPD and the PAD.

- 2) The NSF only implements the IPsec databases (no IKE implementation). The Security Controller will provide the required parameters to create valid entries in the SPD and the SAD into the NSF. Therefore, the NSF will have only support for IPsec while automated key management functionality is moved to the controller.

In both cases, an interface/protocol is required to carry out this provisioning between the Security Controller and the NSF. In particular, Case 1 requires the provision of SPD and PAD entries and the IKE credential and information related with the IKE negotiation (e.g. `IKE_SA_INIT`); and Case 2 requires the management of SPD and SAD entries. Based on YANG models in [netconf-vpn] and [I-D.tran-ipsecme-yang], RFC 4301 [RFC4301] and RFC 7296 [RFC7296] this document defines the required interfaces with a YANG model for configuration data for IKE, PAD, SPD and SAD Appendix A . State data is TBD.

This document considers two typical scenarios to manage autonomously IPsec SAs: gateway-to-gateway and host-to-host [RFC6071]. The analysis of the host-to-gateway (roadwarrior) scenario is TBD. In these cases, host or gateways or both may act as NSFs. Finally, it also discusses the situation where two NSFs are under the control of two different Security Controllers.

NOTE: This work pays attention to the challenge "Lack of Mechanism for Dynamic Key Distribution to NSFs" defined in [I-D.ietf-i2nsf-problem-and-use-cases] in the particular case of the establishment and management of IPsec SAs. In fact, this I-D could be considered as a proper use case for this particular challenge in [I-D.ietf-i2nsf-problem-and-use-cases].

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]. When these words appear in lower case, they have their natural language meaning.

3. Terminology

This document uses the terminology described in [RFC7149], [RFC4301], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow],

[ITU-T.X.1252], [ITU-T.X.800] and [I-D.ietf-i2nsf-terminology]. In addition, the following terms are defined below:

- o Software-Defined Networking. A set of techniques enabling 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].
- o Flow/Data Flow. Set of network packets sharing a set of characteristics, for example IP dst/src values or QoS parameters.
- o Security Controller. A Controller is a management Component that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. In the context of this document, it provides IPsec management information.
- o Network Security Function (NSF). Software that provides a set of security-related services.
- o Flow-based NSF. A NSF that inspects network flows according to a set of policies intended for enforcing security properties. The NSFs considered in this document falls into this classification.
- o Flow-based Protection Policy. The set of rules defining the conditions under which a data flow MUST be protected with IPsec, and the rules that MUST be applied to the specific flow.
- o Internet Key Exchange (IKE) v2 Protocol to establish IPsec Security Associations (SAs). It requires information about the required authentication method (i.e. preshared keys), DH groups, modes and algorithms for IKE SA negotiation, etc.
- o Security Policy Database (SPD). It includes information about IPsec policies direction (in, out), local and remote addresses, inbound and outbound SAs, etc.
- o Security Associations Database (SAD). It includes information about IPsec SAs, such as SPI, destination addresses, authentication and encryption algorithms and keys to protect IP flow.
- o Peer Authorization Database (PAD). It provides the link between the SPD and a security association management protocol such as IKE or our SDN-based solution.

4. Objectives

- o To describe the architecture for the SDN-based IPsec management, which implements a security service to allow the establishment and management of IPsec security associations from a central point to protect specific data flows.
- o To define the interfaces required to manage and monitor the IPsec Security Associations in the NSF from a Security Controller. YANG models are defined for configuration and state data for IPsec management.

5. SDN-based IPsec management description

As mentioned in Section 1, two cases are considered:

5.1. Case 1: IKE/IPsec in the NSF

In this case the NSF ships an IKE implementation besides the IPsec support. The Security Controller is in charge of managing and applying SPD and PAD entries (deriving and delivering IKE Credentials such as a pre-shared key, certificates, etc.), and applying other IKE configuration parameters (e.g. IKE_SA_INIT algorithms) to the NSF for the IKE negotiation.

With these entries, the IKE implementation can operate to establish the IPsec SAs. The application (administrator) establishes the IPsec requirements and information about the end points information (through the Client Facing Interface), and the Security Controller translates those requirements into SPD and PAD entries that will be installed into the NSF (through the NSF Facing Interface). With that information, the NSF can just run IKE to establish the required IPsec SA (when the data flow needs protection). Figure 1 shows the different layers and corresponding functionality.

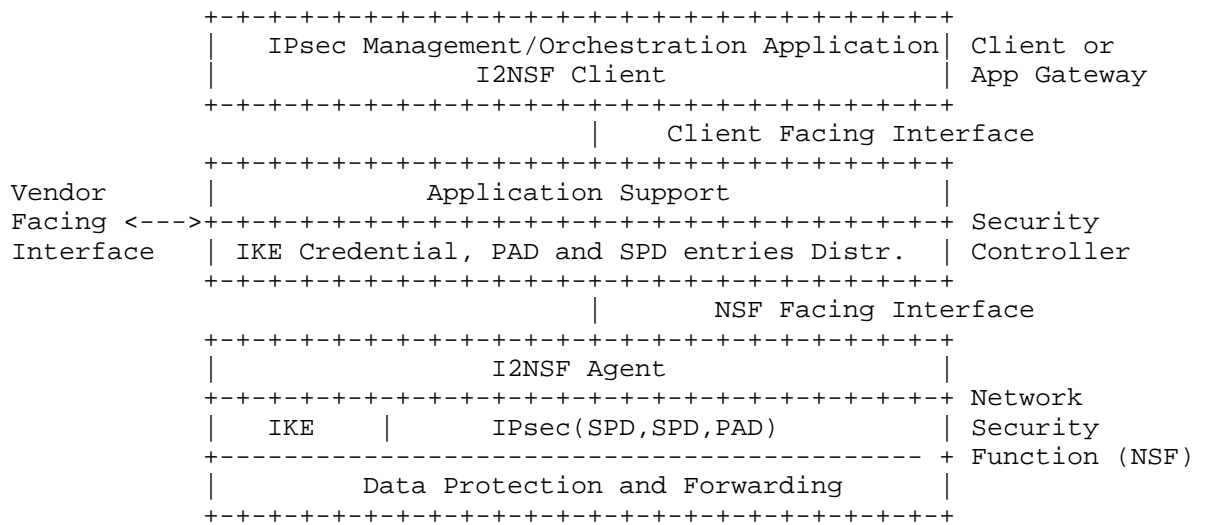


Figure 1: Case 1: IKE/IPsec in the NSF

5.1.1. Interface Requirements for Case 1

SDN-based IPsec flow protection services provide dynamic and flexible management of IPsec SAs in flow-based NSF. In order to support this capability in case 1, the following interface requirements are to be met:

- o A YANG data model for Configuration data for IKE, SPD and PAD.
- o A YANG data model for State data for IKE, SPD, PAD and SAD (Note that SAD entries are created in runtime by IKE.)
- o In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface is required to exchange IPsec-related information.

5.2. Case 2: IPsec (no IKE) in the NSF

In this case the NSF does not deploy IKE and, therefore, the Security Controller has to perform the management of IPsec SAs by populating and monitoring the SPD and the SAD.

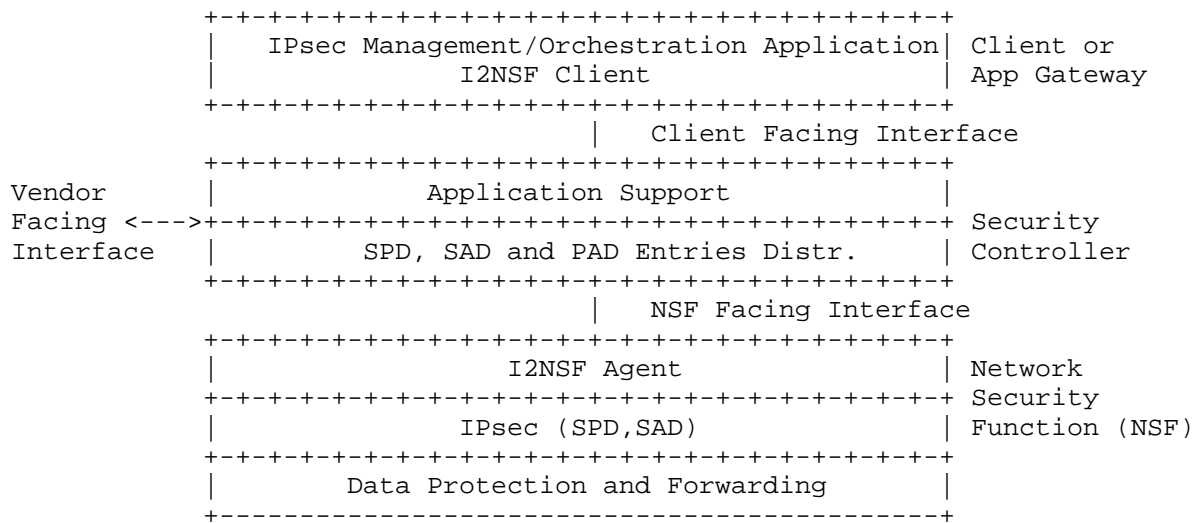


Figure 2: Case 2: IPsec (no IKE) in the NSF

As shown in Figure 2, applications for flow protection run on the top of the Security Controller. When an administrator enforces flow-based protection policies through the Client Facing Interface, the Security Controller translates those requirements into SPD and SAD entries, which are installed in the NSF. PAD entries are not required since there is no IKE in the NSF.

5.2.1. Interface Requirements for Case 2

In order to support case 2, the following requirements are to be met:

- o A YANG data model for Configuration data for SPD and SAD.
- o A YANG data model for State data for SPD and SAD.
- o In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface is required to exchange IPsec-related information.

5.3. Case 1 vs Case 2

Case 1 MAY be easier to deploy than Case 2 because current gateways typically have an IKE/IPsec implementation. Moreover hosts can install easily an IKE implementation. As downside, the NSF needs more resources to hold IKE. Moreover, the IKE implementation needs to

implement an interface so that the I2NSF Agent can interact with them.

Alternatively, Case 2 allows lighter NSFs (no IKE implementation), which benefits the deployment in constrained NSFs. Moreover, IKE does not need to be performed in gateway-to-gateway and host-to-host scenarios under the same Security Controller (see Section 7.1). On the contrary, the overload of creation of fresh IPsec SA is shifted to the Security Controller since IKE is not in the NSF. As a consequence, this may result in a more complex implementation in the controller side.

For example, the Security Controller needs to supervise the IPsec SAs states and take care of the rekeying process so that, after some period of time (e.g. IPsec SA soft lifetime), it has to create a new IPsec SA and remove the old one. Or the Security Controller needs to process events coming from the I2NSF when for example an IPsec SA is requested (e.g. acquire or expire events). Another example is the NAT traversal support. In general, the SDN paradigm assumes the SDN controller has a view of the network it controls. This view is built either requesting information to the NSFs under its control or because these NSFs inform the SDN controller. Based on this information, the SDN/security controller can guess if there is a NAT configured between two hosts, apply the required policies to both NSFs besides activating the usage of UDP encapsulation of ESP packets [RFC3948].

In those scenarios, the Controller could directly request the NSF for specific data such as networking configuration, NAT support, etc. Protocols such as NETCONF or SNMP can be used here. For example, RFC 7317 [RFC7317] provides a YANG data model for system management or [I-D.sivakumar-yang-nat] a data model for NAT management.

6. YANG configuration data models

In order to support case 1 and case 2 we have modelled the different parameters and values that must be configured to manage IPsec SAs. Specifically, case 1 requires modelling IKEv2, SPD and PAD while case 2 requires models for the SPD and SAD. A single YANG file represents both cases though some part of the models are selectively activated depending a feature defined in the YANG file. For example, the IKE configuration is not enabled in case 2.

In the following, we summarize, by using a tree representation, the different configuration data models (NOTE: State data models are TBD though they are expected to be very similar to the model defined here). The complete YANG configuration data model is in Appendix A

6.1. Security Policy Database (SPD) Model

The definition of this model has been extracted from the specification in section 4.4.1 and Appendix D in [RFC4301]

```

+--rw spd
|   +--rw spd-entry* [rule-number]
|   |   +--rw rule-number      uint64
|   |   +--rw priority?       uint32
|   |   +--rw names* [name]
|   |   |   +--rw name-type?   ipsec-spd-name
|   |   |   +--rw name        string
|   |   +--rw condition
|   |   |   +--rw traffic-selector-list* [ts-number]
|   |   |   |   +--rw ts-number      uint32
|   |   |   |   +--rw direction?    ipsec-traffic-direction
|   |   |   |   +--rw local-addresses* [start end]
|   |   |   |   |   +--rw start      inet:ip-address
|   |   |   |   |   +--rw end        inet:ip-address
|   |   |   |   +--rw remote-addresses* [start end]
|   |   |   |   |   +--rw start      inet:ip-address
|   |   |   |   |   +--rw end        inet:ip-address
|   |   |   |   +--rw next-layer-protocol* ipsec-next-layer-proto
|   |   |   |   +--rw local-ports* [start end]
|   |   |   |   |   +--rw start      inet:port-number
|   |   |   |   |   +--rw end        inet:port-number
|   |   |   |   +--rw remote-ports* [start end]
|   |   |   |   |   +--rw start      inet:port-number
|   |   |   |   |   +--rw end        inet:port-number
|   |   |   |   +--rw selector-priority? uint32
|   |   +--rw processing-info
|   |   |   +--rw action          ipsec-spd-operation
|   |   |   +--rw ipsec-sa-cfg
|   |   |   |   +--rw pfp-flag?    boolean
|   |   |   |   +--rw extSeqNum?   boolean
|   |   |   |   +--rw seqOverflow? boolean
|   |   |   |   +--rw statefulfragCheck? boolean
|   |   |   |   +--rw security-protocol? ipsec-protocol
|   |   |   |   +--rw mode?        ipsec-mode
|   |   |   |   +--rw ah-algorithms
|   |   |   |   |   +--rw ah-algorithm* integrity-algorithm-t
|   |   |   |   +--rw esp-algorithms
|   |   |   |   |   +--rw authentication* integrity-algorithm-t
|   |   |   |   |   +--rw encryption* encryption-algorithm-t
|   |   +--rw tunnel

```

```

|         |--rw local?          inet:ip-address
|         |--rw remote?        inet:ip-address
|         |--rw bypass-df?     boolean
|         |--rw bypass-dscp?   boolean
|         |--rw dscp-mapping?  yang:hex-string
|         |--rw ecn?           boolean
|--rw spd-lifetime
|   |--rw time-soft?           uint32
|   |--rw time-hard?           uint32
|   |--rw time-use-soft?       uint32
|   |--rw time-use-hard?       uint32
|   |--rw byte-soft?           uint32
|   |--rw byte-hard?           uint32
|   |--rw packet-soft?         uint32
|   |--rw packet-hard?         uint32

```

The definition of this model has been extracted from the specification in section 4.4.2 in [RFC4301]

```

+---rw sad {case2}?
+---rw sad-entry* [spi]
+---rw spi ipsec-spi
+---rw seq-number? uint64
+---rw seq-number-overflow-flag? boolean
+---rw anti-replay-window? uint16
+---rw rule-number? uint32
+---rw local-addresses* [start end]
|   +---rw start inet:ip-address
|   +---rw end inet:ip-address
+---rw remote-addresses* [start end]
|   +---rw start inet:ip-address
|   +---rw end inet:ip-address
+---rw next-layer-protocol* ipsec-next-layer-protocol
+---rw local-ports* [start end]
|   +---rw start inet:port-number
|   +---rw end inet:port-number
+---rw remote-ports* [start end]
|   +---rw start inet:port-number
|   +---rw end inet:port-number
+---rw security-protocol? ipsec-protocol
+---rw ah-sa

```

```

|   +--rw integrity-algorithm?    integrity-algorithm-t
|   +--rw key?                    string
+--rw esp-sa
|   +--rw encryption
|   |   +--rw encryption-algorithm?    encryption-algorithm-t
|   |   +--rw key?                      string
|   |   +--rw iv?                       string
|   +--rw integrity
|   |   +--rw integrity-algorithm?    integrity-algorithm-t
|   |   +--rw key?                      string
|   +--rw combined
|   |   +--rw combined-algorithm?    combined-algorithm-t
+--rw sa-lifetime
|   +--rw time-soft?              uint32
|   +--rw time-hard?              uint32
|   +--rw time-use-soft?          uint32
|   +--rw time-use-hard?          uint32
|   +--rw byte-soft?              uint32
|   +--rw byte-hard?              uint32
|   +--rw packet-soft?            uint32
|   +--rw packet-hard?            uint32
|   +--rw action?                  lifetime-action
+--rw mode?                        ipsec-mode
+--rw statefulfragCheck?           boolean
+--rw dscp?                         yang:hex-string
+--rw tunnel
|   +--rw local?                   inet:ip-address
|   +--rw remote?                  inet:ip-address
|   +--rw bypass-df?               boolean
|   +--rw bypass-dscp?             boolean
|   +--rw dscp-mapping?            yang:hex-string
|   +--rw ecn?                     boolean
+--rw path-mtu?                     uint16
+--rw encaps
|   +--rw espinudp?                boolean
|   +--rw sport?                   inet:port-number
|   +--rw dport?                   inet:port-number
|   +--rw oaddr?                   inet:ip-address

```

```

rpcs:
+---x sadb_register
+---w input
|   +---w base-list* [version]
|   |   +---w version      string
|   |   +---w msg_type?    sadb-msg-type
|   |   +---w msg_satype?  sadb-msg-satype
|   |   +---w msg_seq?    uint32
+---ro output
+---ro base-list* [version]
|   +---ro version      string
|   +---ro msg_type?    sadb-msg-type
|   +---ro msg_satype?  sadb-msg-satype
|   +---ro msg_seq?    uint32
+---ro algorithm-supported*
+---ro authentication
|   +---ro name?        integrity-algorithm-t
|   +---ro ivlen?       uint8
|   +---ro min-bits?    uint16
|   +---ro max-bits?    uint16
+---ro encryption
|   +---ro name?        encryption-algorithm-t
|   +---ro ivlen?       uint8
|   +---ro min-bits?    uint16
|   +---ro max-bits?    uint16

notifications:
+---n spd-expire
|   +---ro index?    uint64
+---n sadb_acquire
|   +---ro state      uint32
+---n sadb_expire
|   +---ro state      uint32

```

6.3. Peer Authorization Database (PAD) Model

The definition of this model has been extracted from the specification in section 4.4.3 in [RFC4301] (NOTE: We have observed that many implementations integrate PAD configuration as part of the IKE configuration.)

```

+--rw pad {case1}?
  +--rw pad-entries* [pad-entry-id]
    +--rw pad-entry-id          uint64
    +--rw (identity)?
      | +--:(ipv4-address)
      | | +--rw ipv4-address?      inet:ipv4-address
      | +--:(ipv6-address)
      | | +--rw ipv6-address?      inet:ipv6-address
      | +--:(fqdn-string)
      | | +--rw fqdn-string?       inet:domain-name
      | +--:(rfc822-address-string)
      | | +--rw rfc822-address-string? string
      | +--:(dnX509)
      | | +--rw dnX509?            string
      | +--:(id_key)
      | | +--rw id_key?            string
    +--rw pad-auth-protocol?     auth-protocol-type
    +--rw auth-method
    +--rw auth-m?                auth-method-type
    +--rw pre-shared
    | +--rw secret?              string
    +--rw rsa-signature
    +--rw key-data?              string
    +--rw key-file?              string
    +--rw ca-data*               string
    +--rw ca-file?               string
    +--rw cert-data?             string
    +--rw cert-file?             string
    +--rw crl-data?              string
    +--rw crl-file?              string

```

6.4. Internet Key Exchange (IKE) Model

The model related to IKEv2 has been extracted from reading IKEv2 standard in [RFC7296], and observing some open source implementations, such as Strongswan or Libreswan.

```

+--rw ikev2 {case1}?
|   +--rw ike-connection
|   |   +--rw ike-conn-entries* [conn-name]
|   |   |   +--rw conn-name          string
|   |   |   +--rw autostartup         boolean
|   |   |   +--rw nat-traversal?      boolean
|   |   |   +--rw version?            enumeration
|   |   |   +--rw phase1-lifetime     uint32
|   |   |   +--rw phase1-authby       auth-method-type
|   |   |   +--rw phase1-authalg*     integrity-algorithm-t
|   |   |   +--rw phase1-encalg*      encryption-algorithm-t
|   |   |   +--rw dh_group             uint32
|   |   |   +--rw local
|   |   |   |   +--rw (my-identifier-type)?
|   |   |   |   |   +--:(ipv4)
|   |   |   |   |   |   +--rw ipv4?          inet:ipv4-address
|   |   |   |   |   +--:(ipv6)
|   |   |   |   |   |   +--rw ipv6?          inet:ipv6-address
|   |   |   |   |   +--:(fqdn)
|   |   |   |   |   |   +--rw fqdn?          inet:domain-name
|   |   |   |   |   +--:(dn)
|   |   |   |   |   |   +--rw dn?            string
|   |   |   |   |   +--:(user_fqdn)
|   |   |   |   |   |   +--rw user_fqdn?     string
|   |   |   |   +--rw my-identifier      string
|   |   +--rw remote
|   |   |   +--rw (my-identifier-type)?
|   |   |   |   +--:(ipv4)
|   |   |   |   |   +--rw ipv4?          inet:ipv4-address
|   |   |   |   +--:(ipv6)
|   |   |   |   |   +--rw ipv6?          inet:ipv6-address
|   |   |   |   +--:(fqdn)
|   |   |   |   |   +--rw fqdn?          inet:domain-name
|   |   |   |   +--:(dn)
|   |   |   |   |   +--rw dn?            string
|   |   |   |   +--:(user_fqdn)
|   |   |   |   |   +--rw user_fqdn?     string
|   |   |   +--rw my-identifier      string
|   +--rw local-addr               inet:ip-address
|   +--rw remote-addr              inet:ip-address
|   +--rw pfs_group?               uint32
|   +--rw phase2-lifetime          uint32
|   +--rw phase2-authalg*          integrity-algorithm-t
|   +--rw phase2-encalg*           encryption-algorithm-t

```

7. Use cases examples

This section explains how different traditional configurations, that is, host-to-host and gateway-to-gateway are deployed using our SDN-based IPsec management service. In turn, these configurations will be typical in modern networks where, for example, virtualization will be key.

7.1. Host-to-Host or Gateway-to-gateway under the same controller

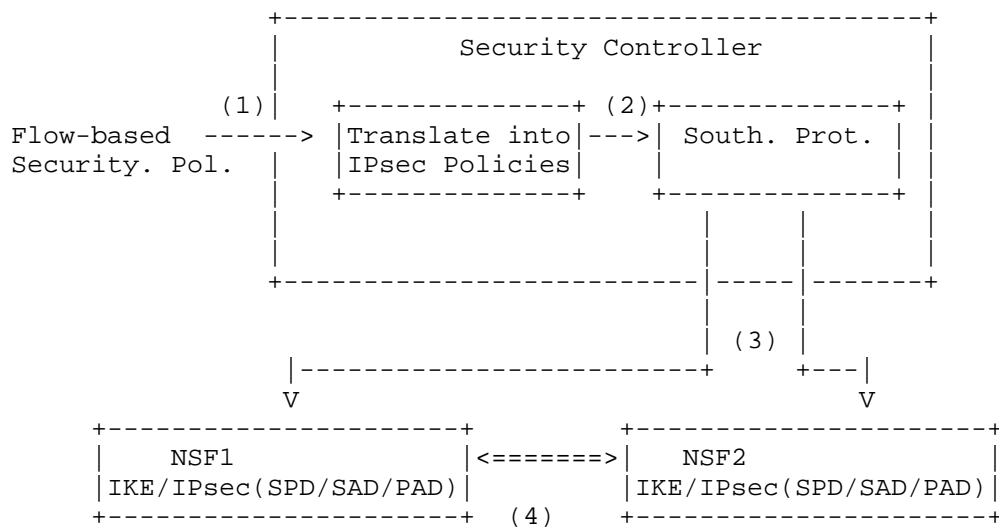


Figure 3: Gateway-to-Gateway single controller flow for case 1 .

Figure 3 describes the case 1:

1. The administrator defines general flow-based security policies. The controller looks for the NSFs involved (NSF1 and NSF2).
2. The controller generates IKE credentials for them and translates the policies into SPD and PAD entries.
3. The controller inserts the SPD and PAD entries in both NSF1 and NSF2.
4. The flow is protected with the IPsec SA established with IKEv2.

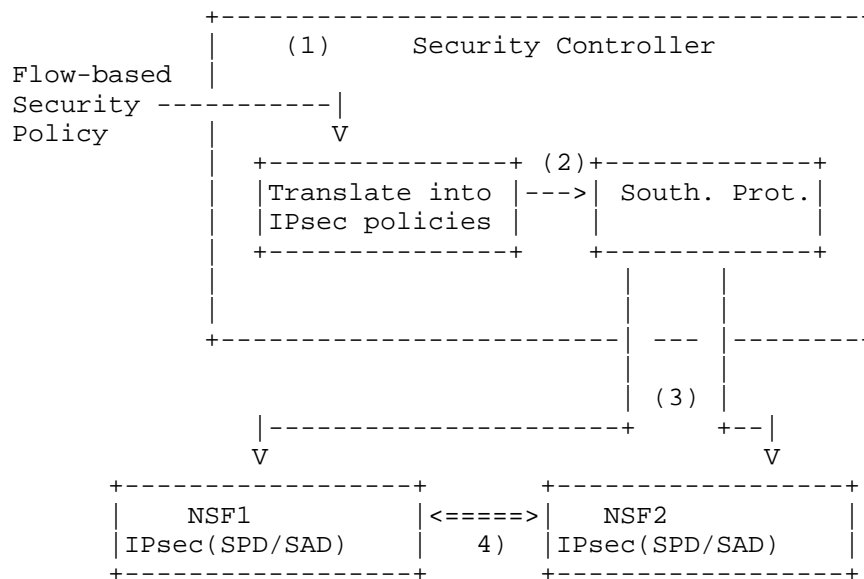


Figure 4: Host-to-Host / Gateway-to-Gateway single controller flow for case 2.

In case 2, flow-based security policies defined by the administrator are also translated into IPsec SPD entries and inserted into the corresponding NSFs. Besides, fresh SAD entries will be also generated by the controller and enforced in the NSFs. In this case the execution of IKE is not necessary in the controller, and it provides the cryptographic material for the IPsec SAs. These keys will be also distributed securely through the southbound interface. Note that this is possible because both NSFs are managed by the same controller.

Figure 4 describes the case 2, when a data packet needs to be protected in the path between the NSF1 and NSF2:

1. The administrator establishes the flow-based security policies. The controller looks for the involved NSFs.
2. The controller translates the flow-based security policies into IPsec SPD and SAD entries.
3. The controller inserts the these entries in both NSF1 and NSF2 IPsec databases.
4. The flow is protected with the IPsec SA established by the Security Controller.

Both NSF's could be two hosts that exchange traffic and require to establish an end-to-end security association to protect their communications (host-to-host) or two gateways (gateway-to-gateway)), for example, within an enterprise that needs to protect the traffic between, for example, the networks of two branch offices.

Applicability of these configurations appear in current and new networking scenarios. For example, SD-WAN technologies are providing dynamic and on-demand VPN connections between branch offices or between branches and SaaS cloud services. Beside, IaaS services providing virtualization environments are deployments solutions based on IPsec to provide secure channels between virtual instances (Host-to-Host) and providing VPN solutions for virtualized networks (Gateway-to-Gateway).

In general (for case 1 and case 2), this system presents various advantages:

1. It allows to create a IPsec SA among two NSF's, with only the application of more general flow-based security policies at the application layer. Thus, an administrator/s can manage all security associations in a centralized point with an abstracted view of the network;
2. All NSF's deployed after the application of the new policies are NOT manually configured, therefore allowing its deployment in an automated manner.

7.2. Host-to-Host or Gateway-to-gateway under different Security controllers

It is also possible that two NSF's (i.e. NSF1 and NSF2) are under the control of two different security controllers. This may happen, for example, when two organizations, namely Enterprise A and Enterprise B, have their headquarters interconnected through a WAN connection and they both have deployed a SDN-based architecture to provide connectivity to all their clients.

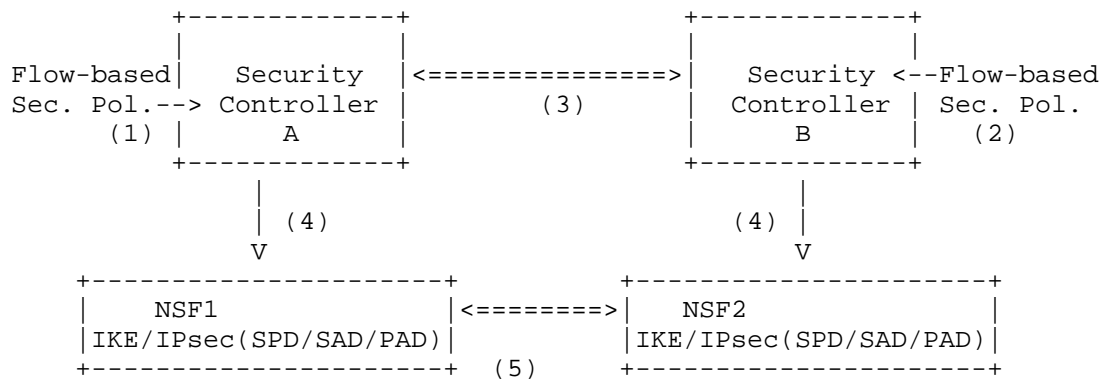


Figure 5: Different security controllers in Case 1

Figure 5 describes case 1 when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow-based Security Policies in Security Controller A.
2. The B's administrator establishes general Flow-based Security Policies in Security Controller B.
3. The Security Controller A realizes that protection is required between the NSF1 and NSF2, but the NSF2 is under the control of another Security Controller (Security Controller B), so it starts negotiations with the other controller to agree on the IPsec SPD policies and IKE credentials for their respective NSFs. NOTE: This may require extensions in the East/West interface.
4. Then, both Security Controllers enforce the IKE credentials and related parameters and the SPD and PAD entries in their respective NSFs.
5. The flow is protected with the IPsec SA established with IKEv2 between both NSFs.

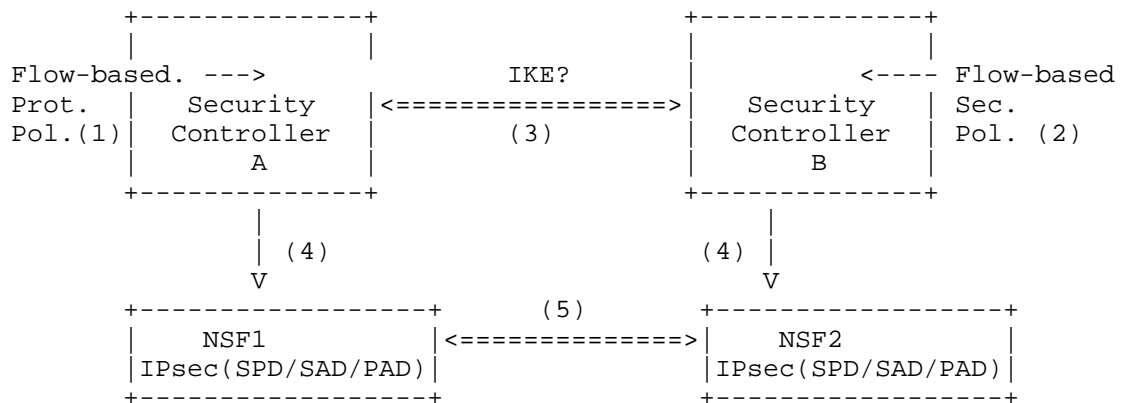


Figure 6: Different security controllers in case 2

Figure 5 describes case 1 when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow Protection Policies in Security Controller A.
 2. The B's administrator establishes general Flow Protection Policies in Security Controller B.
 3. The Security Controller A realizes that the flow between NSF1 and NSF2 MUST be protected. Nevertheless, the controller notices that NSF2 is under the control of another Security Controller, so it starts negotiations with the other controller to agree on the IPsec SPD and SAD entries that define the IPsec SAs. NOTE: It would worth evaluating IKE as the protocol for the the East/West interface in this case.
 4. Once the controllers have agreed on key material and the details of the IPsec SA, they both enforce this information into their respective NSFs.
 5. The flow is protected with the IPsec SA established by both Security Controllers in their respective NSFs.
8. Implementation notes

At the time of writing this document, we have implemented a proof-of-concept using NETCONF as southbound protocol, and the YANG model described in Appendix A. The netopeer implementation [netopeer] has been used for both case 1 and case 2 using host-to-host and gateway-

to-gateway configuration. For the case 1, we have used Strongswan [strongswan] distribution for the IKE implementation.

Note that the proposed YANG model provides the models for SPD, SAD, PAD and IKE, but, as describe before, only part of them are required depending of the case (1 or 2) been applied. The Controller should be able to know the kind of case to be applied in the NSF and to select the corresponding models based on the YANG features defines for each one

Internally to the NSF, the NETCONF server (that implements the I2NSF Agent) is able to apply the required configuration updating the corresponding NETCONF datastores (running, startup, etc.). Besides, it can deal with the SPD and SAD configuration at kernel level, through different APIs. For example, the IETF RFC 2367 (PF_KEYv2) [RFC2367] provides a generic key management API that can be used not only for IPsec but also for other network security services to manage the IPsec SAD. Besides, as an extension to this API, the document [I-D.pfkey-spd] specifies some PF_KEY extensions to maintain the SPD. This API is accessed using sockets.

An alternative key management API based on Netlink socket API [RFC3549] is used to configure IPsec on the Linux Operating System.

To allow the NETCONF server implementation interacts with the IKE daemon, we have used the Versatile IKE Configuration Interface (VICI) in Strongswan. This allows changes in the IKE part of the configuration data to be applied in the IKE daemon dynamically.

9. Security Considerations

TBD.

10. Acknowledgements

Authors want to thank Sowmini Varadhan, David Carrel, Yoav Nir, Tero Kivinen, Paul Wouters, Graham Bartlett, Sandeep Kampati, Linda Dunbar, Carlos J. Bernardos, Alejandro Perez-Mendez, Alejandro Abad-Carrascosa, Ignacio Martinez and Ruben Ricart for their valuable comments.

11. References

11.1. 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>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.

11.2. 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-04 (work in progress), October 2016.
- [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.
- [I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.
- [I-D.jeong-i2nsf-sdn-security-services-05]
Jeong, J., Kim, H., Park, J., Ahn, T., and S. Lee, "Software-Defined Networking Based Security Services using Interface to Network Security Functions", draft-jeong-i2nsf-sdn-security-services-05 (work in progress), July 2016.

- [I-D.pfkey-spd]
Sakane, S., "PF_KEY Extensions for IPsec Policy Management in KAME Stack", October 2002.
- [I-D.sivakumar-yang-nat]
Sivakumar, S., Boucadair, M., and S. <>, "YANG Data Model for Network Address Translation (NAT)", draft-sivakumar-yang-nat-06 (work in progress), March 2017.
- [I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-01 (work in progress), June 2015.
- [ITU-T.X.1252]
"Baseline Identity Management Terms and Definitions", April 2010.
- [ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.
- [ITU-T.Y.3300]
"Recommendation ITU-T Y.3300", June 2014.
- [netconf-vpn]
Stefan Wallin, "Tutorial: NETCONF and YANG", January 2014.
- [netopeer]
CESNET, CESNET., "NETCONF toolset Netopeer", November 2016.
- [ONF-OpenFlow]
ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.
- [ONF-SDN-Architecture]
"SDN Architecture", June 2014.
- [RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, DOI 10.17487/RFC2367, July 1998, <<http://www.rfc-editor.org/info/rfc2367>>.
- [RFC3549] Salim, J., Khosravi, H., Kleen, A., and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", RFC 3549, DOI 10.17487/RFC3549, July 2003, <<http://www.rfc-editor.org/info/rfc3549>>.

- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <<http://www.rfc-editor.org/info/rfc3948>>.
- [RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <<http://www.rfc-editor.org/info/rfc6071>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<http://www.rfc-editor.org/info/rfc7149>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<http://www.rfc-editor.org/info/rfc7317>>.
- [strongswan] CESNET, CESNET., "StrongSwan: the OpenSource IPsec-based VPN Solution", April 2017.

Appendix A. Appendix A: YANG model IPsec Configuration data

```
<CODE BEGINS> file "ietf-ipsec.yang"
module ietf-ipsec {

    namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec";

    prefix "eipsec";

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

    organization "University of Murcia";

    contact
    " Rafael Marin Lopez
    Dept. Information and Communications Engineering (DIIC)
    Faculty of Computer Science-University of Murcia
    30100 Murcia - Spain
    Telf: +34868888501
    e-mail: rafa@um.es

    Gabriel Lopez Millan
    Dept. Information and Communications Engineering (DIIC)
    Faculty of Computer Science-University of Murcia
    30100 Murcia - Spain
    Tel: +34 868888504
    email: gabilm@um.es
    ";

    description "Data model for IPSec";

    revision "2017-05-02" {
        description
        "Initial revision.";
        reference "";
    }

    feature casel { description "feature case 1: IKE SPD PAD"; } // IKE/IPSec in
the NSF's
    feature case2 { description "feature case 2: SPD SAD"; } // Only IPSec in th
e NSF's
```

```

typedef encryption-algorithm-t {
    type enumeration {
enum reserved-0 {description "reserved";}
enum des-iv4 { description "DES IV 4";}
enum des { description "DES"; }
enum 3des { description "3DES"; }
enum rc5 { description "RC5"; }
enum idea { description "IDEA"; }
enum cast { description "CAST"; }
enum blowfish { description "BlowFish"; }
enum 3idea { description "3IDEA"; }
enum des-iv32 { description "DES-IV32"; }
enum reserved-10 { description "reserved-10"; }
enum null { description "NULL"; }
enum aes-cbc { description "AES-CBC"; }
enum aes-ctr { description "AES-CTR"; }
enum aes-ccm-8 { description "AES-CCM-8"; }
enum aes-ccm-12 { description "AES-CCM-12"; }
enum aes-ccm-16 { description "AES-CCM-16"; }
enum reserved-17 { description "reserved-17"; }
enum aes-gcm-8-icv { description "AES-GCM-8-ICV"; }
enum aes-gcm-12-icv { description "AES-GCM-12-ICV"; }
enum aes-gcm-16-icv { description "AES-GCM-16-ICV"; }
enum null-auth-aes-gmac { description "Null-Auth-AES-GMAC"; }
enum ieee-pl1619-xts-aes {
    description
    "encr-ieee-pl1619-xts-aes --> Reserved for IEEE P1619 XTS
-AES.";
    }
enum camellia-cbc { description "CAMELLIA-CBC"; }
enum camellia-ctr { description "CAMELLIA-CTR"; }
enum camellia-ccm-8-icv { description "CAMELLIA-CCM-8-ICV"; }
enum camellia-ccm-12-icv { description "CAMELLIA-CCM-12-ICV"; }
enum camellia-ccm-16-icv { description "CAMELLIA-CCM-16-ICV"; }
enum aes-cbc-128 { description "AES-CBC-128"; }
enum aes-cbc-192 { description "AES-CBC-192"; }
enum aes-cbc-256 { description "AES-CBC-256"; }
enum blowfish-128 { description "BlowFish-128"; }
enum blowfish-192 { description "BlowFish-192"; }
enum blowfish-256 { description "BlowFish-256"; }
enum blowfish-448 { description "BlowFish-448"; }
enum camellia-128 { description "CAMELLIA-128"; }
enum camellia-192 { description "CAMELLIA-192"; }
enum camellia-256 { description "CAMELLIA-256"; }
    }
description "Encryption algorithms --> RFC_5996";
}

```

```
typedef integrity-algorithm-t {  
    type enumeration {  
        enum none { description "NONE"; }  
        enum hmac-md5-96 { description "HMAC-MD5-96"; }  
        enum hmac-sha1-96 { description "HMAC-SHA1-96"; }  
        enum des-mac { description "DES-MAC"; }  
        enum kpdck-md5 { description "KPDCK-MD5"; }  
        enum aes-xcbc-96 { description "AES-XCBC-96"; }  
        enum hmac-md5-128 { description "HMAC-MD5-128"; }  
        enum hmac-sha1-160 { description "HMAC-SHA1-160"; }  
        enum aes-cmac-96 { description "AES-CMAC-96"; }  
        enum aes-128-gmac { description "AES-128-GMAC"; }  
        enum aes-192-gmac { description "AES-192-GMAC"; }  
        enum aes-256-gmac { description "AES-256-GMAC"; }  
        enum hmac-sha2-256-128 { description "HMAC-SHA2-256-128"; }  
        enum hmac-sha2-384-192 { description "HMAC-SHA2-384-192"; }  
        enum hmac-sha2-512-256 { description "HMAC-SHA2-512-256"; }  
        enum hmac-sha2-256-96 { description "HMAC-SHA2-256-096"; }  
    }  
    description "Integrity Algorithms --> RFC_5996";  
}  
  
typedef combined-algorithm-t {  
    type enumeration {  
        enum AES-GCM-16-ICV { description "AES-GCM-16-ICV"; }  
        enum AES-CCM { description "AES-CCM"; }  
    }  
    description "Combined Algorithms --> RFC 7321";  
}  
  
typedef auth-protocol-type {  
    type enumeration {  
        enum IKEv1 { // not supported by model  
            description "Authentication protocol based on IKEv1";  
        }  
        enum IKEv2 {  
            description "Authentication protocol based on IKEv2";  
        }  
        enum KINK { // not supported by model  
            description "Authentication protocol based on KINK";  
        }  
    }  
    description "Peer authentication protocols";  
}  
  
typedef ipsec-mode {
```

```
    type enumeration {
        enum TRANSPORT { description "Transport mode"; }
        enum TUNNEL { description "Tunnel mode"; }
        enum BEET { description "Bound End-to-End Tunnel (BEET) mode for ESP."; } /*Supported by XFRM*/
        enum RO { description "Route Optimization mode for Mobile IPv6"; } /*Supported by XFRM*/
        enum IN_TRIGGER {description "In trigger mode for Mobile IPv6"; } /*Supported by XFRM*/
    }
    description "type define of ipsec mode";
}

typedef ipsec-protocol {

    type enumeration {
        enum ah { description "AH Protocol"; }
        enum esp { description "ESP Protocol"; }
        enum comp { description "IP Compression"; } /*Supported by XFRM*/
        enum route2 { description "Routing Header type 2. Mobile IPv6"; } /*Supported by XFRM*/
        enum hao {description "Home Agent Option"; } /*Supported by XFRM*/
    }
    description "type define of ipsec security protocol";
}

typedef ipsec-spi {

    type uint32 { range "1..max"; }
    description "SPI";
}

typedef lifetime-action {
    type enumeration {
        enum terminate {description "Terminate the IPsec SA";}
        enum replace {description "Replace the IPsec SA with a new one";}
    }
    description "Action when lifetime expiration";
}

typedef ipsec-traffic-direction {

    type enumeration {
        enum INBOUND { description "Inbound traffic"; }
        enum OUTBOUND { description "Outbound traffic"; }
        enum FORWARD { description "Forwarded traffic"; }
    }
    description "IPsec traffic direction";
}
```

```
typedef ipsec-spd-operation {  
    type enumeration {  
        enum PROTECT { description "PROTECT the traffic with IPsec"; }  
        enum BYPASS { description "BYPASS the traffic"; }  
        enum DISCARD { description "DISCARD the traffic"; }  
    }  
    description "The operation when traffic matches IPsec security policy";  
}
```

```
typedef ipsec-next-layer-proto {  
    type enumeration {  
        enum TCP { description "PROTECT the traffic with IPsec"; }  
        enum UDP { description "BYPASS the traffic"; }  
        enum SCTP { description "PROTECT the traffic with IPsec"; }  
        enum DCCP { description "PROTECT the traffic with IPsec"; }  
        enum ICMP { description "PROTECT the traffic with IPsec"; }  
        enum IPv6-ICMP { description "PROTECT the traffic with IPsec"; }  
        enum MH { description "PROTECT the traffic with IPsec"; }  
        enum GRE { description "PROTECT the traffic with IPsec"; }  
    }  
    description "Next layer proto on top of IP";  
}
```

```
typedef ipsec-spd-name {  
    type enumeration {  
        enum id_rfc_822_addr {  
            description "Fully qualified user name string.";  
        }  
        enum id_fqdn {  
            description "Fully qualified DNS name.";  
        }  
        enum id_der_asn1_dn {  
            description "X.500 distinguished name.";  
        }  
        enum id_key {  
            description "IKEv2 Key ID.";  
        }  
    }  
    description "IPsec SPD name type";  
}
```

```

typedef auth-method-type {
    /* Most implementations also provide XAUTH protocol, others used are: BL
    ISS, P12, NTLM, PIN */

    type enumeration {
        enum pre-shared {
            description "Select pre-shared key message as the authentication method";
        }
        enum rsa-signature {
            description "Select rsa digital signature as the authentication method";
        }
        enum dss-signature {
            description "Select dss digital signature as the authentication method";
        }
        enum eap {
            description "Select EAP as the authentication method";
        }
    }
    description "Peer authentication method";
}

/*##### PAD grouping #####*/

grouping auth-method-grouping {
    description "Peer authentication method data";

    container auth-method {
        description "Peer authentication method container";

        leaf auth-m {
            type auth-method-type;
            description "Type of authentication method (preshared, rsa, etc.)";
        }

        container pre-shared {
            when "../auth-m = 'pre-shared'";
            leaf secret { type string; description "Pre-shared secret value"; }
            description "Shared secret value";
        }

        container rsa-signature {
            when "../auth-m = 'rsa-signature'";
            leaf key-data {
                type string;
                description "RSA private key data - PEM";
            }

            leaf key-file {
                type string;
            }
        }
    }
}

```

```

        description "RSA private key file name ";
    }

    leaf-list ca-data {
        type string;
        description "List of trusted CA certs - PEM";
    }
    leaf ca-file {
        type string;
        description "List of trusted CA certs file";
    }
    leaf cert-data {
        type string;
        description "X.509 certificate data - PEM4";
    }
    leaf cert-file {
        type string;
        description "X.509 certificate file";
    }
    leaf crl-data {
        type string;
        description "X.509 CRL certificate data in base64";
    }
    leaf crl-file {
        type string;
        description " X.509 CRL certificate file";
    }
    }
    description "RSA signature container";
}
}
}

grouping identity-grouping {
    description "Identification type. It is an union identity";
    choice identity {
        description "Choice of identity.";

        leaf ipv4-address {
            type inet:ipv4-address;
            description "Specifies the identity as a single four (4) octet IPv4 address. An example is, 10.10.10.10. ";
        }
        leaf ipv6-address {
            type inet:ipv6-address;
            description "Specifies the identity as a single sixteen (16) octet IPv6 address. An example is FF01::101, 2001:DB8:0:0:8:800:200C:417A .";
        }
        leaf fqdn-string {
            type inet:domain-name;
            description "Specifies the identity as a Fully-Qualified Domain Name (FQDN) string. An example is: example.com. The string MUST not contain any terminators (e.g., NULL, CR, etc.).";
        }
    }
}

```

```

    }
    leaf rfc822-address-string {
        type string;
        description "Specifies the identity as a fully-qualified RFC822 email address string. An example is, jsmith@example.com. The string MUST not contain any terminators (e.g., NULL, CR, etc.).";
    }
    leaf dnX509 {
        type string;
        description "Specifies the identity as a distinguished name in the X.509 tradition.";
    }
    leaf id_key {
        type string;
        description "Key id";
    } /* From RFC4301 list of id types */
}
} /* grouping identity-grouping */

/*##### end PAD grouping #####*/

/*##### SAD and SPD grouping #####*/

grouping ip-addr-range {
    description "IP address range grouping";
    leaf start {
        type inet:ip-address;
        description "Start IP address";
    }
    leaf end {
        type inet:ip-address;
        description "End IP address";
    }
}

grouping port-range {
    description "Port range grouping";
    leaf start {
        type inet:port-number;
        description "Start IP address";
    }
    leaf end {
        type inet:port-number;
        description "End IP address";
    }
}

grouping tunnel-grouping {
    description "Tunnel mode grouping";
    leaf local{ type inet:ip-address; description "Local tunnel endpoint"; }
    leaf remote{ type inet:ip-address; description "Remote tunnel endpoint"; }
}

```



```

    leaf bypass-df { type boolean; description "bypass DF bit"; }
    leaf bypass-dscp { type boolean; description "bypass DSCP"; }
    leaf dscp-mapping { type yang:hex-string; description "DSCP mapping"; }
    leaf ecn { type boolean; description "Bit ECN"; } /* RFC 4301 ASN1 notation.
Annex C*/
}

grouping selector-grouping {
  description "Traffic selector grouping";
  list local-addresses {
    key "start end";
    uses ip-addr-range;
    description "List of local addresses";
  }
  list remote-addresses {
    key "start end";
    uses ip-addr-range;
    description "List of remote addresses";
  }
  leaf-list next-layer-protocol { type ipsec-next-layer-proto; description "Li
st of Next Layer Protocol"; }
  list local-ports {
    key "start end";
    uses port-range;
    description "List of local ports";
  }

  list remote-ports {
    key "start end";
    uses port-range;
    description "List of remote ports";
  }
}

/*##### SAD grouping #####*/

grouping ipsec-sa-grouping {
  description "Configure Security Association (SA). Section 4.4.2.1 in RFC 430
1";

  leaf spi { type ipsec-spi; description "Security Parameter Index"; }
  leaf seq-number { type uint64; description "Current sequence number of IPsec
packet."; }
  leaf seq-number-overflow-flag { type boolean; description "The flag indicati
ng whether overflow of the sequence number counter should prevent transmission o
f additional packets on the SA, or whether rollover is permitted."; }
  leaf anti-replay-window { type uint16 { range "0 | 32..1024"; } description
"Anti replay window"; }
  leaf rule-number { type uint32; description "This value links the SA with the
SPD entry"; }

  uses selector-grouping;

  leaf security-protocol { type ipsec-protocol; description "Security protocol
of IPsec SA: Either AH or ESP."; }

  container ah-sa {

```

```
    when "../security-protocol = 'ah'";
    description "Configure Authentication Header (AH) for SA";
    leaf integrity-algorithm { type integrity-algorithm-t; description "Configure Authentication Header (AH)."; }
    leaf key { type string; description "AH key value"; }
  }

  container esp-sa {
    when "../security-protocol = 'esp'";
    description "Set IPsec Encapsulation Security Payload (ESP)";

    container encryption {
      description "Configure encryption for IPsec Encapsulation Security Payload (ESP)";
      leaf encryption-algorithm { type encryption-algorithm-t; description "Configure ESP encryption"; }
      leaf key { type string; description "ESP encryption key value"; }
      leaf iv { type string; description "ESP encryption IV value"; }
    }

    container integrity {
      description "Configure authentication for IPsec Encapsulation Security Payload (ESP)";
      leaf integrity-algorithm { type integrity-algorithm-t; description "Configure Authentication Header (AH)."; }
      leaf key { type string; description "ESP integrity key value"; }
    }

    container combined {
      description "ESP combined mode algorithms (encryption and integrity)";
      leaf combined-algorithm { type combined-algorithm-t; description "Combined algorithm AEAD"; }
    }
  }

  container sa-lifetime {
    description "This may be expressed as a time or byte count, or a simultaneous use of both with the first lifetime to expire taking precedence";
    leaf time-soft { type uint32; default 0; description "Soft time lifetime"; }
    leaf time-hard { type uint32; default 0; description "Hard time lifetime"; }
    leaf time-use-soft { type uint32; default 0; description "Use Soft time lifetime"; }
    leaf time-use-hard { type uint32; default 0; description "Use Hard time lifetime"; }
    leaf byte-soft { type uint32; default 0; description "Byte soft lifetime"; }
    leaf byte-hard { type uint32; default 0; description "Byte hard lifetime"; }
    leaf packet-soft { type uint32; default 0; description "Packet soft lifetime"; }
    leaf packet-hard { type uint32; default 0; description "Packet hard lifetime"; }
    leaf action { type lifetime-action; description "action lifetime"; }
  }

  leaf mode { type ipsec-mode; description "SA Mode"; }
  leaf statefulfragcheck { type boolean; description "TRUE stateful fragment checking, FALSE no stateful fragment checking"; }
  leaf dscp { type yang:hex-string; description "DSCP value"; }
```



```

    container tunnel {
        when "../mode = 'TUNNEL'";
        uses tunnel-grouping;
        description "Container for tunnel grouping";
    }

    leaf path-mtu { type uint16; description "Maximum size of an IPsec packet that
    can be transmitted without fragmentation"; }

    container encap { /* This is defined by XFRM */
        description "Encapsulation container";
        leaf espinudp {type boolean; description "TRUE espinudp; FALSE espinudp-no
    nike";}
        leaf sport {type inet:port-number; description "Encapsulation source port"
    ;}
        leaf dport {type inet:port-number; description "Encapsulation destination
    port"; }
        leaf oaddr {type inet:ip-address; description "Encapsulation Original Addr
    ess ";}
    }

}

/*##### end SAD grouping #####*/

/*##### SPD grouping #####*/

grouping ipsec-policy-grouping {
    description "Holds configuration information for an IPSec SPD entry.";

    leaf rule-number {
        type uint64;
        description "SPD index. RFC4301 does not mention an index however real imp
    lementations provide a policy index/or id to refer a policy. ";
    }
    leaf priority {type uint32; default 0; description "Policy priority";}
    list names {
        key "name";
        leaf name-type {
            type ipsec-spd-name;
            description "SPD name type.";
        }
        leaf name {
            type string; description "Policy name";
        }
        description "List of policy names";
    }

    container condition {
        description "SPD condition --> RFC4301";
    }
}

```

```

    list traffic-selector-list {
        key "ts-number";

        leaf ts-number { type uint32; description "Traffic selector number"; }
        leaf direction { type ipsec-traffic-direction; description "in/fwd/out"; }
    }

    uses selector-grouping;
    leaf selector-priority { type uint32; default 0; description "It establishes a priority to the traffic selector"; }
    ordered-by user;

    description "List of traffic selectors";
}

container processing-info {
    description "SPD processing --> RFC4301";
    leaf action { type ipsec-spd-operation; mandatory true; description "If the action is bypass or discard processing container ipsec-sa-cfg is empty"; }

    container ipsec-sa-cfg {
        when "../action = 'PROTECT'";

        leaf pfp-flag { type boolean; description "Each selector has with a pfp flag."; }
        leaf extSeqNum { type boolean; description "TRUE 64 bit counter, FALSE 32 bit"; }
        leaf seqOverflow { type boolean; description "TRUE rekey, FALSE terminate & audit"; }
        leaf statefulfragCheck { type boolean; description "TRUE stateful fragment checking, FALSE no stateful fragment checking"; }
        leaf security-protocol { type ipsec-protocol; description "Security protocol of IPsec SA: Either AH or ESP."; }
        leaf mode { type ipsec-mode; description "transport/tunnel"; }

        container ah-algorithms {
            when "../security-protocol = 'ah'";
            leaf-list ah-algorithm {
                type integrity-algorithm-t;
                description "Configure Authentication Header (AH).";
            }
            description "AH algorithms ";
        }

        container esp-algorithms {
            when "../security-protocol = 'esp'";
            description "Configure Encapsulating Security Payload (ESP).";
            leaf-list authentication { type integrity-algorithm-t; description "Configure ESP authentication"; }
            leaf-list encryption { type encryption-algorithm-t; description "Configure ESP encryption"; }
        }

        container tunnel {
            when "../mode = 'TUNNEL'";
            uses tunnel-grouping;
        }
    }
}

```

```

        description "tunnel grouping container";
    }
    description " IPsec SA configuration container";
}

container spd-lifetime {
    description "SPD lifetime parameters";
    leaf time-soft { type uint32; default 0; description "Soft time lifetime";
}
    leaf time-hard { type uint32; default 0; description "Hard time lifetime";
}
    leaf time-use-soft { type uint32; default 0; description "Use soft lifetime";
}
    leaf time-use-hard { type uint32; default 0; description "Use hard lifetime";
}
    leaf byte-soft { type uint32; default 0; description "Byte soft lifetime";
}
    leaf byte-hard { type uint32; default 0; description "Hard soft lifetime";
}
    leaf packet-soft { type uint32; default 0; description "Packet soft lifetime";
}
    leaf packet-hard { type uint32; default 0; description "Packet hard lifetime";
}
} /* grouping ipsec-policy-grouping */

/*##### end SPD grouping #####*/

/*##### IKEv2-grouping #####*/

    grouping isakmp-proposal {
description "ISAKMP proposal grouping";
        leaf phasel-lifetime {
            type uint32;
            mandatory true;
            description "lifetime for IKE Phase 1 SAs";
        }
        leaf phasel-authby {
            type auth-method-type;
            mandatory true;
            description "Auth method for IKE Phase 1 SAs";
        }
        leaf-list phasel-authalg {
            type integrity-algorithm-t;
            description "Auth algorithm for IKE Phase 1 SAs";
        }
        leaf-list phasel-encalg {
            type encryption-algorithm-t;
            description "Auth algorithm for IKE Phase 1 SAs";
        }
        leaf dh_group {
            type uint32;

```

```

        mandatory true;
        description "Group number for Diffie Hellman Exponentiat
ion";
    }
} /* list isakmp-proposal */

grouping phase2-info {
description "IKE Phase 2 Information";
    leaf local-addr {
        type inet:ip-address;
        mandatory true;
        description "IKEv2 Local address";
    }
    leaf remote-addr {
        type inet:ip-address;
        mandatory true;
        description "IKEv2 Remote address";
    }
    leaf pfs_group {
        type uint32;
        description
            "If non-zero, require perfect forward secrecy
            when requesting new SA. The non-zero value is
            the required group number";
    }
    leaf phase2-lifetime {
        type uint32;
        mandatory true;
        description "lifetime for IKE Phase 2 SAs";
    }
    leaf-list phase2-authalg {
        type integrity-algorithm-t;
        description "Auth algorigthm for IKE Phase 2 SAs";
    }
    leaf-list phase2-encalg {
        type encryption-algorithm-t;
        description "Auth algorithm for IKE Phase 2 SAs";
    }
}

grouping local-grouping {
    description "Configure the local peer in an IKE connection";

    container local {
        description "Local container";
        choice my-identifier-type {
            default ipv4;
            case ipv4 {

```

```

        leaf ipv4 {
            type inet:ipv4-address;
            description "IPv4 dotted-decimal address
";
        }
    }
    case ipv6 {
        leaf ipv6 {
            type inet:ipv6-address;
            description "numerical IPv6 address";
        }
    }
    case fqdn {
        leaf fqdn {
            type inet:domain-name;
            description "Fully Qualified Domain name
";
        }
    }
    case dn {
        leaf dn {
            type string;
            description "Domain name";
        }
    }
    case user_fqdn {
        leaf user_fqdn {
            type string;
            description "User FQDN";
        }
    }
    }
    description "Local ID type";
}
leaf my-identifier {
    type string;
    mandatory true;
    description "Local id used for authentication";
}
}

grouping remote-grouping {
    description "Configure the remote peer in an IKE connection";
    container remote {
        description "Remote container";
        choice my-identifier-type {
            default ipv4;
            case ipv4 {
                leaf ipv4 {
                    type inet:ipv4-address;

```



```

                                description "IPv4 dotted-decimal address
";
                                }
                                }
                                case ipv6 {
                                    leaf ipv6 {
                                        type inet:ipv6-address;
                                        description "numerical IPv6 address";
                                    }
                                }
                                case fqdn {
                                    leaf fqdn {
                                        type inet:domain-name;
                                        description "Fully Qualified Domain name
";
                                    }
                                }
                                case dn {
                                    leaf dn {
                                        type string;
                                        description "Domain name";
                                    }
                                }
                                case user_fqdn {
                                    leaf user_fqdn {
                                        type string;
                                        description "User FQDN";
                                    }
                                }
                                }
                                description "Local ID type";
                            }
                            leaf my-identifier {
                                type string;
                                mandatory true;
                                description "Local id used for authentication";
                            }
                        }
                    }
}

/*##### End IKEv2-groupingUMU #####*/

/*##### Register grouping #####*/

typedef sadb-msg-type {

    type enumeration {
        enum sadb_reserved { description "SADB_RESERVED";}
        enum sadb_getspi { description "SADB_GETSPI";}
        enum sadb_update { description "SADB_UPDATE";}
        enum sadb_add { description "SADB_ADD";}
    }
}

```

```

    enum sadb_delete { description "SADB_DELETE"; }
    enum sadb_get { description "SADB_GET"; }
    enum sadb_acquire { description "SADB_ACQUIRE"; }
    enum sadb_register { description "SADB_REGISTER"; }
    enum sadb_expire { description "SADB_EXPIRE"; }
    enum sadb_flush { description "SADB_FLUSH"; }
    enum sadb_dump { description "SADB_DUMP"; }
    enum sadb_x_promisc { description "SADB_X_PROMISC"; }
    enum sadb_x_pchange { description "SADB_X_PCHANGE"; }
    enum sadb_max { description "SADB_MAX"; }
}
description "PF_KEY base message types";
}

typedef sadb-msg-satype {

    type enumeration {
        enum sadb_satype_unspec { description "SADB_SATYPE_UNSPEC"; }
        enum sadb_satype_ah { description "SADB_SATYPE_AH"; }
        enum sadb_satype_esp { description "SADB_SATYPE_ESP"; }
        enum sadb_satype_rsvp { description "SADB_SATYPE_RSVP"; }
        enum sadb_satype_ospfv2 { description "SADB_SATYPE OSPFv2"; }
        enum sadb_satype_ripv2 { description "SADB_SATYPE_RIPv2"; }
        enum sadb_satype_mip { description "SADB_SATYPE_MIP"; }
        enum sadb_satype_max { description "SADB_SATYPE_MAX"; }
    }
    description "PF_KEY Security Association types";
}

grouping base-grouping {
    description "Configuration for the message header format";
    list base-list {
        key "version";
        leaf version { type string; description "Version of PF_KEY (MUST be PF_KEY
_V2)"; }
        leaf msg_type { type sadb-msg-type; description "Identifies the type of me
ssage"; }
        leaf msg_satype { type sadb-msg-satype; description "Defines the type of S
ecurity Association"; }
        leaf msg_seq { type uint32; description "Sequence number of this message."
; }
    }
    description "Configuration for a specific message header format";
}

grouping algorithm-grouping {
    description "List of supported authentication and encryption algorithms";
    list algorithm-supported {
        container authentication {
            description "Authentication algorithm supported";
            leaf name { type integrity-algorithm-t; description "Name of authenticat
ion algorithm"; }
            leaf ivlen { type uint8; description "Length of the initialization vecto
r to be used for the algorithm"; }
        }
    }
}

```

```

        leaf min-bits { type uint16; description "The minimun acceptable key len
gth, in bits"; }
        leaf max-bits { type uint16; description "The maximun acceptable key len
gth, in bits"; }
    }
    container encryption {
        description "Encryption algorithm supported";
        leaf name { type encryption-algorithm-t; description "Name of encryption
algorithm"; }
        leaf ivlen { type uint8; description "Length of the initialization vecto
r to be used for the algorithm"; }
        leaf min-bits { type uint16; description "The minimun acceptable key len
gth, in bits"; }
        leaf max-bits { type uint16; description "The maximun acceptable key len
gth, in bits"; }
    }
    description "List for a specific algorithm";
}
}

```

```
/*##### End Register grouping #####*/
```

```
/*##### ipsec #####*/
```

```

    container ietf-ipsec {
        description "Main IPsec container ";

        container ikev2 {
            if-feature casel;
            description "Configure the IKEv2";

            container ike-connection {
                description "IKE connections configuration";

                list ike-conn-entries {
                    key "conn-name";
                    description "IKE peer connetion information";
                    leaf conn-name {
                        type string;
                        mandatory true;
                        description "Name of IKE connect
ion";
                    }
                }

                leaf autostartup {
                    type boolean;
                    mandatory true;
                    description "if True: automatically start tunnel
at startup; else we do lazy tunnel setup based on trigger from datapath";
                }
                leaf nat-traversal {
                    type boolean;
                    default false;
                    description "Enable/Disable NAT traversal";
                }
                leaf version {

```

```
        type enumeration {
            /* we only support ikev2 in this version */
            enum ikev2 {value 2; description "IKE version 2";}
        }
        description "IKE version";
    }

    uses isakmp-proposal;
    uses local-grouping;
    uses remote-grouping;
    uses phase2-info;

    } /* ike-conn-entries */
    } /* container ike-connection */
} /* container ikev2 */

container ipsec {
    description "Configuration IPsec";

    container spd {
        description "Configure the Security Policy Database (SPD)";
        list spd-entry {
            key "rule-number";
            uses ipsec-policy-grouping;
            ordered-by user;
            description "List of SPD entries";
        }
    }

    container sad {
        if-feature case2;
        description "Configure the IPSec Security Association Database (SAD)";
        list sad-entry {
            key "spi";
            uses ipsec-sa-grouping;
            description "List of SAD entries";
        }
    }

    container pad {
        if-feature case1;
        description "Configure Peer Authorization Database (PAD)";

        list pad-entries {
            key "pad-entry-id";
            ordered-by user;
            description "Peer Authorization Database (PAD)";
        }
    }
}
```

```

        leaf pad-entry-id {
            type uint64;
            description "SAD index. ";
        }

        uses identity-grouping;

        leaf pad-auth-protocol {
            type auth-protocol-type;
            description "IKEv1, IKEv2, KINK, etc. ";
        }
        uses auth-method-grouping;
    }
}

} /* container ietf-ipsec */

/*##### State Data #####*/

// TBD

/*##### RPC and Notifications #####*/

/* Note: not yet completed */
// Those RPCs are needed by a Security Controller in case 2 */

rpc sadb_register {
    description "Allows netconf to register its key socket as able to acquire new security associations for the kernel";
    input {
        uses base-grouping;
    }
    output {
        uses base-grouping;
        uses algorithm-grouping;
    }
}

notification spd-expire {
    description "A SPD entry has expired";
    leaf index {
        type uint64;
        description "SPD index. RFC4301 does not mention an index however real implementations (e.g. XFRM or PFKEY_v2 with KAME extensions provide a policy index to refer a policy. ";
    }
}

```

```
notification sadb_acquire {
  description "A IPsec SA is required ";
  leaf state {
    type uint32;
    mandatory "true";
    description
      "Request the creation of a SADB entry";
  }
}

notification sadb_expire {
  description ".....";
  leaf state {
    type uint32;
    mandatory "true";
    description
      "Notify the expiration of a entry in the SADB";
  }
}

} /*module ietf-ipsec*/
```

<CODE ENDS>

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
EMail: rafa@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 04
EMail: gabilm@um.es

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

S. Hares
Huawei
J. Jeong
J. Kim
Sungkyunkwan University
R. Moskowitz
HTT Consulting
L. Xia
Huawei
July 3, 2017

I2NSF Capability YANG Data Model
draft-hares-i2nsf-capability-data-model-03

Abstract

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

Status of This Memo

This Internet-Draft is submitted 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 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
3.1. Tree Diagrams	3
4. High-Level YANG	4
4.1. Capabilities per NSF	4
4.2. Network Security Control	4
4.3. Content Security Control	5
4.4. Attack Mitigation Control	7
4.5. Information on Capabilities	9
4.6. Location for Capabilities	10
4.7. IT Resources linked to Capabilities	10
4.8. Actions	11
5. YANG Modules	11
6. IANA Considerations	33
7. Security Considerations	34
8. Acknowledgments	34
9. References	34
9.1. Normative References	34
9.2. Informative References	34
Appendix A. Changes from draft-hares-i2nsf-capability-data-model-01	36
Authors' Addresses	36

1. Introduction

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

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

This document provides a YANG model that defines the capabilities for security devices that can be utilized by I2NSF NSF-facing interface between the I2NSF security controller and the NSF devices to express the capabilities of NSF devices. This YANG model can also be used by the I2NSF user (or I2NSF client) to provide a complete list of the I2NSF capabilities that can be controlled by the security controller. This document defines a YANG [RFC6020] data model based on the

[i2nsf-nsf-cap-im]. Terms used in document are defined in [i2nsf-terminology]. [i2nsf-nsf-cap-im] defines the following type of functionality in NSFs.

- o Network Security Control
- o Content Security Control
- o 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-nsf-cap-im] [i2rs-rib-data-model][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

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

3.1. Tree Diagrams

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

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".

- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. 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* [nsf-capabilities-id]
    +--rw nsf-capabilities-id          uint 8
    +--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
```

Figure 1: High-Level YANG of I2NSF Capability Interface

Each of these section mirror sections in: [i2nsf-nsf-cap-im]. The high-level YANG for net-sec-control-capabilities, con-sec-control-capabilities, and attack-mitigation-capabilities. This draft also utilizes the concepts originated in Basile, Liroy, Pitscheider, and Zhao[2015] concerning conflict resolution, 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
    |   uses capabilities-information

```

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
    |   |   uses capabilities-information
+--rw ips
    |   +--rw ips-support? boolean
    |   +--rw ips-fcn* [ips-fcn-name]
    |   |   +--rw ips-fcn-name string //std or vendor name
    |   |   uses capabilities-information
+--rw ids
    |   +--rw ids-support? boolean
    |   +--rw ids-fcn* [ids-fcn-name]
    |   |   +--rw ids-fcn-name string //std or vendor name
    |   |   uses capabilities-information
+--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
    |   |   uses capabilities-information
+--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
|           uses capabilities-information
+--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
|           uses capabilities-information
+--rw dns-filter
|   +--rw dns-filter-support?  boolean
|   +--rw dns-filter-fcn*    [dns-filter-name]
|       +--rw dns-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw ftp-filter
|   +--rw ftp-filter-support?  boolean
|   +--rw ftp-filter-fcn*    [ftp-filter-fcn-name]
|       +--rw ftp-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw games-filter
|   +--rw games-filter-support?  boolean
|   +--rw games-filter-fcn*    [games-filter-fcn-name]
|       +--rw games-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw p2p-filter
|   +--rw p2p-filter-support?  boolean
|   +--rw p2p-filter-fcn*    [p2p-filter-fcn-name]
|       +--rw p2p-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw rpc-filter
|   +--rw rpc-filter-support?  boolean
|   +--rw rpc-filter-fcn*    [rpc-filter-fcn-name]
|       +--rw rpc-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw sql-filter
|   +--rw sql-filter-support?  boolean
|   +--rw sql-filter-fcn*    [sql-filter-fcn-name]
|       +--rw sql-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw telnet-filter
|   +--rw telnet-filter-support?  boolean
|   +--rw telnet-filter-fcn*    [telnet-filter-fcn-name]
|       +--rw telnet-filter-fcn-name  string  //std or vendor name
|           uses capabilities-information
+--rw tftp-filter
|   +--rw tftp-filter-support?  boolean
|   +--rw tftp-filter-fcn*    [tftp-filter-fcn-name]
|       +--rw tftp-filter-fcn-name  string  //std or vendor name
```

```

|       uses capabilities-information
+--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
|       uses capabilities-information
+--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
|       uses capabilities-information
+--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
|       uses capabilities-information
+--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
|       uses capabilities-information

```

Figure 3: High-Level YANG of Content Security Control

4.4. Attack Mitigation Control

The 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
|                   uses capabilities-information
|           +--rw udp-flood-attack

```

```

|         |--rw udp-flood-attack-support?  boolean
|         |--rw udp-flood-fcn*   [udp-flood-fcn-name]
|         |         |--rw udp-flood-fcn-name  string
|         |         |         uses capabilities-information
|--rw icmp-flood-attack
|         |--rw icmp-flood-attack-support?  boolean
|         |--rw icmp-flood-fcn*   [icmp-flood-fcn-name]
|         |         |--rw icmp-flood-fcn-name  string
|         |         |         uses capabilities-information
|--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
|         |         |         uses capabilities-information
|--rw ipv6-related-attack
|         |--rw ipv6-related-attack-support?  boolean
|         |--rw ipv6-related-fcn*   [ipv6-related-fcn-name]
|         |         |--rw ipv6-related-fcn-name  string
|         |         |         uses capabilities-information
+---: (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
|         |         |         uses capabilities-information
|--rw https-flood-attack
|         |--rw https-flood-attack-support?  boolean
|         |--rw https-flood-fcn*   [https-flood-fcn-name]
|         |         |--rw https-flood-fcn-name  string
|         |         |         uses capabilities-information
|--rw dns-flood-attack
|         |--rw dns-flood-attack-support?  boolean
|         |--rw dns-flood-fcn*   [dns-flood-fcn-name]
|         |         |--rw dns-flood-fcn-name  string
|         |         |         uses capabilities-information
|--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
|         |         |         uses capabilities-information
|--rw ssl-ddos-attack
|         |--rw ssl-ddos-attack-support?  boolean
|         |--rw ssl-ddos-fcn*   [ssl-ddos-fcn-name]
|         |         |--rw ssl-ddos-fcn-name  string
|         |         |         uses capabilities-information
+---: (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
|   |   |   |   uses capabilities-information
|   +---rw port-scanning-attack
|   |   +---rw port-scanning-attack-support?  boolean
|   |   +---rw port-scanning-fcn*  [port-scanning-fcn-name]
|   |   |   +---rw port-scanning-fcn-name  string
|   |   |   |   uses capabilities-information
+---: (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
|   |   |   |   uses capabilities-information
|   +---rw teardrop-attack
|   |   +---rw teardrop-attack-support?  boolean
|   |   +---rw tear-drop-fcn*  [tear-drop-fcn-name]
|   |   |   +---rw tear-drop-fcn-name  string
|   |   |   |   uses capabilities-information
+---: (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
|   |   |   |   uses capabilities-information
|   +---rw tracert-attack
|   |   +---rw tracert-attack-support?  boolean
|   |   +---rw tracert-fcn*  [tracert-fcn-name]
|   |   |   +---rw tracert-fcn-name  string
|   |   |   |   uses capabilities-information

```

Figure 4: High-Level YANG of Attack Mitigation Control

4.5. Information on Capabilities

This section provides information on capabilities. This section has information on capabilities location and IT resources. Additional input is needed.

Capabilities Information

```

+--rw capabilities-information
+--rw nsf-location
|   uses i2nsf-nsf-location
+--rw it-resources
    uses i2nsf-it-resources

```

Figure 5: High-Level YANG of Information on Capabilities

4.6. Location for Capabilities

This section provides location for capabilities. This section has location for capabilities. Additional input is needed.

```

+--rw nsf-location
|   uses i2nsf-nsf-location

```

NSF Location

```

+--rw i2nsf-nsf-location
+--rw nsf-address
+--rw (nsf-address-type)?
+--:(ipv4-address)
|   +--rw ipv4-address inet:ipv4-address
+--:(ipv6-address)
    +--rw ipv6-address inet:ipv6-address

```

Figure 6: High-Level YANG of Capabilities Location

4.7. 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 7: High-Level YANG of IT Resources

4.8. 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-nsf-cap-im].

<CODE BEGINS> file "ietf-i2nsf-capability@2017-07-03.yang"

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

  import ietf-inet-types{
    prefix inet;
  }

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

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

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

     WG Chair: Linda Dunbar
```

<mailto:Linda.duhbar@huawei.com>

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

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

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

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

revision "2017-07-03" {
  description "The second revision";
  reference
    "draft-xibassnez-i2nsf-capability-01
    draft-hares-i2nsf-capability-data-model-02";
}

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

grouping i2nsf-nsf-location {
  description
    "This provides a location for capabilities.";
  container nsf-address {
    description
      "This is location information for capabilities.";
    choice nsf-address-type {
      description
        "nsf address type: ipv4 and ipv4";
      case ipv4-address {
        description
          "ipv4 case";
        leaf ipv4-address {
          type inet:ipv4-address;
          mandatory true;
          description
            "nsf address type is ipv4";
        }
      }
      case ipv6-address {
        description
```

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

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

grouping capabilities-information {
    description
        "This includes information of capabilities.";
    uses i2nsf-nsf-location;
    uses i2nsf-it-resources;
}

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";
  }
  uses capabilities-information;
}
}
}

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";
    }
    uses capabilities-information;
}
```

```
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";
        }
        uses capabilities-information;
    }
}
```

```
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";
        }
        uses capabilities-information;
    }
}
```

```
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";
        }
        uses capabilities-information;
    }
}
```

```
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";
        }
        uses capabilities-information;
    }
}

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";
        }
        uses capabilities-information;
    }
}

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

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

```
        key "dns-filter-fcn-name";
        description
            "dns-filter-fcn";

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

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

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

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

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

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



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

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

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

    leaf p2p-filter-support {
      type boolean;
      mandatory true;
      description
        "p2p-filter-support";
    }
    list p2p-filter-fcn {
      key "p2p-filter-fcn-name";
      description
        "p2p-filter-fcn";

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

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

    leaf rpc-filter-support {
      type boolean;
      mandatory true;
```

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

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

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

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

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

container telnet-filter {
    description
        "telnet-filter";

    leaf telnet-filter-support {
```

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

        leaf telnet-filter-fcn-name {
            type string;
            mandatory true;
            description
                "telnet-filter-fcn-name";
        }
        uses capabilities-information;
    }
}

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

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

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

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";
        }
        uses capabilities-information;
    }
}

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";
        }
        uses capabilities-information;
    }
}

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";
        }
        uses capabilities-information;
    }
}

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";
        }
        uses capabilities-information;
    }
}
```

```
    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";
        }
        uses capabilities-information;
      }
    }
  }
}

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";
      }
      uses capabilities-information;
    }
  }
  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";
      }
      uses capabilities-information;
    }
  }
  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";
        }
        uses capabilities-information;
    }
}
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";
        }
        uses capabilities-information;
    }
}
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";
        }
        uses capabilities-information;
    }
}

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";
                }
            }
            uses capabilities-information;
        }
    }
    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";
      }
      uses capabilities-information;
    }
  }
  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";
      }
      uses capabilities-information;
    }
  }
  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";
        }
        uses capabilities-information;
    }
}

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";
        }
        uses capabilities-information;
    }
}

}

}

}

}

}

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";
      }
      uses capabilities-information;
    }
  }
}
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";
    }
    uses capabilities-information;
  }
}
}
```

```
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";
      }
      uses capabilities-information;
    }
  }
}
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";
    }
    uses capabilities-information;
  }
}
```

```
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";
      }
      uses capabilities-information;
    }
  }
}
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";
    }
    uses capabilities-information;
  }
}
```

```

    }
  }
}

list nsf-capabilities {
  key "nsf-capabilities-id";
  description
    "nsf-capabilities";
  leaf nsf-capabilities-id {
    type uint8;
    mandatory true;
    description
      "nsf-capabilities-id";
  }

  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";
  }
}

```

<CODE ENDS>

Figure 8: 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. Acknowledgments

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

This document has greatly benefited from inputs by Daeyoung Hyun, Dongjin Hong, 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-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 2017.
- [i2nsf-nsf-cap-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01 (work in progress), March 2017.
- [i2nsf-problem-statement]
Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-16 (work in progress), May 2017.

[i2nsf-terminology]

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

[i2rs-rib-data-model]

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

[supa-policy-info-model]

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

Appendix A. Changes from draft-hares-i2nsf-capability-data-model-01

The following changes are made from draft-hares-i2nsf-capability-data-model-01:

- o This draft is revised to support the acquisition of the information of NSFs such as an NSF's IP address and resources related to capabilities.
- o To support the capability information, location, and resources of an NSF, container component is replaced with grouping component.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

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

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

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

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

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

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

EMail: Frank.xialiang@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

S. Hyun
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
Huawei
July 3, 2017

Service Function Chaining-Enabled I2NSF Architecture
draft-hyun-i2nsf-nsf-triggered-steering-03

Abstract

This document describes an architecture of the I2NSF framework using security function chaining for security policy enforcement. Security function chaining enables composite inspection of network traffic by steering the traffic through multiple types of network security functions according to the information model for NSFs capabilities in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve security function chaining. 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/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Objective	3
3. Terminology	4
4. Architecture	5
4.1. SFC Policy Manager	8
4.2. SFC Catalog Manager	8
4.3. Developer's Management System	9
4.4. Classifier	9
4.5. Service Function Forwarder (SFF)	9
5. Use Cases	10
5.1. Dynamic Path Alternation	10
5.2. Enforcing Different SFPs Depending on Trust Levels	11
5.3. Effective Load Balancing with Dynamic SF Instantiation	12
6. Discussion	13
7. Security Considerations	14
8. Acknowledgements	14
9. References	15
9.1. Normative References	15
9.2. Informative References	15
Appendix A. Changes from draft-hyun-i2nsf-nsf-triggered-steering-02	16

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various security functions cooperatively analyze network traffic [RFC7498][i2nsf-problem][nsf-capability-im]. In addition, depending on the characteristics of network traffic and their suspiciousness level, the different types of network traffic need to be analyzed through the different sets of security functions. In order to meet such requirements, besides security policy rules for individual security functions, we need an additional policy about service function chaining (SFC) for network security which determines a set of security functions through which network traffic packets should pass for inspection. In addition, [nsf-capability-im] proposes an information model for NSFs capabilities that enables a security function to trigger further inspection by executing additional security functions 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 chaining between security functions.

In this document, we propose an architecture that integrates additional components from Service Function Chaining (SFC) into the I2NSF framework to support security function chaining. We extend the security controller's functionalities such that it can interpret a high-level policy of security function chaining into a low-level policy and manage them. It also keeps the track of the available service function (SF) instances for security functions and their information (e.g., network information and workload), and makes a decision on which SF instances to use for a given security function chain/path. Based on the forwarding information provided by the security controller, the service function forwarder (SFF) performs network traffic steering through various required security functions. A classifier is deployed for the enforcement of SFC policies given by the security controller. It performs traffic classification based on the policies so that the traffic passes through the required security function chain/path by the SFF.

2. Objective

- o Policy configuration for security function chaining: SFC-enabled I2NSF architecture allows policy configuration and management of security function chaining. Based on the chaining policy, relevant network traffic can be analyzed through various security functions in a composite, cooperative manner.
- o Network traffic steering for security function chaining: SFC-enabled I2NSF architecture allows network traffic to be steered through multiple required security functions based on the SFC

policy. Moreover, the I2NSF information model for NSFs capabilities [nsf-capability-im] requires a security function to call another security function for further inspection based on its own inspection result. To meet this requirement, SFC-enabled I2NSF architecture also enables traffic forwarding from one security function to another security function.

- o Load balancing over security function instances: SFC-enabled I2NSF architecture provides load balancing of incoming traffic over available 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 security function.

3. Terminology

This document uses the following terminology described in [RFC7665], [RFC7665][i2nsf-terminology][ONF-SFC-Architecture].

- o Service Function/Security Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers) [RFC7665]. In this document, SF is used to represent both Service Function and Security Function. Sample 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.
- o Classifier: An element that performs Classification. It uses a given policy from SFC Policy Manager.
- o Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification [RFC7665].
- o Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, an SFF is responsible for delivering traffic to a classifier when needed and supported, transporting traffic to another SFF (in the same or the different type of overlay), and terminating the Service Function Path (SFP) [RFC7665].

- o Service Function Path (SFP): The service function path is a constrained specification of where packets assigned to a certain service function path must be forwarded. While it may be so constrained as to identify the exact locations for packet processing, it can also be less specific for such locations [RFC7665].
- o SFC Policy Manager: It is responsible for translating a high-level policy into a low-level policy, and performing the configuration for SFC-aware nodes, passing the translated policy and configuration to SFC-aware nodes, and maintaining a stabilized network.
- o SFC Catalog Manager: It is responsible for keeping the track of the information of available SF instances. For example, the information includes the supported transport protocols, IP addresses, and locations for the SF instances.
- o Control Nodes: It collectively refer to SFC Policy Manager, SFC Catalog Manager, SFF, and Classifier.
- o Service Path Identifier (SPI): It identifies a service path. The classifier MUST use this identifier for path selection and the Control Nodes MUST use this identifier to find the next hop [sfc-nsh].
- o Service Index (SI): It provides a location within the service path. SI MUST be decremented by service functions or proxy nodes after performing the required services [sfc-nsh].
- o Network Service Header (NSH): The header is used to carry SFC related information. Basically, SPI and SI should be conveyed to the Control Nodes of SFC via this header.
- o SF Forwarding Table: SFC Policy Manager maintains this table. It contains all the forwarding information on SFC-enabled I2NSF architecture. Each entry includes SFF identifier, SPI, SI, and next hop information. For example, an entry ("SFF: 1", "SPI: 1", "SI: 1", "IP: 192.168.xx.xx") is interpreted as follows: "SFF 1" should forward the traffic containing "SPI 1" and "SI 1" to "IP=192.168.xx.xx".

4. Architecture

This section describes an SFC-enabled I2NSF architecture and the basic operations of service chaining. It also includes details about each component of the architecture.

Figure 1 describes the components of SFC-enabled I2NSF architecture. Our architecture is designed to support a composite inspection of traffic packets in transit. According to the inspection result of each SF, the traffic packets could be steered to another SF for further detailed analysis. It is also possible to reflect a high-level SFC-related policy and a configuration from I2NSF Client on the components of the original I2NSF framework. Moreover, the proposed architecture provides load balancing, auto supplementary SF generation, and the elimination of unused SFs. In order to achieve these design purposes, we integrate several components to the original I2NSF framework. In the following sections, we explain the details of each component.

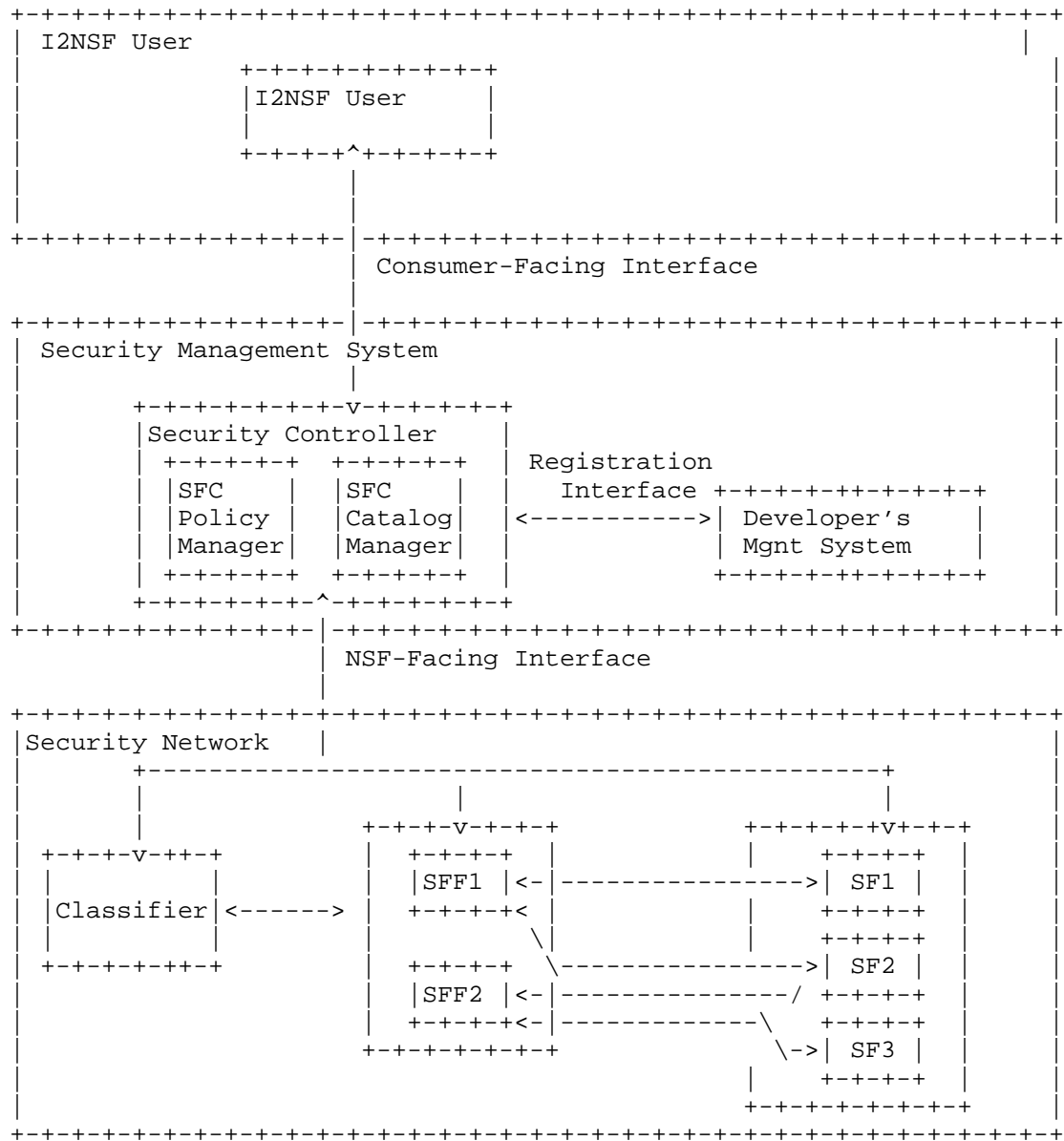


Figure 1: SFC-enabled I2NSF

4.1. SFC Policy Manager

SFC Policy Manager is a core component in our system. It is responsible for the following two things: (1) Interpreting a high-level SFC policy (or configuration) into a low-level SFC policy (or configuration), which is given by I2NSF Client, and delivering the interpreted policy to Classifiers for security function chaining. (2) Generating an SF forwarding table and distributing the forwarding information to SFF(s) by consulting with SFC Catalog Manager. As Figure 1 describes, SFC Policy Manager performs these additional functionalities through Consumer-Facing Interface and NSF-Facing Interface.

Given a high-level SFC policy/configuration from I2NSF Client via Consumer-Facing Interface, SFC Policy Manager interprets it into a low-level policy/configuration comprehensible to Classifier(s), and then delivers the resulting low-level policy to them. Moreover, SFC Policy Manager possibly generates new policies for the flexible change of traffic steering to rapidly react to the current status of SFs. For instance, it could generate new rules to forward all subsequent packets to "Firewall Instance 2" instead of "Firewall Instance 1" in the case where "Firewall Instance 1" is under congestion.

SFC Policy Manager gets information about SFs from SFC Catalog Manager to generate SF forwarding table. In the table generation process, SFC Policy Manager considers various criteria such as SFC policies, SF load status, SF physical location, and supported transport protocols. An entry of the SF forwarding table consists of SFF Identifier, SFP, SI, and next hop information. The examples of next hop information includes the IP address and supported transport protocols (e.g., VxLAN and GRE). These forwarding table updates are distributed to SFFs with either push or pull methods.

4.2. SFC Catalog Manager

In Figure 1, SFC Catalog Manager is a component integrated into Security Controller. It is responsible for the following three things: (1) Maintaining the information of every available SF instance such as IP address, supported transport protocol, service name, and load status. (2) Responding to the queries of available SF instances from SFC Policy Manager so as to help to generate a forwarding table entry relevant to a given SFP. (3) Requesting Developer's Management System to dynamically instantiate supplementary SF instances to avoid service congestion or the elimination of an existing SF instance to avoid resource waste.

Whenever a new SF instance is registered, Developer's Management

System passes the information of the registered SF instance to SFC Catalog Manager, so SFC Catalog Manager maintains a list of the information of every available SF instance. Once receiving a query of a certain SFP from SFC Policy Manager, SFC Catalog Manager searches for all the available SF instances applicable for that SFP and then returns the search result to SFC Policy Manager.

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

4.3. Developer's Management System

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

4.4. Classifier

Classifier is a logical component that may exist as a standalone component or a submodule of another component. In our system, the initial classifier is typically located at an entry point like a border router of the network domain, and performs the initial classification of all incoming packets according to the SFC policies, which are given by SFC policy manager. The classification means determining the SFP through which a given packet should pass. Once the SFP is decided, the classifier constructs an NSH that specifies the corresponding SPI and SI, and attaches it to the packet. The packet will then be forwarded through the determined SFP on the basis of the NSH information.

4.5. Service Function Forwarder (SFF)

It is responsible for the following two functionalities: (1) Forwarding the packets to the next SFF/SF. (2) Handling re-classification request from SF.

An SFF basically takes forwarding functionality, so it needs to find the next SF/SFF for the incoming traffic. It will search its forwarding table to find the next hop information that corresponds to the given traffic. In the case where the SFF finds a target entry on its forwarding table, it just forwards the traffic to the next SF/SFF specified in the next hop information. If an SFF does not have an entry for a given packet, it will request the next hop information to SFC Policy Manager with SFF identifier, SPI, and SI information. The SFC Policy Manager will respond to the SFF with next hop information, and then the SFF updates its forwarding table with the response, forwarding the traffic to the next hop.

Sometimes an SF may want to forward a packet, which is highly suspicious, to another SF for further security inspection. This is referred to as advanced security action in I2NSF. In this situation, if the next SF may not be the one on the current SFP of the packet, re-classification is required to change the SFP of the packet. If the current SF is capable of re-classifying the packet by itself, the SF updates the SPI field in the NSH in the packet to serve the advanced security action. Otherwise, if the classifier exists as a standalone, the SF appends the inspection result of the packet to the MetaData field of the NSH and delivers it to the source SFF. The attached MetaData includes a re-classification request to change the SFP of the packet to another SFP for stronger inspection. When the SFF receives the traffic requiring re-classification, it forwards the traffic to the Classifier where re-classification will be eventually performed.

SFC defines Rendered Service Path (RSP), which represents the sequence of actual visits by a packet to SFFs and SFs [RFC7665]. If the RSP information of a packet is available, the SFF could check this RSP information to detect whether undesired looping happened on the packet. If the SFF detects looping, it could notify the Security Controller of this looping, and the Security Controller could modify relevant security policy rules to resolve this looping.

5. Use Cases

This section introduces three use cases for the SFC-enabled I2NSF architecture : (1) Dynamic Path Alternation, (2) Enforcing Different SFPs Depending on Trust Levels, and (3) Effective Load Balancing with Dynamic SF Instantiation.

5.1. Dynamic Path Alternation

In SFC-enabled I2NSF architecture, a Classifier determines the initial SFP of incoming traffic according to the SFC policies. The classifier then attaches an NSH specifying the determined SFP of the

packets, and they are analyzed through the SFs of the initial SFP. However, SFP is not a static property, so it could be changed dynamically through re-classification. A typical example is for a certain SF in the initial SFP to detect that the traffic is highly suspicious (likely to be malicious). In this case, the traffic needs to take stronger inspection through a different SFP which consists of more sophisticated SFs.

Figure 2 illustrates an example of such dynamic SFP alternation in a DDoS attack scenario. SFP-1 represents the default Service Function Path that the traffic initially follows, and SFP-1 consists of AVC, Firewall, and IDS/IPS. If the IDS/IPS suspects that the traffic is attempting DDoS attacks, it will change the SFP of the traffic from the default to SFP-2 so that the DDoS attack mitigator can execute a proper countermeasure against the attack.

Such SFP alternation is possible in the proposed architecture with re-classification. In Figure 1, to initiate re-classification, the IDS/IPS appends its own inspection result to the MetaData field of NSH and deliver it to the SFF from which it has originally received the traffic. The SFF then forwards the received traffic including the inspection result from the IDS/IPS to Classifier for re-classification. Classifier checks the inspection result and determines the new SFP (SFP-2) associated with the inspection result in the SFC policy, and updates the NSH with the SPI of SFP-2. The traffic is forwarded to the DDoS attack mitigator.

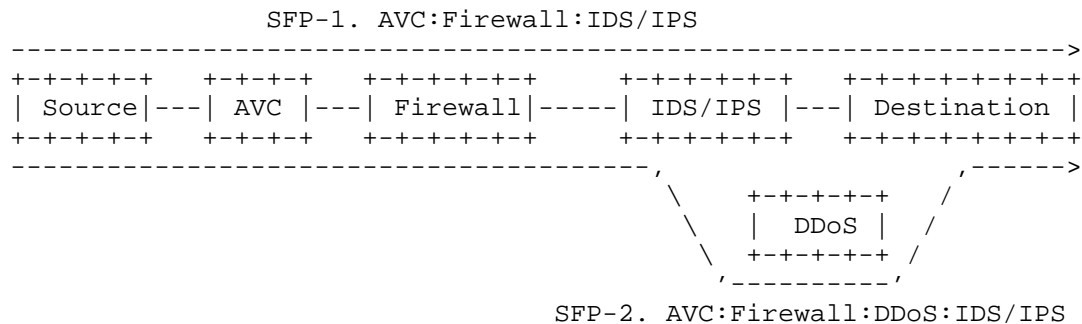


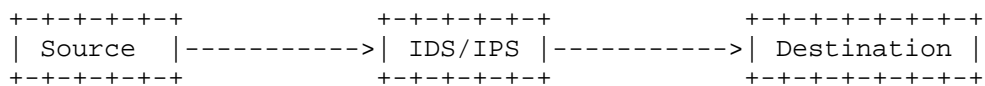
Figure 2: Dynamic SFP Alternation Example

5.2. Enforcing Different SFPs Depending on Trust Levels

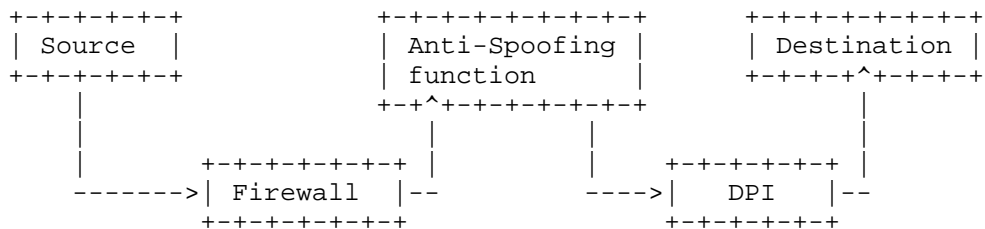
Because the traffic coming from a trusted source is highly likely to be harmless, it does not need to be inspected excessively. On the other hand, the traffic coming from an untrusted source requires an

in-depth inspection. By applying minimum required security functions to the traffic from a trusted source, it is possible to prevent the unnecessary waste of resources. In addition, we can concentrate more resources on potential malicious traffic. In the SFC-enabled I2NSF architecture, by configuring an SFC Policy to take into account the levels of trust of traffic sources, we can apply different SFPs to the traffic coming from different sources.

Figure 3(a) and Figure 3(b) represent SFPs applicable to traffic from trusted and untrusted sources, respectively. In Figure 3(a), we assume a lightweight IDS/IPS which is configured to perform packet header inspection only. In this scenario, when receiving the traffic from a trusted source, the classifier determines the SFP in Figure 3(a) such that the traffic passes through just a simple analysis by the lightweight IDS/IPS. On the other hand, traffic from an untrusted source passes more thorough examination through the SFP in Figure 3(b) which consists of three different types of SFs.



(a) Traffic flow of trusted source



(b) Traffic flow of untrusted source

Figure 3: Different path allocation depending on source of traffic

5.3. Effective Load Balancing with Dynamic SF Instantiation

In a large-scale network domain, there typically exist a large number of SF instances that provide various security services. It is possible that a specific SF 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 SF 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 the SFC-enabled I2NSF architecture, SFC Catalog Manager performs periodic monitoring of the load status of available SF instances. In addition, it is possible to dynamically generate a new SF 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. SFC Catalog Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.
2. SFC Catalog 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 SFC Catalog Manager.
4. SFC Catalog Manager updates the SFC Information Table to reflect the new firewall instance, and notifies SFC Policy Manager of this update.
5. Based on the received information, SFC Policy Manager updates the forwarding information for traffic steering and sends the new forwarding information to the SFF.
6. 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. Discussion

The information model and data model of security policy rules in the I2NSF framework defines an advanced security action as a type of action to be taken on a packet [nsf-capability-im][nsf-facing-inf-dm]. Through the advanced security action, a basic NSF (e.g., firewall) can call a different

type of NSF for more in-depth security analysis of a packet. If an NSF triggers an advanced security action on a given packet, the packet should be forwarded to the NSF dedicated to the advanced action. That is, the advanced action dynamically determines the next NSF where the packet should go through. So if a forwarding component is configured with the network access information (e.g., IP address, port number) of the next required NSF, it can forward the packet to the NSF. With this advanced security action, it is possible to avoid the overhead for configuring and managing the information of security function chains and paths.

In SFC, re-classification is required to support the situation where the security function path of a packet changes dynamically, and the classifier is responsible for re-classification tasks to change the security function path of a packet. But if the classifier exists as a separate component from an NSF, the packet should be first delivered from the NSF to the classifier for re-classification, and this introduces an additional delay. As already mentioned, the advanced security action in the i2nsf framework can omit the requirement of pre-defined security function chain configuration. If there exists no security function chain/path configurations, there is no need of re-classification as well. That is, the forwarder can simply forward the packet to the next required NSF according to the advanced action determined by the predecessor NSF, without re-classification through the classifier.

7. Security Considerations

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

8. Acknowledgements

This work was supported by Institute for Information and communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This document has greatly benefited from inputs by Sanguk Woo, Yunsuk Yeo, Taekyun Roh, and Sarang Wee.

9. References

9.1. Normative References

- [RFC7665] Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015.
- [sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 2017.

9.2. Informative References

- [RFC7498] Quinn, P. and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, April 2015.
- [nsf-capability-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01 (work in progress), March 2017.
- [nsf-facing-inf-dm] Kim, J., Jeong, J., Park, J., Hares, S., and L. Xia, "I2NSF Network Security Functions-Facing Interface YANG Data Model", draft-kim-i2nsf-nsf-facing-interface-data-model-02 (work in progress), July 2017.
- [i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 2017.
- [i2nsf-problem] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-16 (work in progress), May 2017.
- [i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.
- [ONF-SFC-Architecture] ONF, "L4-L7 Service Function Chaining Solution Architecture", June 2015.

Appendix A. Changes from draft-hyun-i2nsf-nsf-triggered-steering-02

The following changes have been made from
draft-hyun-i2nsf-nsf-triggered-steering-02:

- o Sections 3, 4, and 5 have been revised to describe an architecture, which integrates additional components of service function chaining (SFC) into the I2NSF framework in order to support packet forwarding between NSFs.
- o Section 6 has been added to discuss some drawbacks when SFC is used for packet forwarding between NSFs in the I2NSF framework.

Authors' Addresses

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: <http://imtl.skku.ac.kr/>

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

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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

S. Hyun
J. Jeong
R. Roh
W. Wi
Sungkyunkwan University
J. Park
ETRI
July 3, 2017

I2NSF Registration Interface YANG Data Model
draft-hyun-i2nsf-registration-interface-dm-01

Abstract

This document describes a YANG data model for I2NSF Registration Interface between Security Controller and Developer's Management System. The data model is required for the instance registration of Network Security Functions (NSF) and the dynamic life cycle management of NSF instances.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Terminology	3
3.1. Tree Diagrams	4
4. High-Level YANG	4
4.1. Registration Interface	4
4.2. Registration Request	4
4.3. Life-Cycle Management Request	5
4.4. NSF Profile	5
4.5. NSF Access Information	6
4.6. NSF Performance Capability	6
5. YANG Modules	7
6. Security Considerations	13
7. Acknowledgments	14
8. References	14
8.1. Normative References	14
8.2. Informative References	14
Appendix A. Changes from draft-hyun-i2nsf-registration-interface-dm-00	15

1. Introduction

This document provides a YANG [RFC6020] data model that defines the required data for I2NSF Registration Interface between Security Controller and Developer's Management System to dynamically manage a pool of the instances of Network Security Functions (NSF). This document defines a YANG data model based on the [i2nsf-reg-inf-im]. The terms used in this document are defined in [i2nsf-terminology].

2. Requirements Language

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

3. Terminology

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

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

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

3.1. Tree Diagrams

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

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

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

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

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

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

4. High-Level YANG

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

4.1. Registration Interface

```
module : ietf-i2nsf-regs-interface
  +--rw regs-req
  |   uses i2nsf-regs-req
  +--rw life-cycle-mgmt-req
  |   uses i2nsf-life-cycle-mgmt-req
```

Figure 1: High-Level YANG of I2NSF Registration Interface

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

4.2. Registration Request

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

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

Figure 2: High-Level YANG of I2NSF Registration Request

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

4.3. Life-Cycle Management Request

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

```
Life-Cycle Management Request
+--rw i2nsf-life-cycle-mgmt-req
  +--rw req-level uint16
  +--rw req-id uint64
  +--rw (req-type)?
    +--rw (req-creation-type)
      +--rw nsf-profile
        |   uses i2nsf-nsf-profile
    +--rw (req-elimination-type)
      +--rw nsf-access-info
        |   uses i2nsf-nsf-access-info
```

Figure 3: High-Level YANG of I2NSF Life Cycle Mgnt Request

Life-cycle management request consists of two types: `req-creation-type` and `req-elimination-type`. The creation type is used to request generation of a new NSF instance with NSF Profile which specifies required NSF capability information. The elimination type is used to remove an existing NSF with NSF Access Information.

4.4. NSF Profile

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

```
NSF Profile
+--rw i2nsf-nsf-profile
  +--rw i2nsf-capability
    |   uses ietf-i2nsf-capability
  +--rw performance-capability
    |   uses i2nsf-nsf-performance-caps
```

Figure 4: High-Level YANG of I2NSF NSF Profile

In Figure 4, `ietf-i2nsf-capability` refers module `ietf-i2nsf-capability` in `[i2nsf-capability-dm]`. we add the performance capability because it is absent in `[i2nsf-capability-dm]`.

4.5. NSF Access Information

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

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

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

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

4.6. NSF Performance Capability

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

```

NSF Performance Capability
+--rw i2nsf-nsf-performance-caps
+--rw vcpus
|   +--rw cpu-num uint16
|   +--rw cpu-topology
|       +-- rw cpu-cores uint16
|       +-- rw cpu-socket uint16
|       +-- rw cpu-threads uint16
|   +--rw (cpu-limit uint16)?
|   +--rw (cpu-reservation uint16)?
+--rw disk
|   +--rw disk-size uint16
|   +--rw (disk-limit uint16)?
|   +--rw (disk-reservation uint16)?
+--rw memory
|   +--rw memory-size uint16
|   +--rw (memory-limit uint16)?
|   +--rw (memory-reservation uint16)?
+--rw bandwidth
|   +--rw outbound
|       +--rw outbound-average uint16
|       +--rw outbound-peak uint16
|   +--rw inbound
|       +--rw inbound-average uint16
|       +--rw inbound-peak uint16

```

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

When the Security Controller asks the Developer Mgmt System to create a new NSF instance, the performance capability is used to specify the spec of the new instance.

5. YANG Modules

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

```

<CODE BEGINS> file "ietf-i2nsf-regs-interface@2017-07-03.yang"
module ietf-i2nsf-regs-interface {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-regs-interface";
  prefix
    regs-interface;
  import ietf-inet-types{
    prefix inet;

```

```
}

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

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

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

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

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

  Editor: Taekyun Roh
  <mailto:tkroh@imtl.skku.ac.kr>

  Editor: Sarang Wi
  <mailto:sarang@imtl.skku.ac.kr>

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

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

description
  "It defines a YANG data module for Registration Interface.";

revision "2017-07-03"{
  description "Initial revision";
  reference
    "draft-hares-i2nsf-capability-data-model-01.txt
    draft-hyun-i2nsf-registration-interface-im-01.txt";
}

grouping i2nsf-nsf-performance-caps {
  description
    "NSF performance capabilities";

  container vcpus{
```

```
description
  "vcpus info";
  container cpu-topology{
    description
      "cpu-topology";
    leaf cores{
      type uint16;
      description
        "cpu-cores";
    }

    leaf cpu-socket{
      type uint16;
      description
        "cpu-socket";
    }

    leaf cpu-threads{
      type uint16;
      description
        "cpu-threads";
    }
  }

  choice cpu-limitation{
    description
      "cpu-limitation";
    leaf cpu-limit{
      type uint16;
      description
        "cpu-limit";
    }
  }

  choice cpu-reservation{
    description
      "cpu-reservation";
    leaf cpu-reserve{
      type uint16;
      description
        "cpu-reserve";
    }
  }

  leaf cpu-num{
    type uint16;
    description
      "cpu-num";
```

```
    }  
  }  
  
  container disk{  
    description  
      "disk info";  
    leaf disk-size{  
      type uint16;  
      description  
        "disk-size";  
    }  
  
    choice disk-limitation{  
      description  
        "disk-limitation";  
      leaf disk-limit{  
        type uint16;  
        description  
          "disk-limit";  
      }  
    }  
  
    choice disk-reservation{  
      description  
        "disk-reservation";  
      leaf disk-reserve{  
        type uint16;  
        description  
          "disk-reserve";  
      }  
    }  
  }  
  
  container memory{  
    description  
      "memory info";  
    leaf memory-size{  
      type uint16;  
      description  
        "memory-size";  
    }  
    choice memory-limitation{  
      description  
        "memory-limitation";  
      leaf memory-limit{  
        type uint16;  
        description  
          "memory-limit";  
      }  
    }  
  }  
}
```

```
    }
  }
  choice memory-reservation{
    description
    "memory-reservation";
    leaf memory-reserve{
      type uint16;
      description
      "memory-reserve";
    }
  }
}
container bandwidth{
  description
  "bandwidth info";
  container inbound{
    description
    "inbound";
    leaf inbound-average{
      type uint16;
      description
      "inbound-average";
    }
    leaf inbound-peak{
      type uint16;
      description
      "inbound-peak";
    }
  }
  container outbound{
    description
    "outbound";
    leaf outbound-average{
      type uint16;
      description
      "outbound-average";
    }
    leaf outbound-peak{
      type uint16;
      description
      "outbound-peak";
    }
  }
}
container i2nsf-nsf-profile {
  description
  "Detail information of an NSF";
  container performance-capability {
    description
```



```
        "performance-capability";
    }
}

    container i2nsf-capability {
        description
            "It refers draft-hares-i2nsf-capability-data-model-01";
    }
}

grouping i2nsf-nsf-access-info {
    description
        "NSF access information";
    leaf nsf-address {
        type inet:ipv4-address;
        mandatory true;
        description
            "nsf-address";
    }
    leaf nsf-port-address {
        type inet:port-number;
        description
            "nsf-port-address";
    }
}

    container i2nsf-regs-req {
        description
            "The capability information of newly created NSF to notify its
            capability to Security Controller";
        container nsf-profile {
            description
                "i2nsf-nsf-profile";
        }
        container nsf-access-info {
            description
                "nsf-access-info";
            uses i2nsf-nsf-access-info;
        }
    }

    container i2nsf-life-cycle-mgmt-req {
        description
            "Required information for req-creation-type and
            req-elimination-type";
        leaf req-level {
            type uint16;
            description
```

```

    "req-level";
}
leaf req-id {
    type uint64;
    mandatory true;
    description
        "req-id";
}
choice req-type {
    description
        "req-type";
    case req-creation-type {
        description
            "req-creation-type";
    }
    case req-elimination-type{
        description
            "req-elimination-type";
        container nsf-access-info {
            description
                "nsf-access-info";
            uses i2nsf-nsf-access-info;
        }
    }
}
container nsf-profile {
    description
        "i2nsf-nsf-profile";
}
}
}
}
<CODE ENDS>
```

Figure 7: Data Model of I2NSf Registration Interface

6. Security Considerations

The information model of the I2NSF Registration Interface is based on the I2NSF framework without any architectural changes. Thus, this document shares the security considerations of the I2NSF framework architecture that are specified in [i2nsf-framework] for the purpose of achieving secure communication among components in the proposed architecture.

7. Acknowledgments

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

8. References

8.1. Normative References

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

8.2. Informative References

- [capability-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01 (work in progress), March 2017.
- [i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 2017.
- [i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.
- [nsf-triggered-steering] Hyun, S., Woo, S., Yeo, Y., Jeong, J., and J. Park, "NSF-Triggered Traffic Steering", draft-hyun-i2nsf-nsf-triggered-steering-in-i2nsf-02 (work in progress), March 2017.
- [i2nsf-reg-inf-im] Hyun, S., Woo, S., Yeo, Y., Jeong, J., and J. Park, "Registration Interface

Information Model", draft-hyun-i2nsf-registration-interface-im-02 (work in progress), March 2017.

[i2nsf-capability-dm] Hares, S., Moskowitz, R., Xia, L., Jeong, J., and J. Kim, "I2NSF Capability YANG Data Model", i2nsf-capability draft-hares-i2nsf-capability-data-model-01, March 2017.

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

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

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

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

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

- o The description of NSF Performance Capability is specified in more detail than the previous version.
- o The description of YANG data model for Registration interface was clarified.

Authors' Addresses

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: <http://imtl.skku.ac.kr/>

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

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

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: tkroh@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: sarang@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

S. Hyun
J. Jeong
S. Woo
Y. Yeo
Sungkyunkwan University
J. Park
ETRI
July 3, 2017

I2NSF Registration Interface Information Model
draft-hyun-i2nsf-registration-interface-im-02

Abstract

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Terminology	3
4. Objectives	4
5. Information Model	5
5.1. Life-Cycle Managment Mechanism	6
5.2. Registration Mechanism	6
5.3. NSF Access Information	7
5.4. NSF Profile (Capabilities of an NSF instance)	7
5.4.1. Packet Content-Matching Capability	8
5.4.2. Content-Matching Capability	8
5.4.3. Context-Matching Capability	8
5.4.4. Attack-Mitigation Capability	9
5.4.5. Action Capability	9
5.4.6. Performance Capability	9
6. Security Considerations	11
7. Acknowledgements	11
8. References	11
8.1. Normative References	11
8.2. Informative References	11
Appendix A. Changes from draft-hyun-i2nsf-registration-interface-im-01	12

1. Introduction

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

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

2. Requirements Language

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

3. Terminology

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

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy [nsf-triggered-steering].

- o Advanced Inspection/Action: As like the I2NSF information model for NSF-facing interface [nsf-capability-im], Advanced Inspection/Action means that a security function calls another security function for further inspection based on its own inspection result [nsf-triggered-steering].
- o Network Security Function Profile (NSF Profile): NSF Profile specifies the security and performance capability of an NSF instance. Each NSF instance has its own NSF Profile which describes the type of security service it can provide and its resource capacity. [nsf-triggered-steering].

4. Objectives

- o Efficient network resource utilization through dynamic instantiation of NSFs and load balancing: In I2NSF framework, it is sometimes possible that a specific NSF experiences heavy traffic loads. For example, under DDoS attacks, a huge volume of traffic would be delivered to DoS attack mitigator function to cope with the attacks. In this case, we should allocate a large portion of resources to that DoS attack mitigator function by creating a sufficient number of DoS mitigator instances. In addition, after the attack is terminated, we should eliminate some of the instances no longer used. In this way, we can achieve efficient resource utilization. For this purpose, it is essential to define an information model of registration interface for dynamic instantiation/elimination of NSF instances.
- o Creating an NSF instance to serve another NSF's inspection request: In I2NSF framework, an NSF can trigger another type of NSF(s) for more advanced security inspection of the traffic. In this case, the next NSF is determined by the current NSF's inspection result and client's policy. However, if there is no available NSF instance to serve the former NSF's request, we should create an NSF instance by requesting Developer's Management System (DMS) through registration interface.
- o Creating NSF instances required to enforce security policy rules from Client: In I2NSF framework, users decide which security service is necessary in the system. If there is no NSF instances to enforce the client's security policy, then we should also create the required instances by requesting DMS via registration interface.
- o Registering NSF instances from Developer's Management System: Depending on system's security requirements, it may require some NSFs by default. In this case, DMS creates these default NSF instances without the need of receiving requests from Security

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

5. Information Model

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

- 1) Security Controller first recognizes the set of capabilities (i.e., NSF Profile) or the signature of a specific NSF required or wasted in the current system.
- 2) Developer's Management System (DMS) matches the recognized information to an NSF based on the information model definition.
- 3) Developer's Management System creates or eliminates NSFs matching with the above information.
- 4) Security Controller can then add/remove the corresponding NSF instance to/from its list of available NSF instances in the system.

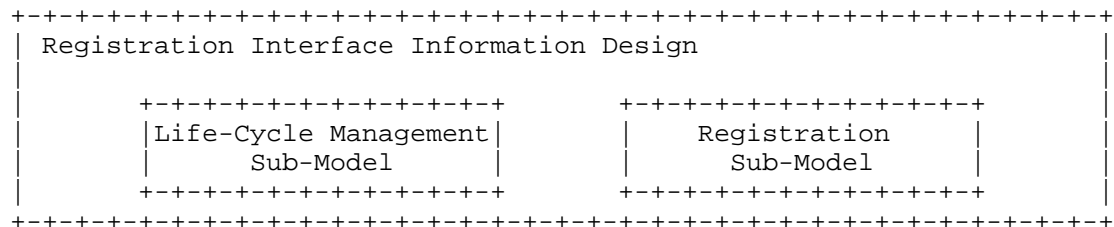


Figure 1: The Registration Interface Information Model Design

As illustrated in Figure 1, the information model for Registration Interface consists of two sub-models: life-cycle management, registration sub-models. The life-cycle management functionality and the registration functionality use NSF Profile to achieve their goals. In this context, NSF Profile is the capability objects that

describe and/or prescribe inspection capability an NSF instance can provide.

5.1. Life-Cycle Managment Mechanism

For the life-cycle management of NSFs, Security Controller in I2NSF framework requires two types of requests: Instance Creation and Elimination Request Messages. Security Controller sends the request messages to DMS when required. Once receiving the request, DMS conducts creating/eliminating the corresponding NSF instance and responds Security Controller with the results. There are several cases requiring creation of a new NSF instance which provides specific security inspection functionalities and elimination of an existing NSF which is unused for a period of time. For example,

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

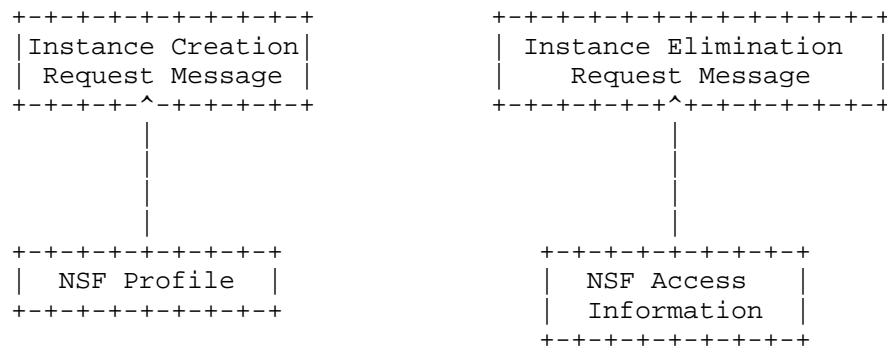


Figure 2: Life-Cycle Management Sub-Model Overview

5.2. Registration Mechanism

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

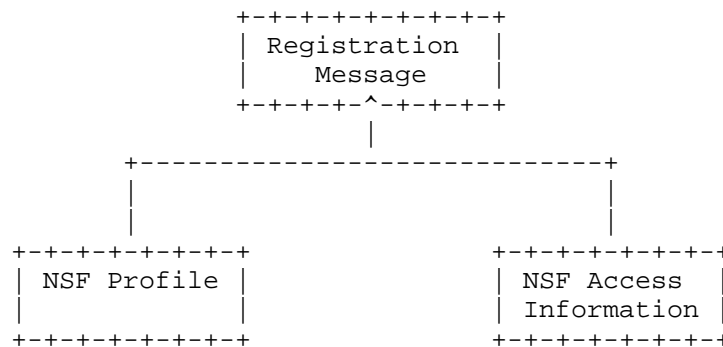


Figure 3: Registration Mechanism Sub-Model Overview

5.3. NSF Access Information

NSF Access Information contains the followings that are required to communicate with an NSF: IPv4 address, IPv6 address, port number, and supported transport protocol(s) (e.g., Virtual Extensible LAN (VXLAN) [RFC 7348], Generic Protocol Extension for VXLAN (VXLAN-GPE) [nvo3-vxlan-gpe], Generic Route Encapsulation (GRE), Ethernet etc.). In this document, NSF Access Information is used to identify a specific NSF instance (i.e. NSF Access Information is the signature(unique identifier) of an NSF instance in the overall system).

5.4. NSF Profile (Capabilities of an NSF instance)

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

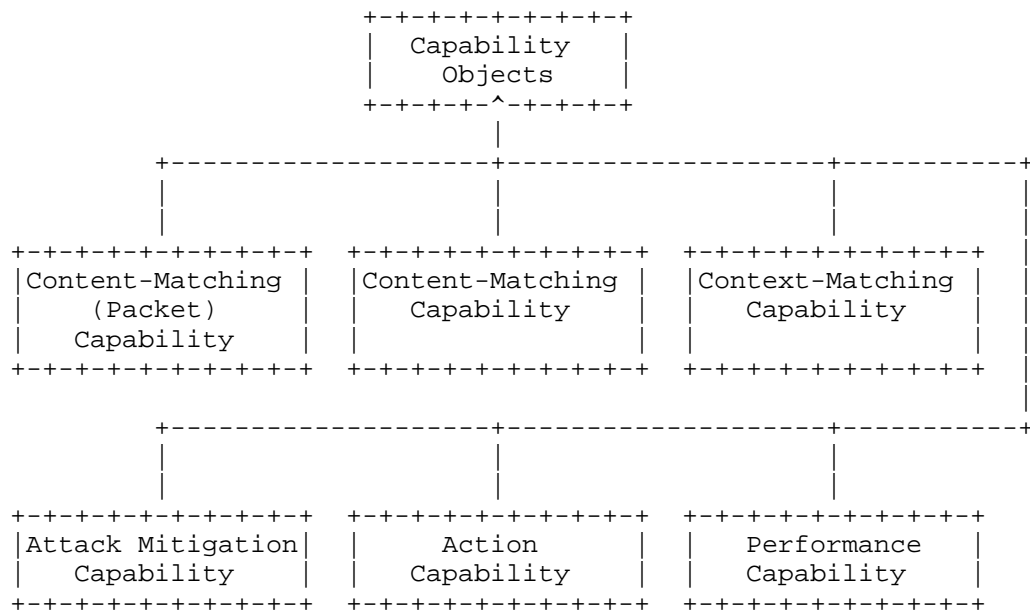


Figure 4: NSF Profile Overview

5.4.1. Packet Content-Matching Capability

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

5.4.2. Content-Matching Capability

Content security is another category of security capabilities applied to application layer. Through detecting the contents carried over the traffic in application layer, these capabilities can realize various security functions, such as defending against intrusion, inspecting virus, filtering malicious URL or junk email, blocking illegal web access or malicious data retrieval. [nsf-capability-im]

5.4.3. Context-Matching Capability

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

5.4.4. Attack-Mitigation Capability

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

- 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, ssl DDoS.
- o Single-packet attack:
 - * Scanning and sniffing attacks: IP sweep, port scanning, etc
 - * Malformed packet attacks: Ping of Death, Teardrop, etc
 - * Special packet attacks: Oversized ICMP, Tracert, IP timestamp option packets, etc

5.4.5. Action Capability

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

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

5.4.6. Performance Capability

This information represents the processing capability of an NSF. This information can be used to determine whether the NSF is in congestion by comparing this with the workload that the NSF currently

undergoes. Moreover, this information can specify an available amount of each type of resources such as processing power and memory which are available on the NSF. (The registration interface can control the usages and limitations of the created instance and make the appropriate request according to the status.) As illustrated in Figure 5, this information consists of four items: vCPUs, Disk, Memory, and Bandwidth.

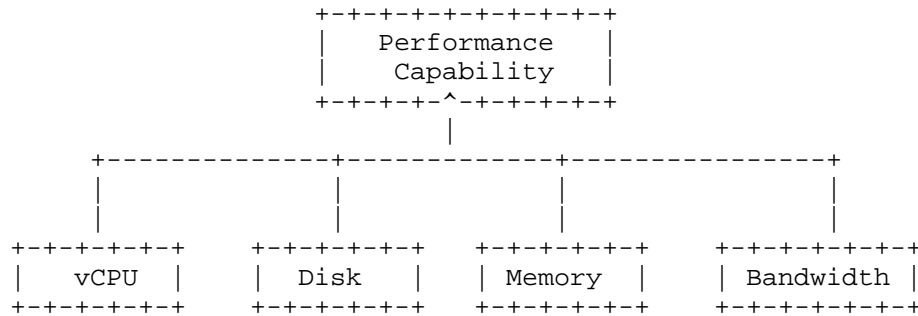


Figure 5: Performance Capability Overview

5.4.6.1. vCPU

This information specifies the details of a virtual CPU (vCPU) available on an NSF. With this information, it is possible to configure the vCPU topology by setting the number of processing cores, socket type, and the number of threads per core. This information can also specify the upper limit of the processing power and the minimum reserved processing power of vCPU.

5.4.6.2. Disk

This information specifies the total size of the disk (in gigabyte). In addition, the capability of the disk can be expressed as the number of I/O operations per second (IOPS) this disk can perform. This information can also specify two attributes of the disk: disk limit and disk reservation. The disk limit represents the maximum limit of the disk space, and the disk reservation represents the minimum guaranteed space of the disk.

5.4.6.3. Memory

This information has two attributes to describe an available memory resource: memory limit and memory reservation. The memory limit specifies the upper limit of an available memory in MB, and the memory reservation specifies the guaranteed minimum amount of memory

in MB.

5.4.6.4. Bandwidth

This information specifies the networking capability of an NSF. This information can have separate attributes for each direction of inbound and outbound. For each direction, this information specifies the amount of traffic this NSF can send/receive per second (kilobytes/sec).

6. Security Considerations

The information model of the registration interface is based on the I2NSF framework without any architectural changes. Thus, this document shares the security considerations of the I2NSF framework that are specified in [i2nsf-framework] for the purpose of achieving secure communication between components in the proposed architecture.

7. Acknowledgements

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

8. References

8.1. Normative References

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

8.2. Informative References

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

[i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 2017.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia,

L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.

[nsf-triggered-steering] Hyun, S., Jeong, J., Park, J., and S. Hares, "NSF-Triggered Traffic Steering", draft-hyun-i2nsf-nsf-triggered-steering-03 (work in progress), July 2017.

[nvo3-vxlan-gpe] Maino, Ed., F., Kreeger, Ed., L., and U. Elzur, Ed., "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.

Appendix A. Changes from draft-hyun-i2nsf-registration-interface-im-01

The following changes are made from draft-hyun-i2nsf-registration-interface-im-01:

- o The description of NSF access information and performance capability is specified in more detail than the previous version.
- o Miscellaneous expressions in the whole descriptions are corrected.

Authors' Addresses

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: <http://imtl.skku.ac.kr/>

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

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

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: suwoo@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: yunsuk@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

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

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislaamovic
Nokia
L. Xia
Huawei
July 3, 2017

Requirements for Client-Facing Interface to Security Controller
draft-ietf-i2nsf-client-facing-interface-req-02

Abstract

This document captures requirements for Client-Facing interface to the Security Controller as defined by [I-D.ietf-i2nsf-framework]. 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 the User Community. This gives ability to decouple policy definition from policy enforcement on a specific security functional 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 January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Conventions Used in this Document	4
3. Guiding principle for Client-Facing Interface definition . .	5
3.1. User-construct based modeling	5
3.2. Basic rules for Client-Facing Interface definition . . .	6
3.3. Deployment Models for Implementing Security Policies . .	7
4. Functional Requirements for the Client-Facing Interface . . .	10
4.1. Requirement for Multi-Tenancy in Client-Facing interface	11
4.2. Requirement for Authentication and Authorization of	
Client-Facing interface	12
4.3. Requirement for Role-Based Access Control (RBAC) in	
Client-Facing interface	12
4.4. Requirement to protect Client-Facing interface from	
attacks	13
4.5. Requirement to protect Client-Facing interface from	
misconfiguration	13
4.6. Requirement to manage policy lifecycle with rich set of	
controls	13
4.7. Requirement to define dynamic Policy Endpoint Group . . .	14
4.8. Requirement to express rich set of Policy Rules	16
4.9. Requirement to express rich set of Policy Actions	17
4.10. Requirement for consistent policy enforcement	19
4.11. Requirement to detect and correct policy conflicts	19
4.12. Requirement for backward compatibility	19
4.13. Requirement for Third-Party integration	20
4.14. Requirement to collect telemetry data	20
5. Operational Requirements for the Client-Facing Interface . .	20
5.1. API Versioning	20
5.2. API Extensibility	21
5.3. APIs and Data Model Transport	21
5.4. Notification and Monitoring	21

5.5. Affinity	21
5.6. Test Interface	21
6. Security Considerations	22
7. IANA Considerations	22
8. Acknowledgements	22
9. Normative References	22
Authors' Addresses	23

1. Introduction

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

Even if 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 that requires mapping of business needs to complex security features with vendor specific configurations. 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:

- o 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.
- o 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.
- o 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.
- o An organization may deploy multiple security functions in either virtual or physical form 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 a given 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 Policy in an abstract form with choice of wide variety of NSF as policy enforcement point. The implementation of security management applications or controller is out of scope for I2NSF working group.

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-form 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 Security 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 definition

Client-Facing Interface must ensure that a Security Admin can deploy a NSF from any vendor and should still be able to use the same consistent interface. In essence, this interface allows a Security Admin to express a Security Policy enforced on the NSF to be independent of vendor and its implementation. 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 vendor 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. To further complicate the matters, when changes happen in the network topology, the Security Policy may require modifications accordingly. This may be a highly manual task based on network design and becomes unmanageable in virtualized environment.

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:

- o Not dependent on a particular network topology or the NSF location in the network
- o Not forced to express Security Policy with proprietary vendor specific interfaces for a given NSF
- o Independent of NSF type that will implement a specific Security Policy; e.g., the interface remains same no matter if a specific Security Policy is enforced on a stateful firewall, IDP, IDS, Router or a Switch
- o Declarative/Descriptive model instead of Imperative/Prescriptive model - What security policy need to be expressed (declarative) instead of how it is implemented (imperative)
- o Not dependent on vendor's' implementation or form-factor (physical, virtual) of the NSF
- o Not dependent on how a NSF becomes operational - network connectivity and other hosting requirements.
- o 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.
- o Not depending on specific data plane implementation of NSF, e.g. encapsulation, service function chains.

Note that the rules stated above only apply to the Client-Facing interface, which a Security Admin 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 Enterprises deploy vendor provided management systems to create Security Policies and any changes to these Security Policies are made manually over time by Security Admin. This approach may not be suitable and nor sufficient for modern highly automated data centers that are largely virtualized and rely on various management systems and controllers to implement dynamic Security Policies over large number of NSF in the network.

There are two distinct deployment models for Security Controller. Although, these have no direct impact on the Client-Facing interface, but illustrate the overall Security Policy management framework in an organization and how the Client-Facing interface remain same which is the main objective of this document. 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.

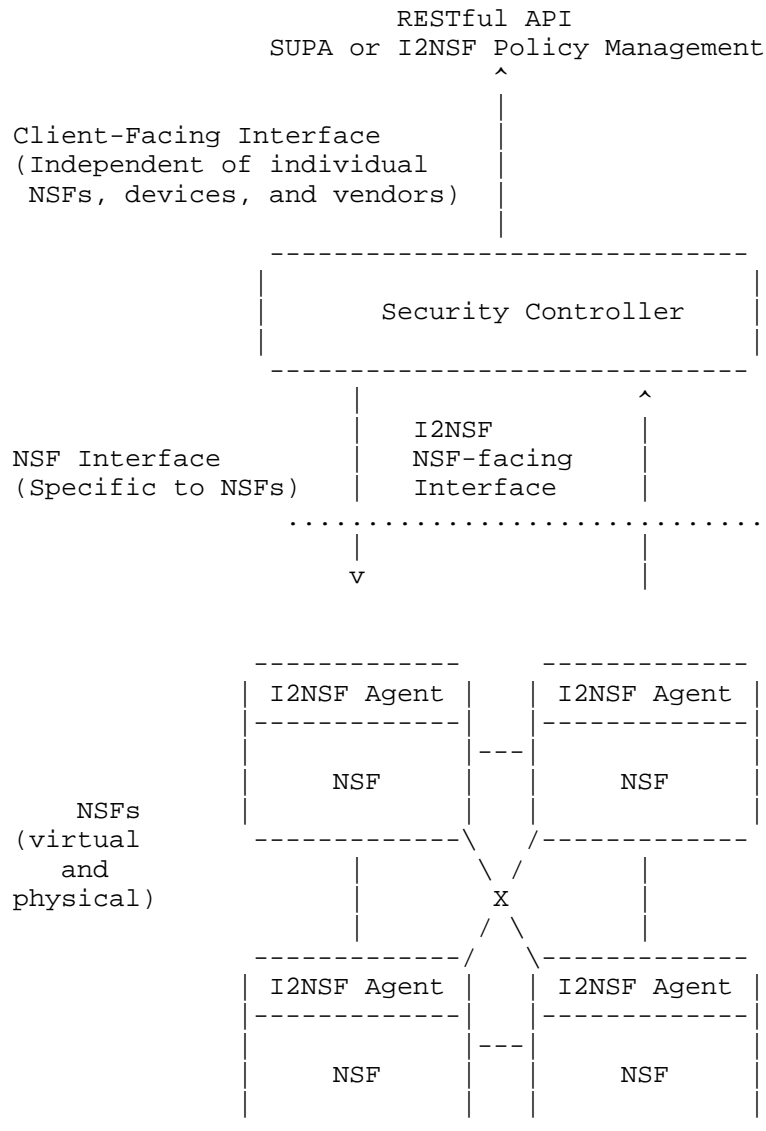


Figure 1: Deployment without Management System

- 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.

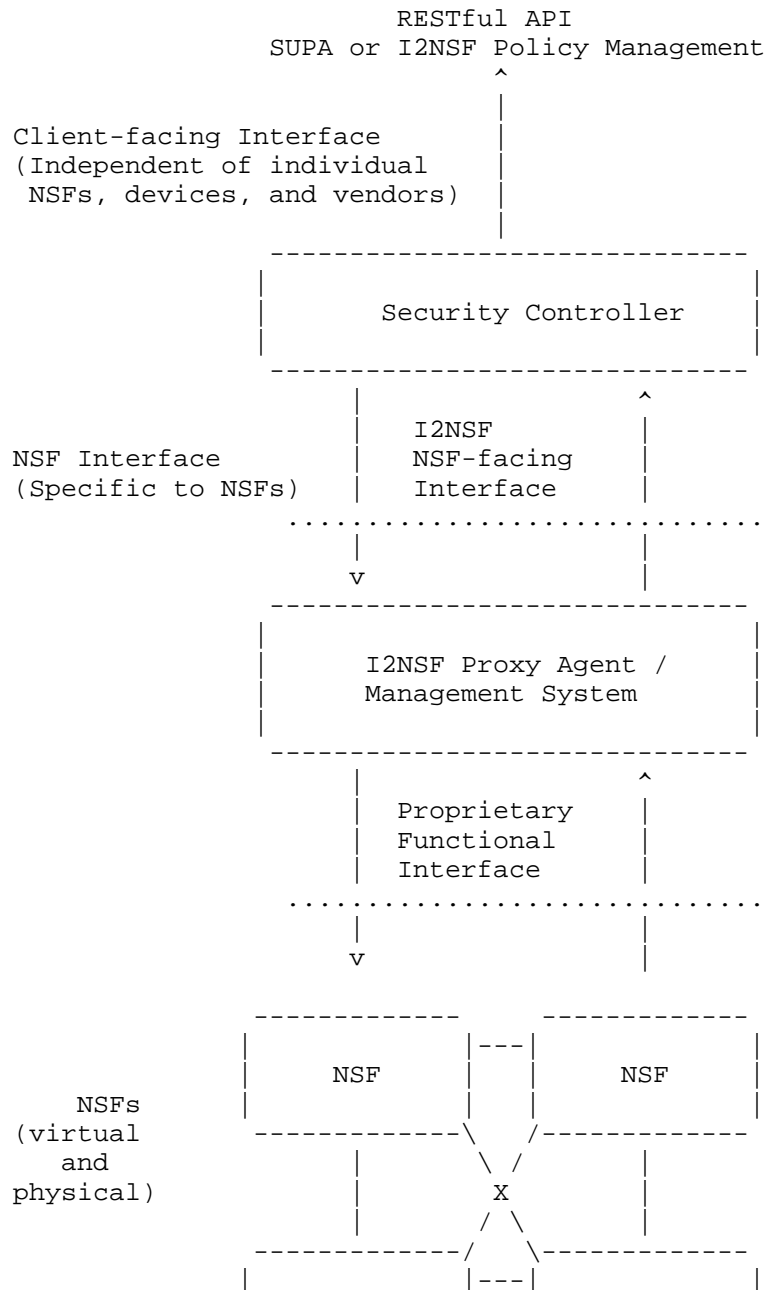




Figure 2: Deployment with Management System or I2NSF Proxy Agent

As mentioned above, these models discussed here don't affect the definition of Client-Facing interface, they do give an overall context for defining a Security Policy interface based on abstraction. This can help in implementing a Security Controller.

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 Security Admin'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 their business needs and objectives.

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:

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

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

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

A specific implementation may not support all these requirements but in order to define a base set of requirements which would work for most use-cases, this document will make an attempt to classify these requirements in three categories:

MUST: This means, the requirement must be supported by Client-Facing interface.

RECOMMENDED: This means, we recommend that Client-Facing interface support this requirement since it might be applicable to large number of use-cases but some vendor may choose to omit if their focus is only certain market segments.

MAY: This means, the requirement is not mandatory for Client-Facing interface but may be needed for specific use-cases.

4.1. Requirement for Multi-Tenancy in Client-Facing interface

An organization may have internal tenants and might want a framework wherein each tenant manages its own Security Policies with isolation from other tenants. This requirement may be applicable to Service Providers and Large Enterprises so we classify this requirement in RECOMMENDED category. If an implement does not support this requirement, it must support a default implicit tenant created by Security Controller that owns all the Security Policies.

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 able to manage its own Security Policies without affecting other tenants.

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.

The following objects are needed to fulfill this requirement:

Policy-Tenant: An entity that owns and manages Security Policies applied to its own asset and 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-Facing interface

A Security Admin must be authenticated and authorized in order to manage Security Policies. We classify this requirement in MUST category since without proper authentication and authorization, the security posture of entire organization can be easily compromised.

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 [RFC6749], 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 to 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-Facing interface

A tenant in organization may have multiple users with each user given certain privileges. Some user such as "Admin" may have all the permission but other may have limited permissions. We classify this requirement in RECOMMENDED category since it aligns with Multi-Tenancy requirement. If this requirement is not supported, a default privilege must be assigned to all the users.

The following objects are needed to fulfill this requirement:

Policy-Authorization-Role: Defines the permissions assigned to a user such as creating and managing policies on specified resources. A user may not be allowed to change existing policies but only view them.

4.4. Requirement to protect Client-Facing interface from attacks

The interface must be protected against attacks from malicious clients or a client impersonator. Potential attacks could come from Botnets, hosts infected with virus or some unauthorized entities. This requirement is highly RECOMMENDED since it may not be needed if the entire framework is deployed in a very controlled environment. But if needed, we recommend that Security Controller uses an out-of-band communication channel for Client-Facing interface. In addition, it is also recommended that traffic on Client-Facing interface communication 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-Facing interface from misconfiguration

There must be protections from mis-configured clients. System and policy parameters 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. We consider this to be a MUST requirement but implementation aspects would depend upon each individual API communication.

4.6. Requirement to manage policy lifecycle with rich set of controls

In order to provide a more sophisticated security framework, there should be a mechanism so that a policy becomes dynamically active/enforced or inactive based on multiple different criteria such as Security Admin's manual intervention or some external event. We consider the requirement listed here to be a MUST for a wide variety of use-cases.

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 threat detection. 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 activated by Security Admin manually using a client interface such as GUI or CLI.
- 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 configured but gets activated or deactivated upon specified timing calendar with Security Policy definition.

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 Security Admin 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". This requirement is the fundamental

building block of Client-Facing interface; so making it a MUST requirement. But object defined here may not support all use-cases and may not be required by everyone so it is left up to vendor whether all or partial set of these object is supported.

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 "Metadata Driven Groups". The metadata is a tag associated with endpoint information such as User, Application, or Device. The mapping from metadata 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 endpoint.

Client-Facing interface must support Endpoint Groups as a target for a Security Policy. The following metadata driven groups MAY be used for configuring Security Policies:

User-Group: This group identifies a set of users based on a tag or static information such as user-names. The tag identifying users, 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 device information. The tag identifying the devices, 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 application names. The tag identifying applications, 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 locations. Tag may correspond 1:1 to location. The tag identifying locations 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 Policy Rules is a central component of any Security Policy but rule requirements may vary based on use-cases and it is hard to define a complete set that works for everyone. In order to build a rich interface, we are going to take a different approach; we will define the building block of rules and let Security Admin build rules using these construct so that Security Policies meet their requirements:

Segmentation policies : This set of policies create rules for communication between two Endpoint Groups. An organization may restrict certain communication between a set of user and applications for example. The segmentation policy may be a micro-segmentation rule between components of complex applications or related to hybrid cloud deployment based on location.

Threat policies: This set of policies creates rules to prevent communication with externally or internally identified threats. The threats may be well knows such as threat feeds from external sources or dynamically identified by using specialty devices in the network.

Governance and Compliance policies: This set of policies creates rules to implement business requirement such as controlling access to internal or external resources for meeting regulatory compliance or business objectives.

In order to build a generic rule engine to satisfy diverse set of Policy Rules, we propose following objects:

In order to build a generic rule engine to satisfy diverse set of Policy Rules, we propose following objects:

Source Policy Endpoint Group: A source target of the Policy Rule. This may be special object "ALL" if all groups meet this criteria.

Destination Policy Endpoint Group: A destination target of the Policy Rule. This may be a special object "ALL", if all groups meet this criteria.

Direction: By default rules are applied in either direction but this object can be used to make rule definition uni-directional.

Threat Group: An object that represents a set of static or dynamic threats such as Botnet, GeoIP, URL feeds or virus and malware signatures detected dynamically. This object can be used as source or destination target in a rule.

Match Condition: An object that represents a set of allowed interactions. It could be as simple as group of application names or L4 ports allowed between two Endpoint Groups. It could very well that all traffic is allowed between two groups.

Exceptions: In order to truly build rules which are Security Admin and built with user semantics, we should allow to specify exceptions to the match criteria. This will greatly simplify Security Admin's task. E.g., we could build a rule that allows all traffic between two groups except a particular application or threat source.

Actions: Action is what makes rule and Policy work. The Action is defined in details in next section. We RECOMMEND that there be a one-to-one mapping between rule and action otherwise if multiple rules are associated with one action, it may be a difficult to manage Security Policy lifecycle as they evolve.

4.9. Requirement to express rich set of Policy Actions

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

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 is useful for threat remediation purposes. If a security breach or infection point is detected, a Security Admin would like to isolate for purpose of remediation or controlling attack surface.

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, IPS, IDS, honeypot, or VPN, etc.

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 for consistent policy enforcement

As proposed in this document that the Client-Facing interface MUST be built using higher-level "User-Constructs" that are independent of network design and implementations. In order to achieve this, it becomes important that Security Controller functionality becomes more complex that keep track of various objects that are used to express Security Policies. The Security Controller MUST evaluate the Security Policies whenever these objects and network topology change to make sure that Security Policy is consistently enforced as expressed.

Although this document does not specify how Security Controller achieve this and any implementation challenges. It is assumed that once Security Controller uses Client-Facing interface to accept Security Policies; it would maintain the security posture as per the Security Policies during all changes in network or Endpoints and other building blocks of the framework.

An event must be logged by Security Controller when a Security Policy is updated due to changes in it's building blocks such as Endpoint Group contents or the Security Policy is moved from one enforcement point to another because the Endpoint has moved in the network. This may help in debugging and auditing for compliance reasons. The Security Admin may optionally receive notifications if supported and desired.

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 a newly expressed Security Policy could conflict with existing Security Policies applied to a set of Policy Endpoint Groups. This MUST be detected and Security Admin be allowed for manual correction if needed.

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 framework in a network may require the use of a specialty device such as honeypot, behavioral analytic, or SIEM for threat detection; the device may provide threat information such as threat feeds, virus signatures, and malicious file hashes.

The Client-Facing interface must allow Security Admin to include these devices under Security Controller's Client-Facing interface so that a Security Policy could be expressed using information from such devices; basically it allows ability to integrate third part devices into the Security Policy framework.

4.14. Requirement to collect telemetry data

One of the most important aspect of security is to have visibility into the network. As threats become more sophisticated, Security Admin must be able to gather different types of telemetry data from various NSFs 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.

Client-Facing interface must provide a set of telemetry data available to Security Admin from Security Controller. The Security Admin should be able to subscribe and receive to this data set.

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 important since Security Controller could be deployed by using multiple componenets and different pieces may come from different vendors; it is difficult to isolate and debug issues without ablility to track each component's operational behavior. Even if the vendor is same for all the components, it is hard to imagine that all pieces would be released in lock step by the vendor.

Without API versioning, it is hard to debug and figure out issues when deploying Security Controller and its components built overtime across multiple release cycles. Although API versioning does not guarantee that Security Controller would always work but it helps in debugging if the problem is caused by an API mismatch.

5.2. API Extensibility

Abstraction and standardization of Client-Facing interface is of tremendous value to Security Admins 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 SHALL be a way to extend existing APIs or to create a new API that addresses specific vendors's new NSF functionality.

5.3. APIs and Data Model Transport

The APIs for interface SHALL be derived from the YANG based data model. The data model for Client-Facing 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 and Monitoring

Client-Facing interface must allow ability to collect various alarms, events, statistics about enforcement and policy violations from NSFs in the network. The events and alarms may be associated with a specific policy or associated with operating conditions of a specific NSF in general. The statistics may be a measure of potential Security Policy violations or general data that reflect operational behavior of a NSF. The events, alarms and statistics may also be used as an input to automate Security Policy lifecycle management.

5.5. Affinity

Client-Facing interface must allow Security Admin to pass any additional metadata that a user may want to provide with a Security Policy e.g., if the policy needs to be enforced by a very highly secure NSF with Trusted Platform Module (TPM) chip. Another example would be, if a policy can not be enforced by a multi-tenant NSF. This would Security Admin control on operating environment

5.6. Test Interface

Client-Facing interface must support 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 existing policies or

if there are enough resources and capability to enforce this policy. The test interface would provide a mechanism to Security Admin where policies could be tested in the actual environment before enforcement.

6. Security Considerations

Client-Facing interface to Security controller must be protected to make sure that entire security posture is not compromised. This draft mandates that interface must have proper authentication and authorization control mechanisms to ward off malicious attacks. The draft does not specify a particular mechanism as different organization may have different needs based on their specific deployment environment and moreover new methods may evolve to better suit contemporary requirements.

Authentication and authorization alone may not be sufficient for Client-Facing interface; the interface API must be validated for proper input 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.

We recommend that all attack surface must be examined with careful consideration of the operating environment and available industry best practices must be used such as process and standards to protect security controller against malicious or inadvertent attacks.

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-framework]

Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-06 (work in progress), July 2017.

[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-16 (work in progress), May 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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

J. Jeong
S. Hyun
Sungkyunkwan University
T. Ahn
Korea Telecom
S. Hares
Huawei
D. Lopez
Telefonica I+D
July 3, 2017

Applicability of Interfaces to Network Security Functions to Networked
Security Services
draft-jeong-i2nsf-applicability-00

Abstract

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Terminology	3
4. I2NSF Framework	4
5. Use Cases	6
5.1. Firewall: Centralized Firewall System	6
5.2. Deep Packet Inspection: Centralized VoIP/VoLTE Security System	7
5.3. Attack Mitigation: Centralized DDoS-attack Mitigation System	9
6. Security Considerations	11
7. Acknowledgements	11
8. References	11
8.1. Normative References	11
8.2. Informative References	12

1. Introduction

Interface to Network Security Functions (I2NSF) proposes a standard framework and standard interfaces for networked security services in Network Functions Virtualization (NFV) environments. The I2NSF enables multiple security-vendor products to be used cost-effectively in the NFV environment by utilizing the capabilities of such products and the virtualization of security functions in the NFV platform.

This document describes the applicability of I2NSF to networked security services with use cases, such as firewall, Deep Packet Inspection (DPI), and Distributed Denial of Service (DDoS) attack mitigation.

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], [ITU-T.X.800], [i2nsf-framework], [consumer-facing-inf-im], [consumer-facing-inf-dm], [i2nsf-nsf-cap-im], [nsf-facing-inf-dm], [registration-inf-im], [registration-inf-dm], and [nsf-triggered-steering]. In addition, the following terms are defined below:

- o 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].
- o 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.
- o 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 an SDN switch and a firewall function as a virtual network function (VNF).

- o 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 an SDN switch and a VoIP/VoLTE security function as a VNF.
- o Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation. 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 an SDN switch and a DDoS-attack mitigation function as a VNF.

4. I2NSF Framework

This section describes an extended I2NSF framework with SDN for I2NSF applicability and use cases, such as firewall system, deep packet inspection system, and DDoS-attack mitigation system.

Figure 1 shows an I2NSF framework with SDN networks to support networked security services [i2nsf-framework]. As shown in Figure 1, I2NSF User can use security services by delivering their high-level security policies to Security Controller via Consumer-Facing Interface [consumer-facing-inf-im][consumer-facing-inf-dm].

Security Controller can translate the high-level security policies (received from I2NSF User via Consumer-Facing Interface) into low-level security policies for the corresponding NSFs. These low-level security policies are sent to NSFs via NSF-Facing Interface [i2nsf-nsf-cap-im][nsf-facing-inf-dm].

Security Controller asks NSFs to perform low-level security services via NSF-Facing Interface. The NSFs run as Virtual Network Functions (VNFs) 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 (i.e., SDN Controller). In addition, Security Controller uses Registration Interface [registration-inf-im][registration-inf-dm] to communicate with Developer's Management Aystem for registering (or deregistering) the developer's NSFs into (or from) the NFV system using the I2NSF framework.

Consumer-Facing Interface between I2NSF User and Security Controller can be implemented by RESTCONF [RFC8040], which is a protocol based on HTTP for configuring data defined in YANG [RFC6020], using the datastore concepts defined in Network Configuration Protocol

(NETCONF) [RFC6241]. YANG data models can describe high-level security services for the sake of I2NSF User. A data model in [consumer-facing-inf-dm] can be used for the I2NSF Consumer-Facing Interface.

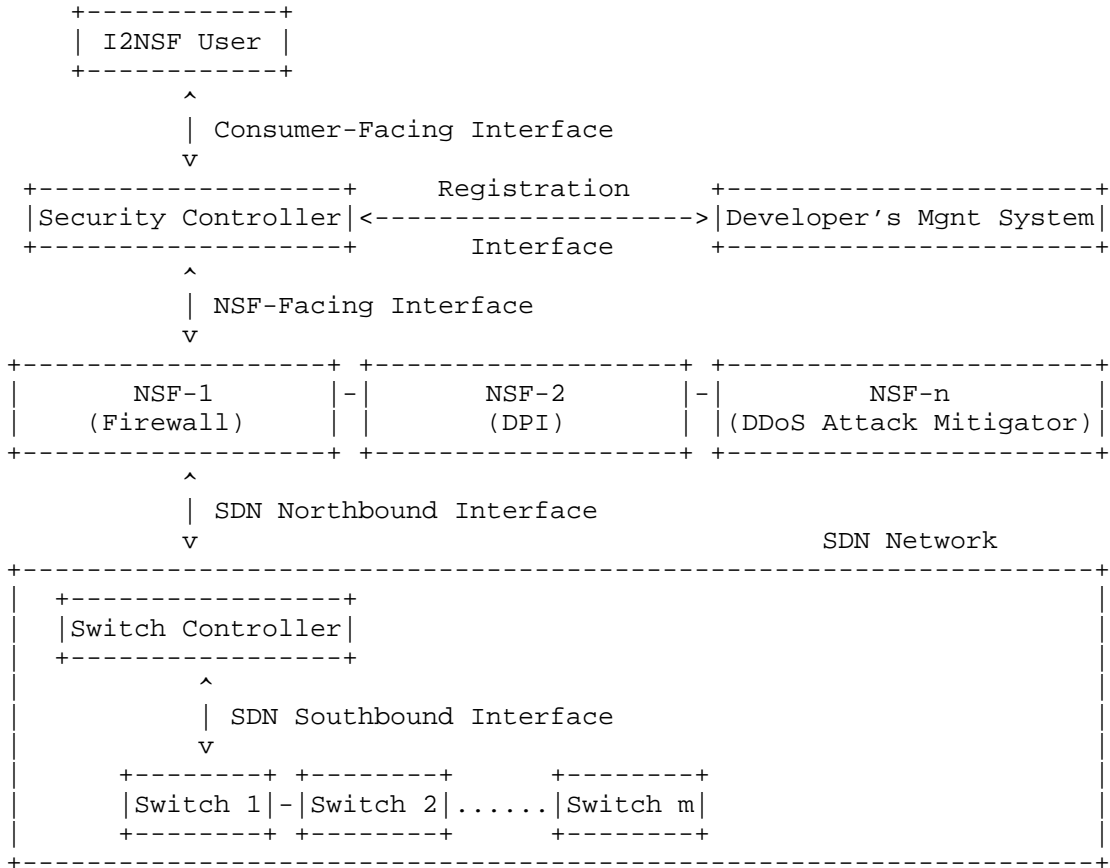


Figure 1: An I2NSF Framework with SDN Networks

NSF-Facing Interface between Security Controller and NSFs can be implemented by NETCONF [RFC6241] for configuring data defined in YANG [RFC6020]. YANG data models can describe low-level security services for the sake of NSFs. A data model in [nsf-facing-inf-dm] can be used for the I2NSF NSF-Facing Interface.

Registration Interface between Security Controller and Developer's Management System can be implemented by RESTCONF [RFC8040] for configuring data defined in YANG [RFC6020]. YANG data models can describe the NSF capabilities of networked security services. A data

model in [registration-inf-dm] can be used for the I2NSF Registration Interface.

Also, the I2NSF framework can enforce multiple chained NSFs for the low-level security policies with a service function chaining (SFC) for the I2NSF architecture in [nsf-triggered-steering].

5. Use Cases

This section introduces three use cases for cloud-based security services: (i) firewall system, (ii) deep packet inspection system, and (iii) attack mitigation system.

5.1. Firewall: 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 the challenges through the above centralized firewall system based on SDN as follows:

- o 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.
- o 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.
- o 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.
- o 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.
- o 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.

5.2. Deep Packet Inspection: 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 the challenges through the above centralized VoIP/VoLTE security system based on SDN as follows:

- o 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.
- o 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.

- o 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.
- o 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.

5.3. Attack Mitigation: 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 management 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 the challenges through the above centralized DDoS-attack mitigation system based on SDN as follows:

- o 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.
- o 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.
- o 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.
- o 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.

So far this document has described the procedure and impact of the three use cases for networked security services using the I2NSF framework with SDN networks. To support these use cases in the proposed data-driven security service framework, YANG data models

described in [consumer-facing-inf-dm], [nsf-facing-inf-dm], and [registration-inf-dm] can be used as Consumer-Facing Interface, NSF-Facing Interface, and Registration Interface, respectively, along with RESTCONF [RFC8040] and NETCONF [RFC6241].

6. Security Considerations

The I2NSF framework with SDN networks 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].

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 Hyoungshick Kim, Jung-Soo Park, Se-Hui Lee, Jinyong Kim, Daeyoung Hyun, and Dongjin Hong.

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. |
| [i2nsf-framework] | Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 2017. |
| [RFC6020] | Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010. |

- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

8.2. Informative References

- [consumer-facing-inf-im] Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislaamovic, S., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-kumar-i2nsf-client-facing-interface-im-02 (work in progress), April 2017.
- [consumer-facing-inf-dm] Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", draft-jeong-i2nsf-consumer-facing-interface-dm-02 (work in progress), July 2017.
- [i2nsf-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01 (work in progress), March 2017.
- [nsf-facing-inf-dm] Kim, J., Jeong, J., Park, J., Hares, S., and L. Xia, "I2NSF Network Security Functions-Facing Interface YANG Data Model", draft-kim-i2nsf-nsf-facing-interface-data-model-02, July 2017.
- [registration-inf-im] Hyun, S., Jeong, J., Woo, S., Yeo, Y., and J. Park, "I2NSF Registration Interface Information Model", draft-hyun-i2nsf-registration-interface-im-02 (work in progress), July 2017.
- [registration-inf-dm] Hyun, S., Jeong, J., Yeo, Y., Woo, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-hyun-i2nsf-registration-dm-01 (work in progress), July 2017.
- [nsf-triggered-steering] Hyun, S., Jeong, J., Park, J., and S.

- Hares, "NSF-triggered Traffic Steering Framework",
draft-hyun-i2nsf-nsf-triggered-steering-03
(work in progress), July 2017.
- [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.

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>

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/>

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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

J. Jeong
E. Kim
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei
July 3, 2017

I2NSF Consumer-Facing Interface YANG Data Model
draft-jeong-i2nsf-consumer-facing-interface-dm-02

Abstract

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
4. Data Modeling for Consumer-Facing Interface	3
5. YANG Data Model for Consumer-Facing Interface	7
6. Security Considerations	34
7. Acknowledgements	34
8. References	35
8.1. Normative References	35
8.2. Informative References	35
Appendix A. Changes from draft-jeong-i2nsf-consumer-facing-interface-dm-01	36
Appendix B. Use Case: Policy Instance Example for VoIP/VoLTE Security Services	36
Appendix C. Policy Instance YANG Example for VoIP/VoLTE Security Services	38
Authors' Addresses	44

1. Introduction

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

Data models are defined at a lower level of abstraction and provide many details. They provide details about the implementation of a protocol's specification, e.g., rules that explain how to map managed objects onto lower-level protocol constructs. Since conceptual

models can be implemented in different ways, multiple data models can be derived by a single information model.

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

2. Requirements Language

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

3. Terminology

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

4. Data Modeling for Consumer-Facing Interface

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

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

This data model is designed to support the I2NSF framework that can be extended according to the security needs. In other words, the model design is independent of the content and meaning of specific policies as well as the implementation approach. This document suggests a VoIP/VoLTE security service as a use case for policy rule generation.

```
module: ietf-i2nsf-consumer-facing-interface
  +-rw ietf-i2nsf-consumer-facing-interface
```



```

+--rw multi-tenancy
|   +--rw policy-domain* [policy-domain-id]
|   |   +--rw policy-domain-id          uint16
|   |   +--rw name                      string
|   |   +--rw address                   string
|   |   +--rw contact                   string
|   |   +--rw date                      yang:date-and-time
|   |   +--rw authentication-method     string
|   +--rw policy-tenant* [policy-tenant-id]
|   |   +--rw policy-tenant-id          uint16
|   |   +--rw name                      string
|   |   +--rw date                      yang:date-and-time
|   |   +--rw domain                   string
|   +--rw policy-role* [policy-role-id]
|   |   +--rw policy-role-id            uint16
|   |   +--rw name                      string
|   |   +--rw date                      yang:date-and-time
|   |   +--rw access-profile            string
|   +--rw policy-user* [policy-user-id]
|   |   +--rw policy-user-id            uint16
|   |   +--rw name                      string
|   |   +--rw date                      yang:date-and-time
|   |   +--rw password                  string
|   |   +--rw email                     string
|   |   +--rw scope-type?               string
|   |   +--rw scope-reference?          string
|   |   +--rw role                      string
|   +--rw policy-mgmt-auth-method* [policy-mgmt-auth-method-id]
|   |   +--rw policy-mgmt-auth-method-id uint16
|   |   +--rw name                      string
|   |   +--rw date                      yang:date-and-time
|   |   +--rw authentication-method     string
|   |   +--rw mutual-authentication     boolean
|   |   +--rw token-server              string
|   |   +--rw certificate-server         string
|   |   +--rw single-sing-on-server     string
|   +--rw policy-endpoint-groups
|   |   +--rw meta-data-source* [meta-data-source-id]
|   |   |   +--rw meta-data-source-id    uint16
|   |   |   +--rw name                  string
|   |   |   +--rw date                  yang:date-and-time
|   |   |   +--rw tag-type?              boolean
|   |   |   +--rw tag-server-information? string
|   |   |   +--rw tag-application-protocol? string
|   |   |   +--rw tag-server-credential? string
|   |   +--rw user-group* [user-group-id]
|   |   |   +--rw user-group-id          uint16
|   |   |   +--rw name?                  string

```

```

+--rw date? yang:date-and-time
+--rw group-type? string
+--rw meta-data-server? string
+--rw group-member? string
+--rw risk-level? uint16
+--rw device-group* [device-group-id]
+--rw device-group-id uint16
+--rw name? string
+--rw date? yang:date-and-time
+--rw group-type? string
+--rw meta-data-server? string
+--rw group-member? string
+--rw risk-level? uint16
+--rw application-group* [application-group-id]
+--rw application-group-id uint16
+--rw name? string
+--rw date? yang:date-and-time
+--rw group-type? string
+--rw meta-data-server? string
+--rw group-member? string
+--rw risk-level? uint16
+--rw location-group* [location-group-id]
+--rw location-group-id uint16
+--rw name? string
+--rw date? yang:date-and-time
+--rw group-type? string
+--rw meta-data-server? string
+--rw group-member? string
+--rw risk-level? uint16
+--rw threat-prevention
+--rw threat-feed* [threat-feed-id]
+--rw threat-feed-id uint16
+--rw name? string
+--rw date? yang:date-and-time
+--rw feed-type? enumeration
+--rw feed-server? string
+--rw feed-priority? uint16
+--rw custom-list* [custom-list-id]
+--rw custom-list-id uint16
+--rw name? string
+--rw date? yang:date-and-time
+--rw list-type? enumeration
+--rw list-property? enumeration
+--rw list-content? string
+--rw malware-scan-group* [malware-scan-group-id]
+--rw malware-scan-group-id uint16
+--rw name? string
+--rw date? yang:date-and-time

```

```

| | | +--rw signature-server? string
| | | +--rw file-types? string
| | | +--rw malware-signatures? string
+--rw event-map-group* [event-map-group-id]
| | | +--rw event-map-group-id uint16
| | | +--rw name? string
| | | +--rw date? yang:date-and-time
| | | +--rw security-events? string
| | | +--rw threat-map? string
+--rw telemetry-data
| | | +--rw telemetry-data* [telemetry-data-id]
| | | | | +--rw telemetry-data-id uint16
| | | | | +--rw name? string
| | | | | +--rw date? yang:date-and-time
| | | | | +--rw logs? boolean
| | | | | +--rw syslogs? boolean
| | | | | +--rw snmp? boolean
| | | | | +--rw sflow? boolean
| | | | | +--rw netflow? boolean
| | | | | +--rw interface-stats? boolean
+--rw telemetry-source* [telemetry-source-id]
| | | +--rw telemetry-source-id uint16
| | | +--rw name? string
| | | +--rw date? yang:date-and-time
| | | +--rw source-type? string
| | | +--rw nsf-access-parameters? string
| | | +--rw nsf-access-credentials? string
| | | +--rw collection-interval? uint16
| | | +--rw collection-method? enumeration
| | | +--rw heartbeat-interval? uint16
| | | +--rw qos-marking? uint8
+--rw telemetry-destination* [telemetry-destination-id]
| | | +--rw telemetry-destination-id uint16
| | | +--rw name? string
| | | +--rw date? yang:date-and-time
| | | +--rw collector-state? string
| | | +--rw collector-access-parameters? string
| | | +--rw collector-access-credentials? string
| | | +--rw data-encoding? string
| | | +--rw data-transport? string
+--rw policy-instance
| | | +--rw policy-calendar* [policy-calendar-id]
| | | | | +--rw policy-calendar-id uint16
| | | | | +--rw name? string
| | | | | +--rw date? yang:date-and-time
| | | | | +--rw enforcement-type? enumeration
| | | | | +--rw time-information? string
| | | | | +--rw event-map? string

```

```

+--rw policy-action* [policy-action-id]
|   +--rw policy-action-id          string
|   +--rw name?                     string
|   +--rw date?                     yang:date-and-time
|   +--rw primary-action?           string
|   +--rw secondary-action?         string
+--rw policy-rule* [policy-rule-id]
|   +--rw policy-rule-id            string
|   +--rw name?                     string
|   +--rw date?                     yang:date-and-time
|   +--rw source?                   string
|   +--rw destination?              string
|   +--rw exception?                string
|   +--rw action?                   string
|   +--rw precedence?               uint8
+--rw policy-instance* [policy-instance-id]
|   +--rw policy-instance-id        string
|   +--rw name?                     string
|   +--rw date?                     yang:date-and-time
|   +--rw rules?                    string
|   +--rw scheduling-type?           enumeration
|   +--rw scheduling-information?    string
|   +--rw owner?                    string

```

Figure 1: Generic Data Model for Consumer-Facing Interface

5. YANG Data Model for Consumer-Facing Interface

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

```

<CODE BEGINS> file "ietf-i2nsf-consumer-facing-interface.yang"
module ietf-i2nsf-consumer-facing-interface {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-consumer-facing-interface";
  prefix
    capability-interface;

  import ietf-yang-types {
    prefix 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: Jaehoon Paul Jeong
  <mailto:pauljeong@skku.edu>";

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

revision "2017-07-03" {
  description "Initial revision";
  reference
    "draft-kumar-i2nsf-client-facing-interface-im-02";
}

//Groupings
container ietf-i2nsf-consumer-facing-interface {
  description
    " ";
  container multi-tenancy {
    description
      "The descriptions of multi-tenancy.";

    list policy-domain {
      key "policy-domain-id";
      leaf policy-domain-id {
        type uint16;
        mandatory true;
        description
          "This represents the list of domains.";
      }
    }
    description
      "this represent the list of policy domains";
    leaf name {
      type string;
      mandatory true;
    }
  }
}
```

```
        description
            "Name of the organization or customer representing
            this domain.";
    }

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

    leaf contact {
        type string;
        mandatory true;
        description
            "contact information of the organization
            or customer.";
    }

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

    leaf authentication-method {
        type string;
        mandatory true;
        description
            "The description of authentication method;
            token-based, password, certificate,
            single-sign-on";
    }
}

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

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

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

leaf domain {
    type string;
    mandatory true;
    description
        "This field identifies the domain to which this
        tenant belongs. This should be reference to a
        'Policy-Domain' object.";
}
}

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

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

```
    }

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

list policy-user {
    key "policy-user-id";
    leaf policy-user-id {
        type uint16;
        description
            "This represents the policy-user-id.";
    }
    description
        "This represents the list of policy users.";
    leaf name {
        type string;
        mandatory true;
        description
            "The name of a user.";
    }

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

    leaf password {
        type string;
        mandatory true;
        description
            "User password for basic authentication";
    }

    leaf email {
        type string;
        mandatory true;
        description
            "The email account of a user";
    }
}
```



```
    }

    leaf scope-type {
        type string;
        description
            "identifies whether a user has domain-wide
            or tenant-wide privileges";
    }

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

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

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

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

```
    }

    leaf authentication-method {
        type string;
        mandatory true;
        description
            "The description of authentication method;
            token-based, password, certificate,
            single-sign-on";
    }

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

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

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

    leaf single-sign-on-server {
        type string;
        mandatory true;
        description
            "The single-sign-on-server information
            if the authentication method is
            single-sign-on-based";
    }
}

container policy-endpoint-groups {
    description
```

"A logical entity in their business environment, where a security policy is to be applied.";

```
list meta-data-source {
  key "meta-data-source-id";
  leaf meta-data-source-id {
    type uint16;
    mandatory true;
    description
      "This represents the meta-data source id.";
  }
  description
    "This represents the meta-data source.";
  leaf name {
    type string;
    mandatory true;
    description
      "This identifies the name of the
        meta-datas-ource.";
  }
  leaf date {
    type yang:date-and-time;
    mandatory true;
    description
      "This identifies the date this object was
        created or last modified.";
  }

  leaf tag-type {
    type boolean;
    description
      "This identifies the group type; user group,
        app group or device group.";
  }

  leaf tag-server-information {
    type string;
    description
      "The description of suthentication method;
        token-based, password, certificate,
        single-sign-on";
  }
  leaf tag-application-protocol {
    type string;
    description
      "This filed identifies the protocol e.g. LDAP,
        Active Directory, or CMDB";
  }
}
```

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

list user-group{
    key "user-group-id";
    leaf user-group-id {
        type uint16;
        mandatory true;
        description
            "This represents the the user group id.";
    }
    description
        "This represents the user group.";
    leaf name {
        type string;
        description
            "This field identifies the name of user-group.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "when this user-group was created or last modified.";
    }
    leaf group-type {
        type string;
        description
            "This describes the group type; User-tag,
            User-name or IP-address.";
    }
    leaf meta-data-server {
        type string;
        description
            "This references metadata source";
    }
    leaf group-member {
        type string;
        description
            "This describes the user-tag information";
    }
}
```

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

list device-group{
    key "device-group-id";
    leaf device-group-id {
        type uint16;
        description
            "This represents a device group id.";
    }
    description
        "This represents a device group.";
    leaf name {
        type string;
        description
            "This field identifies the name of
             a device-group.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "The date when this group was create or
             last modified.";
    }

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

    leaf meta-data-server {
        type string;
        description
            "This references meta-data-source
             object.";
    }

    leaf group-member {
        type string;
        description
            "This describes the device-tag, device-name or
```

```
        IP-address information";
    }

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

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

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

    leaf group-type {
        type string;
        description
            "This identifies the group type;
            application-tag, application-name or
            IP-address.";
    }

    leaf meta-data-server {
        type string;
        description
            "This references meta-data-source
            object.";
    }
}
```

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

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

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

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

    leaf group-type {
        type string;
        description
            "This identifies the group type;
            location-tag, location-name or
            IP-address.";
    }

    leaf meta-data-server {
```

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

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

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

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

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

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



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

leaf feed-server {
  type string;
  description
    "this contains threat feed server information.";
}

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

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

```
        "Name of the custom-list.";
    }

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

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

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

```
        or whitelist.";
    }

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

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

    leaf signature-server {
        type string;
        description
            "This describes the signature server of the
            malware-scan-group.";
    }

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

    leaf malware-signatures {
        type string;
        description
```

```
        "This contains a list of malware signatures or hash.";
    }
}

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

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

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

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

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

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

```
list telemetry-data {
  key "telemetry-data-id";
  leaf telemetry-data-id {
    type uint16;
    mandatory true;
    description
      "This is ID for telemetry-data-id.";
  }
  description
    "This is ID for telemetry-data.";
  leaf name {
    type string;
    description
      "Name of the telemetry-data object.";
  }

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

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

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

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

  leaf sflow {
    type boolean;
    description
      "This field identifies whether 'sFlow' data
```

```
        need to be collected.";
    }

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

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

list telemetry-source {
    key "telemetry-source-id";
    leaf telemetry-source-id {
        type uint16;
        mandatory true;
        description
            "This is ID for telemetry-source-id.";
    }
    description
        "This is ID for telemetry-source.";
    leaf name {
        type string;
        description
            "This identifies the name of this object.";
    }
}

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

leaf source-type {
    type string;
    description
        "This should have one of the following type of
        the NSF telemetry source: NETWORK-NSF,
        FIREWALL-NSF, IDS-NSF, IPS-NSF,
        PROXY-NSF, VPN-NSF, DNS, ACTIVE-DIRECTORY,
```

```
        IP Reputation Authority, Web Reputation
        Authority, Anti-Malware Sandbox, Honey Pot,
        DHCP, Other Third Party, ENDPOINT";
    }

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

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

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

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

```

```
        "This field contains a method of collection,
        i.e., whether it is PUSH-based or PULL-based.";
    }

    leaf heartbeat-interval {
        type uint16;
        units seconds;
        description
            "time in seconds the source sends telemetry
            heartbeat.";
    }

    leaf qos-marking {
        type uint8;
        description
            "DSCP value must be contained in this field.";
    }
}
list telemetry-destination {
    key "telemetry-destination-id";
    leaf telemetry-destination-id {
        type uint16;
        description
            "this represents the telemetry-destination-id";
    }
    description
        "This object contains information related to
        telemetry destination. The destination is
        usually a collector which is either a part of
        Security Controller or external system
        such as Security Information and Event
        Management (SIEM).";

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

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

    leaf collector-state {
        type string;
```



```
        description
            "This describes collector state information.";
    }
    leaf collector-access-parameters {
        type string;
        description
            "ip address and port number of the nsf
            providing telemetry data.";
    }

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

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

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

    leaf data-transport {
        type string;
        description
            "This field contains streaming telemetry data
            protocols. This could be gRPC, protocol
            buffer over UDP, etc.";
    }
}

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

```
list policy-calendar {
  key "policy-calendar-id";
  leaf policy-calendar-id {
    type uint16;
    description
      "this represents the policy-calendar-id.";
  }
  description
    "This object contains information related to
    scheduling a policy. The policy could be
    activated based on a time calendar or security
    event including threat level changes.";

  leaf name {
    type string;
    description
      "Name of the policy-calendar object.";
  }

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

  leaf enforcement-type {
    type enumeration {
      enum unknown {
        description
          "enforcement-type is unknown.";
      }
      enum admin-enforced {
        description
          "enforcement-type is ADMIN-ENFORCED.";
      }
      enum time-enforced {
        description
          "enforcement-type is TIME-ENFORCED.";
      }
      enum event-enforced {
        description
          "enforcement-type is EVENT-ENFORCED.";
      }
    }
    description
      "This field identifies whether the policy
      enforcement is 'ADMIN-ENFORCED' or
```

```
        'TIME-ENFORCED', or 'EVENT-ENFORCED'.";
    }

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

    leaf event-map {
        type string;
        description
            "This field contains security events and
            threat map in order to determine when a
            policy need to be activated.";
    }
}

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

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

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

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

    leaf secondary-action {
        type string;
        description
            "This field identifies additional actions if
            a rule is matched. This could be one of 'LOG',
            'SYSLOG', 'SESSION-LOG', etc.";
    }
}

list policy-rule {
    key "policy-rule-id";
    leaf policy-rule-id {
        type string;
        mandatory true;
        description
            "this represents the policy-rule-id";
    }
    description
        "This object represents rules that a
        Security Admin want to define in order
        to express its business objectives in
        a Security Policy.";
    leaf name {
        type string;
        description
            "This field identifies the name of
            this object.";
    }

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

    leaf source {
        type string;
        description
            "This field identifies the source of
            the traffic. This could be reference to
```

```
        either 'Policy Endpoint Group' or
        'Threat-Feed' or 'Custom-List' if Security
        Admin wants to specify the source; otherwise,
        the default is to match all traffic.";
    }

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

    leaf exception {
        type string;
        description
            "This field identifies the exception
            consideration when 'Source' and
            'Destination' are matched for a given
            communication. This should be reference
            to 'Policy Endpoint Group' object.";
    }

    leaf action {
        type string;
        description
            "This field identifies the action taken
            when 'Source' and 'Destination' are matched
            for a given communication.";
    }

    leaf precedence {
        type uint8;
        description
            "This field identifies the precedence
            assigned to this rule by Security Admin.
            This is helpful in conflict resolution
            when two or more rules match a given
            traffic class.";
    }
}

list policy-instance {
```

```
key "policy-instance-id";
leaf policy-instance-id {
  type string;
  mandatory true;
  description
    "this represents the policy-instance-id";
}
description
  "This object represents a mechanism to
  express a Security Policy by Security Admin
  to Security Controller via Consumer-Facing
  Interface. The policy would be enforced by
  an NSF.";
leaf name {
  type string;
  description
    "This field identifies the name of this
    object.";
}

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

leaf rules {
  type string;
  description
    "This field contains a list of rules.
    If the rule does not have a user-defined
    precedence, then any conflict must be
    manually resolved.";
}

leaf scheduling-type {
  type enumeration {
    enum unknown {
      description
        "scheduling-type is unknown.";
    }
    enum time-calendar {
      description
        "scheduling-type is time-calendar.";
    }
    enum event-map {
      description
```

```

    "scheduling-type is event-map.";
  }
}
description
  "This field specifies when this policy
  should be scheduled. The policy could be
  scheduled based on time calendar or
  event-map.";
}

leaf scheduling-information {
  type string;
  description
    "This field contains either the 'Calendar'
    or 'Event-map' based on 'Schedule type'.";
}

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

```

Figure 2: YANG Data Model for Consumer-Facing_interface

6. Security Considerations

The data model for the I2NSF Consumer-Facing Interface is derived from the I2NSF Consumer-Facing Interface Information Model [client-facing-inf-im], so the same security considerations with the information model should be included in this document. The data model needs to support a mechanism to protect Consumer-Facing Interface to Security Controller.

7. Acknowledgements

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

This document has greatly benefited from inputs by Hyoungshick Kim, Hoon Ko, Mahdi F. Dachmehchi, Seungjin Lee, Jinyong Tim Kim, and Daeyoung Hyun.

8. References

8.1. Normative References

- [RFC3444] Pras, A., "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

8.2. 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-05 (work in progress), May 2017.

[client-facing-inf-req]

Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislaamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-01 (work in progress), April 2017.

[client-facing-inf-im]

Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislaamovic, S., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-kumar-i2nsf-client-facing-interface-im-02 (work in progress), April 2017.

[i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Birkholz, H., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.

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

Appendix A. Changes from draft-jeong-i2nsf-consumer-facing-interface-dm-01

The following changes have been made from draft-jeong-i2nsf-consumer-facing-interface-dm-01:

- o The block diagram representing the overall architecture of security management system has been removed in this draft (Section 5 in draft-jeong-i2nsf-consumer-facing-interface-dm-01) as it is more suitable to be included in the information model draft than the data model.
- o Sections 4 and 5 have been revised to produce a data tree model and YANG data model according to the information model suggested in the draft about the I2NSF Consumer-Facing Interface Information Model [client-facing-inf-im].
- o Overall editorial errors are corrected.

Appendix B. Use Case: Policy Instance Example for VoIP/VoLTE Security Services

The following shows the example data tree model for the VoIP/VoLTE services. Multi-tenancy, endpoint groups, threat prevention, and telemetry data components are general part of the tree model, so we can just modify the policy instance in order to generate and enforce high-level policies.

The policy-calendar can act as a scheduler to set the start and end time to block calls which uses suspicious ids, or calls from other countries.

```

module ietf-i2nsf-consumer-facing-interface-policy-instance
  +--rw policy-instance
    +--rw policy-rule* [policy-rule-id]
      | +--rw policy-rule-id      uint16
      | +--rw name?              string
      | +--rw date?              yang:date-and-time
      | +--rw source?            string
      | +--rw destination?       string
      | +--rw exception?         boolean
      | +--rw exception-detail?  string
    +--rw action* [action-id]
      | +--rw action-id          string
      | +--rw name?              string
      | +--rw date?              yang:date-and-time
      | +--rw primary-action?     string
      | +--rw secondary-action?   string
    +--rw precedence* [precedence-id]
      | +--rw precedence-id      string
      | +--rw rule-exist?        boolean
    +--rw event* [event-id]
      | +--rw event-id           string
      | +--rw security-event?    string
      | +--rw threat-map?        string
      | +--rw enable?            boolean
    +--rw condition* [condition-id]
      | +--rw condition-id       string
      | +--rw caller* [caller-id]
      |   | +--rw caller-id       uint16
      |   | +--rw caller-id-id?   string
      |   | +--rw caller-country? string
      |   | +--rw caller-city?    string
      | +--rw callee* [callee-id]
      |   | +--rw callee-id       uint16
      |   | +--rw callee-id-id?   string
      |   | +--rw callee-country? string
      |   | +--rw callee-city?    string
    +--rw policy-calendar* [policy-calendar-id]
      | +--rw policy-calendar-id  uint16
      | +--rw name?              string
      | +--rw date?              yang:date-and-time
      | +--rw enforcement-type?   string
      | +--rw begin-time?         yang:date-and-time
      | +--rw end-time?           yang:date-and-time

```

Figure 3: Policy Instance Example for VoIP/VoLTE Security Services

Appendix C. Policy Instance YANG Example for VoIP/VoLTE Security Services

The following YANG data model is a policy instance for VoIP/VoLTE security services. The policy-calendar can act as a scheduler to set the start time and end time to block malicious calls which use suspicious IDs, or calls from other countries.

<CODE BEGINS> file "ietf-i2nsf-consumer-facing-inf-voip"

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

  import ietf-yang-types {
    prefix 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: Jaehoon Paul Jeong
    <mailto:pauljeong@skku.edu>";

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

  revision "2017-07-03" {
    description "Initial revision";
```

```
reference
  "draft-kumar-i2nsf-client-facing-interface-im-02";
}

//Groupings
container policy-instance {
  description
    "this describes the policy instances.";

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

    leaf policy-rule-id {
      type uint16;
      description
        "policy rule id.";
    }

    leaf name {
      type string;
      description
        "Name of the policy-rule.";
    }

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

    leaf source {
      type string;
      description
        "This references either end-point-group,
        threat-feed, or custom-list.";
    }

    leaf destination {
      type string;
      description
        "This references either end-point-group,
        threat-feed, or custom-list.";
    }

    leaf exception {
```

```
        type boolean;
        description
            "This describes whether an exception has
            occurred or not.";
    }

    leaf exception-detail{
        type string;
        description
            "This includes detailed information about
            source and destination of
            an exception.";
    }
}
list action {
    key "action-id";
    description
        "This represents the action of a policy-rule.";
    leaf action-id {
        type string;
        mandatory true;
        description
            "This represents the action-id of a policy-rule.";
    }
    leaf name {
        type string;
        description
            "The action name.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "When the action was taken.";
    }
}

leaf primary-action {
    type string;
    description
        "This includes actions such as permit,
        mirroring, rate-limit, ips, app-firewall,
        auth-session, and etc";
}

leaf secondary-action {
    type string;
    description
        "This includes optional actions such as
        logging, system logging and session logging.";
```

```
    }  
  }  
  list precedence {  
    key "precedence-id";  
    description  
      "This describes whether there is a preceeding  
       rule and causes problems.";  
    leaf precedence-id {  
      type string;  
      mandatory true;  
      description  
        "This represent the precedence-id of  
         a policy-rule.";  
    }  
    leaf rule-exist {  
      type boolean;  
      description  
        "This determines whether there is a preceeding.";  
    }  
  }  
  list event {  
    key "event-id";  
    description  
      "This represents the security event of a  
       policy-rule.";  
    leaf event-id {  
      type string;  
      mandatory true;  
      description  
        "This represents the event-id.";  
    }  
    leaf security-event {  
      type string;  
      description  
        "This references the security event in the  
         threat-prevention .";  
    }  
    leaf threat-map {  
      type string;  
      description  
        "This references the threat-map in the  
         threat-prevention.";  
    }  
    leaf enable {  
      type boolean;  
      description  
        "This determines whether the condition  
         matches the security event or not.";  
    }  
  }  
}
```

```
    }
  }
  list condition {
    key "condition-id";
    description
      "This represents the condition of a
        policy-rule.";
    leaf condition-id {
      type string;
      description
        "This represents the condition-id.";
    }
    list caller {
      key "caller-id";
      description
        "this represents the list of callers.";
      leaf caller-id {
        type uint16;
        description
          "the id of the caller.";
      }
      leaf caller-id-id {
        type string;
        description
          "The caller's number.";
      }
      leaf caller-country {
        type string;
        description
          "This determines the country of the caller.";
      }
      leaf caller-city {
        type string;
        description
          "This determines the city of the caller.";
      }
    }
  }

  list callee {
    key "callee-id";
    description
      "this represents the list of callees";
    leaf callee-id {
      type uint16;
      description
        "The id of the callee.";
    }
    leaf callee-id-id {
```

```
        type string;
        description
            "The callee's number.";
    }
    leaf callee-country {
        type string;
        description
            "This determines the country of the callee.";
    }
    leaf callee-city {
        type string;
        description
            "This determines the city of the callee.";
    }
}
list policy-calendar {
    key "policy-calendar-id";
    description
        "this represents the policy calendar list.";
    leaf policy-calendar-id {
        type uint16;
        description
            "The id of the policy calendar.";
    }
    leaf name {
        type string;
        description
            "The name of the policy-calendar.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "The date when this calender was
            created or last modified.";
    }
    leaf enforcement-type {
        type string;
        description
            "Whether the policy enforcement is
            admin-enforced, time-enforced, or
            event-enforced.";
    }
    leaf begin-time {
        type yang:date-and-time;
        description
            "The starting time for blocking
            suspicious calls.";
    }
}
```



```
    }  
    leaf end-time {  
      type yang:date-and-time;  
      description  
        "The time when blocking ends.";  
    }  
  }  
}  
}  
<CODE ENDS>
```

Figure 4: Policy Instance YANG Example for VoIP/VoLTE Security Services

Authors' Addresses

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

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

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

Phone: +82 31 299 4104
EMail: eskim86@skku.edu
URI: <http://seclab.skku.edu/people/eunsoo-kim/>

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

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

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

EMail: rkkumar@juniper.net

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

J. Kim
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
L. Xia
Huawei
July 3, 2017

I2NSF Network Security Functions-Facing Interface YANG Data Model
draft-kim-i2nsf-nsf-facing-interface-data-model-02

Abstract

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Terminology	3
3.1. Tree Diagrams	3
4. Information Model Structure	4
5. YANG Model	11
6. Security Considerations	65
7. Acknowledgements	65
8. References	66
8.1. Normative References	66
8.2. Informative References	66
Appendix A. Changes from draft-kim-i2nsf-nsf-facing-interface-data-model-01	66

1. Introduction

This document defines a YANG [RFC6020] data model for security services with the information model for Network Security Functions (NSF) facing interface in Interface to Network Security Functions (I2NSF). It provides a specific information model and the corresponding data models for three security capabilities (i.e., network security functions), such as network security control, content security control, and attack mitigation control, as defined in [i2nsf-nsf-cap-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-nsf-cap-im][i2rs-rib-data-model][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

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

3.1. Tree Diagrams

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

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".

- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. 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-nsf-cap-im].

```

module: ietf-i2nsf-nsf-facing-interface
+--rw cfg-network-security-control
|   +--rw policy* [policy-id]
|   |   +--rw policy-name      string
|   |   +--rw policy-id       uint8
|   |   +--rw rules* [rule-id]
|   |   |   +--rw rule-name          string
|   |   |   +--rw rule-id            uint8
|   |   |   +--rw rule-msg?         string
|   |   |   +--rw rule-rev?         uint8
|   |   |   +--rw rule-gid?         uint8
|   |   |   +--rw rule-class-type?  string
|   |   |   +--rw rule-reference?   string
|   |   |   +--rw rule-priority?    uint8
|   |   +--rw event
|   |   |   +--rw user-security-event* [usr-sec-event-id]
|   |   |   |   +--rw usr-sec-event-id      uint8
|   |   |   |   +--rw usr-sec-event-content  string
|   |   |   |   +--rw usr-sec-event-format   sec-event-format
|   |   |   |   +--rw usr-sec-event-type     enumeration
|   |   |   +--rw device-security-event* [dev-sec-event-id]
|   |   |   |   +--rw dev-sec-event-id      uint8
|   |   |   |   +--rw dev-sec-event-content  string
|   |   |   |   +--rw dev-sec-event-format   sec-event-format
|   |   |   |   +--rw dev-sec-event-type     enumeration
|   |   |   |   +--rw dev-sec-event-type-severity enumeration
|   |   |   +--rw system-security-event* [sys-sec-event-id]
|   |   |   |   +--rw sys-sec-event-id      uint8
|   |   |   |   +--rw sys-sec-event-content  string
|   |   |   |   +--rw sys-sec-event-format   sec-event-format
|   |   |   |   +--rw sys-sec-event-type     enumeration
|   |   |   +--rw time-security-event* [time-sec-event-id]
|   |   |   |   +--rw time-sec-event-id      uint8
|   |   |   |   +--rw time-sec-event-period-begin yang:date-and-time
|   |   |   |   +--rw time-sec-event-period-end  yang:date-and-time
|   |   |   |   +--rw time-sec-event-time-zone  string

```

```

+--rw condition
|   +--rw packet-security-condition* [pkt-security-id]
|   |   +--rw pkt-security-id                               uint8
|   |   +--rw packet-security-mac-condition
|   |   |   +--rw pkt-sec-cond-mac-dest*                   inet:port-number
|   |   |   +--rw pkt-sec-cond-mac-src*                     inet:port-number
|   |   |   +--rw pkt-sec-cond-mac-8021q*                   string
|   |   |   +--rw pkt-sec-cond-mac-ether-type*               string
|   |   |   +--rw pkt-sec-cond-mac-tci*                     string
|   |   +--rw packet-security-ipv4-condition
|   |   |   +--rw pkt-sec-cond-ipv4-header-length*          uint8
|   |   |   +--rw pkt-sec-cond-ipv4-tos*                     uint8
|   |   |   +--rw pkt-sec-cond-ipv4-total-length*            uint16
|   |   |   +--rw pkt-sec-cond-ipv4-id*                       uint8
|   |   |   +--rw pkt-sec-cond-ipv4-fragment*                uint8
|   |   |   +--rw pkt-sec-cond-ipv4-fragment-offset*         uint16
|   |   |   +--rw pkt-sec-cond-ipv4-ttl*                     uint8
|   |   |   +--rw pkt-sec-cond-ipv4-protocol*                uint8
|   |   |   +--rw pkt-sec-cond-ipv4-src*                     inet:ipv4-address
|   |   |   +--rw pkt-sec-cond-ipv4-dest*                    inet:ipv4-address
|   |   |   +--rw pkt-sec-cond-ipv4-ipopts?                  string
|   |   |   +--rw pkt-sec-cond-ipv4-sameip?                   boolean
|   |   |   +--rw pkt-sec-cond-ipv4-geoip*                   string
|   |   +--rw packet-security-ipv6-condition
|   |   |   +--rw pkt-sec-cond-ipv6-dscp*                     string
|   |   |   +--rw pkt-sec-cond-ipv6-ecn*                       string
|   |   |   +--rw pkt-sec-cond-ipv6-traffic-class*            uint8
|   |   |   +--rw pkt-sec-cond-ipv6-flow-label*               uint32
|   |   |   +--rw pkt-sec-cond-ipv6-payload-length*           uint16
|   |   |   +--rw pkt-sec-cond-ipv6-next-header*              uint8
|   |   |   +--rw pkt-sec-cond-ipv6-hop-limit*                uint8
|   |   |   +--rw pkt-sec-cond-ipv6-src*                      inet:ipv6-address
|   |   |   +--rw pkt-sec-cond-ipv6-dest*                     inet:ipv6-address
|   |   +--rw packet-security-tcp-condition
|   |   |   +--rw pkt-sec-cond-tcp-seq-num*                   uint32
|   |   |   +--rw pkt-sec-cond-tcp-ack-num*                   uint32
|   |   |   +--rw pkt-sec-cond-tcp-window-size*               uint16
|   |   |   +--rw pkt-sec-cond-tcp-flags*                     uint8
|   |   +--rw packet-security-udp-condition
|   |   |   +--rw pkt-sec-cond-udp-length*                    string
|   |   +--rw packet-security-icmp-condition
|   |   |   +--rw pkt-sec-cond-icmp-type*                     uint8
|   |   |   +--rw pkt-sec-cond-icmp-code*                     uint8
|   |   |   +--rw pkt-sec-cond-icmp-seg-num*                  uint32
|   +--rw packet-payload-security-condition* [pkt-payload-id]
|   |   +--rw pkt-payload-id                                   uint8
|   |   +--rw pkt-payload-content?                             string
|   |   +--rw pkt-payload-nocase?                             boolean

```

```

+--rw pkt-payload-depth?          uint32
+--rw pkt-payload-offset?         uint32
+--rw pkt-payload-distance?       uint32
+--rw pkt-payload-within?         uint32
+--rw pkt-payload-isdataat?       uint32
+--rw pkt-payload-dsize?          uint32
+--rw pkt-payload-replace?        string
+--rw pkt-payload-pcre?           string
+--rw pkt-payload-rpc
  +--rw pkt-payload-rpc-app-num?   uint32
  +--rw pkt-payload-rpc-version-num? uint32
  +--rw pkt-payload-rpc-procedure-num? uint32
+--rw target-security-condition* [target-sec-cond-id]
  +--rw target-sec-cond-id          uint8
  +--rw service-sec-context-cond
    +--rw name                      string
    +--rw id                        uint8
    +--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 id                        uint8
    +--rw category
      +--rw business-system?       boolean
      +--rw entertainment?         boolean
      +--rw interest?              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-volte-phone? boolean
+---rw user-security-cond* [usr-sec-cond-id]
+---rw usr-sec-cond-id uint8
+---rw user
+---rw (user-name)?
+---:(tenant)
| +---rw tenant uint8
+---:(vn-id)
| +---rw vn-id uint8
+---rw group
+---rw (group-name)?
+---:(tenant)
| +---rw tenant uint8
+---:(vn-id)
| +---rw vn-id uint8
+---rw generic-context-condition* [gen-context-cond-id]
+---rw gen-context-cond-id uint8
+---rw geographic-location
+---rw geographic-location* uint8
+---rw action
+---rw (action-type)?
+---:(ingress-action)
| +---rw ingress-action-type? ingress-action
+---:(egress-action)
| +---rw egress-action-type? egress-action
+---:(apply-profile-action)
+---rw (apply-profile-action-type)?
+---:(content-security-control)
| +---rw content-security-control-types
| +---rw antivirus-insp? boolean
| +---rw ips-insp? boolean
| +---rw ids-insp? boolean
| +---rw url-filtering-insp? boolean
| +---rw data-filtering-insp? boolean
| +---rw mail-filtering-insp? boolean
| +---rw file-blocking-insp? boolean
| +---rw file-isolate-insp? boolean

```

```

|         |--rw pkt-capture-insp?          boolean
|         |--rw application-control-insp?   boolean
|         |--rw voip-volte-insp?           boolean
+---:(attack-mitigation-control)
|   |--rw (attack-mitigation-control-type)?
|   +---:(ddos-attack)
|   |   |--rw ddos-attack-type
|   |   |   |--rw network-layer-ddos-attack
|   |   |   |   |--rw network-layer-ddos-attack-types
|   |   |   |   |--rw syn-flood-insp?      boolean
|   |   |   |   |--rw udp-flood-insp?      boolean
|   |   |   |   |--rw icmp-flood-insp?     boolean
|   |   |   |   |--rw ip-frag-flood-insp?  boolean
|   |   |   |   |--rw ipv6-related-insp?   boolean
|   |   |--rw app-layer-ddos-attack
|   |   |   |--rw app-ddos-attack-types
|   |   |   |--rw http-flood-insp?         boolean
|   |   |   |--rw https-flood-insp?        boolean
|   |   |   |--rw dns-flood-insp?          boolean
|   |   |   |--rw dns-amp-flood-insp?      boolean
|   |   |   |--rw ssl-ddos-insp?           boolean
|   +---:(single-packet-attack)
|   |   |--rw single-packet-attack-type
|   |   |   |--rw scan-and-sniff-attack
|   |   |   |   |--rw scan-and-sniff-attack-types
|   |   |   |   |--rw ip-sweep-insp?       boolean
|   |   |   |   |--rw port-scanning-insp?  boolean
|   |   |--rw malformed-packet-attack
|   |   |   |--rw malformed-packet-attack-types
|   |   |   |--rw ping-of-death-insp?      boolean
|   |   |   |--rw teardrop-insp?           boolean
|   |--rw special-packet-attack
|   |   |--rw special-packet-attack-types
|   |   |--rw oversized-icmp-insp?         boolean
|   |   |--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      uint8
|         +--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 sip-header-expire-time yang:date-and-time
|         |         |         +--rw sip-header-user-agent uint32
|         |         +--rw cell-region* [cell-id-region]
|         |         +--rw cell-id-region  uint32
|         +--rw action
|         |         +--rw (action-type)?
|         |         |         +---:(ingress-action)
|         |         |         |         +--rw ingress-action-type?  ingress-action
|         |         |         +---:(egress-action)
|         |         |         |         +--rw egress-action-type?    egress-action
+---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-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 port-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-nsf-cap-iml].

<CODE BEGINS> file "ietf-i2nsf-nsf-facing-interface@2017-07-03.yang"

```

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

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

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

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

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

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

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

     Editor: Jaehoon Paul Jeong

```

<mailto:pauljeong@skku.edu>

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

```
description
  "This module defines a YANG data module for network security
  functions.";
revision "2017-07-03" {
  description "The second revision";
  reference
    "draft-xibassnez-i2nsf-capability-01;
    draft-kim-i2nsf-nsf-facing-interface-data-model-02";
}

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

```
        "This is used for SecEventFormat.";
    }

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

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

```
    }
    description
        "This is used for egress action.";
}

//Groupings

container cfg-network-security-control {
    description
        "Configuration for Network Security Control.";

    list policy {
        key "policy-id";
        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 uint8;
            mandatory true;
            description
                "The ID of the policy.
                This must be unique.";
        }
    }

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

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



```
    }

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

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

    leaf rule-rev {
        type uint8;
        description
            "The sid keyword is almost every time
             accompanied by reb.";
    }

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

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

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

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

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

  list user-security-event {
    key usr-sec-event-id;
    description
      "The purpose of this class is to represent events that
      are initiated by a user, such as logon and logoff
      events. Information in this event may be used as part
      of a test to determine if the Condition clause in
      this ECA Policy Rule should be evaluated or not.
      Examples include user identification data and the
      type of connection used by the user.";

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

    leaf usr-sec-event-content {
      type string;
      mandatory true;
      description
        "This is a mandatory string that contains the content
        of the UserSecurityEvent. The format of the content
        is specified in the usrSecEventFormat class
        attribute, and the type of event is defined in the
        usrSecEventType class attribute. An example of the
        usrSecEventContent attribute is a string hrAdmin,
```

```
        with the usrSecEventFormat set to 1 (GUID) and the
        usrSecEventType attribute set to 5 (new logon).";
    }

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

```

        "If usrSecEventType is user
        logon";
    }
    enum user-logoff {
        description
            "If usrSecEventType is user
            logoff";
    }
    enum user-access-request {
        description
            "If usrSecEventType is user
            access request";
    }
    enum user-access-granted {
        description
            "If usrSecEventType is user
            granted";
    }
    enum user-access-violation {
        description
            "If usrSecEventType is user
            violation";
    }
}
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 sec-event-format;
  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 enumeration {
    enum unknown {
      description
        "If devSecEventType is unknown";
    }
    enum comm-alarm {
      description
        "If devSecEventType is communications
        alarm";
    }
    enum quality-of-service-alarm {
      description
```

```
        "If devSecEventType is quality of service
        alarm";
    }
    enum process-err-alarm {
        description
            "If devSecEventType is processing error
            alarm";
    }
    enum equipment-err-alarm {
        description
            "If devSecEventType is equipment error
            alarm";
    }
    enum environmental-err-alarm {
        description
            "If devSecEventType is environmental error
            alarm";
    }
}
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 enumeration {
        enum unknown {
            description
                "If devSecEventType is unknown";
        }
        enum cleared {
            description
                "If devSecEventTypeSeverity is cleared";
        }
        enum indeterminate {
            description
                "If devSecEventTypeSeverity is
                indeterminate";
        }
        enum critical {
            description
                "If devSecEventTypeSeverity is critical";
        }
        enum major {
            description
                "If devSecEventTypeSeverity is major";
        }
    }
}
```

```
    }
    enum minor {
        description
            "If devSecEventTypeSeverity is minor";
    }
    enum warning {
        description
            "If devSecEventTypeSeverity is warning";
    }
}
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 sec-event-format;
    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 enumeration {
        enum unknown {
            description
                "If sysSecEventType is unknown";
        }
        enum audit-log-written-to {
            description
                "If sysSecEventTypeSeverity
                is that audit log is written to";
        }
        enum audit-log-cleared {
            description
                "If sysSecEventTypeSeverity
                is that audit log is cleared";
        }
        enum policy-created {
            description
                "If sysSecEventTypeSeverity
                is that policy is created";
        }
        enum policy-edited{
            description
                "If sysSecEventTypeSeverity
                is that policy is edited";
        }
        enum policy-deleted{
            description

```



```
        "If sysSecEventTypeSeverity
        is that policy is deleted";
    }
    enum policy-executed{
        description
        "If sysSecEventTypeSeverity
        is that policy is executed";
    }
}
mandatory true;
description
    "This is a mandatory uint 8 enumerated integer, which
    is used to specify the type of event that involves
    this device.";
}
}

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

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

    leaf time-sec-event-period-begin {
        type yang:date-and-time;
        mandatory true;
        description
            "This is a mandatory DateTime attribute, and
            represents the beginning of a time period."
    }
}
```

```
        It has a value that has a date and/or a time
        component (as in the Java or Python libraries).";
    }

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

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

container condition {
    description
        "TBD";
    list packet-security-condition {
        key pkt-security-id;
        description
            "The purpose of this Class is to represent packet header
            information that can be used as part of a test to
            determine if the set of Policy Actions in this ECA
            Policy Rule should be executed or not. This class is
            abstract, and serves as the superclass of more detailed
            conditions that involve different types of packet
            formats.";
        leaf pkt-security-id {
            type uint8;
            mandatory true;
            description
                "The ID of the packet-security-condition.";
        }
    }
}
```

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

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

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

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

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

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

container packet-security-ipv4-condition {
  description
```

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

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

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

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

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

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

leaf-list pkt-sec-cond-ipv4-fragment-offset {
  type uint16;
```

```
    description
      "Fragment offset field along with Don't Fragment
       and More Fragment flags in the IP protocol
       header are used for fragmentation and reassembly
       of IP datagrams.";
  }

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

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

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

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

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

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

```
        the destination IP.";
    }

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

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

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

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

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

    leaf-list pkt-sec-cond-ipv6-flow-label {
        type uint32;
        description
```

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

  list packet-payload-security-condition {
    key "pkt-payload-id";
    description
      "The ID of the pkt-payload.
       This is key for pkt-payload-list.
       This must be unique.";

    leaf pkt-payload-id {
      type uint8;
      mandatory true;
      description
        "The ID of the packet payload.
         This must be unique.";
    }

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

    leaf pkt-payload-nocase {
```

```
    type boolean;
    description
        "If you do not want to make a distinction
        between uppercase and lowercase characters,
        you can use nocase.";
}

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

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

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

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

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

leaf pkt-payload-dsize {
```

```
    type uint32;
    description
        "With the dsize keyword, you can match on the
        size of the packet payload.";
}

leaf pkt-payload-replace {
    type string;
    description
        "The replace content modifier can only be used
        in ips. It adjusts network traffic.";
}

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

container pkt-payload-rpc{
    description
        "The rpc keyword can be used to match in the
        SUNRPC CALL on the RPC procedure numbers and
        the RPC version.";

    leaf pkt-payload-rpc-app-num {
        type uint32;
        description
            "<application number>.";
    }

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

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

list target-security-condition {
    key "target-sec-cond-id";
    description
```

"Under the circumstances of network, it mainly refers to the service, application, and device.";

```
leaf target-sec-cond-id {
  type uint8;
  mandatory true;
  description
    "The ID of the target.
     This must be unique.";
}
container service-sec-context-cond{
  description
    "A service is an application identified by a
     protocol type and port number, such as TCP,
     UDP, ICMP, and IP.";

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

  leaf id {
    type uint8;
    mandatory true;
    description
      "The ID of the service.
       This must be unique.";
  }

  container protocol {
    description
      "Protocol types:
       TCP, UDP, ICMP, ICMPv6, IP, and etc.";

    leaf tcp {
      type boolean;
      description
        "TCP protocol type.";
    }

    leaf udp {
      type boolean;
      description
        "UDP protocol type.";
    }
  }
}
```

```
    leaf icmp {
        type boolean;
        description
            "ICMP protocol type.";
    }

    leaf icmpv6 {
        type boolean;
        description
            "ICMPv6 protocol type.";
    }

    leaf ip {
        type boolean;
        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.";

      leaf ingress-action-type {
        type ingress-action;
        description
          "Ingress action type: permit, deny, and mirror.";
      }
    }

    case egress-action {
      description
        "The egress actions consist of invoke-signaling,
         tunnel-encapsulation, and forwarding.";

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

    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.";

      leaf antivirus-insp {
        type boolean;
        description
          "Additional inspection of antivirus.";
      }

      leaf ips-insp {
        type boolean;
        description
          "Additional inspection of IPS.";
      }

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

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

      leaf data-filtering-insp {
```

```
        type boolean;
        description
            "Additional inspection of data filtering.";
    }

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

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

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

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

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

    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.";

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

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

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

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

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

          leaf ip-frag-flood-insp {
```



```
        type boolean;
        description
            "Additional Inspection of
             IP Fragment Flood.";
    }

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

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

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

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

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

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

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

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

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

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

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

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

```
    }  
  }  
}  
  
container malformed-packet-attack {  
  description  
    "Malformed Packet Attack.";  
  container malformed-packet-attack-types {  
    description  
      "Malformed packet attack types:  
      Ping of Death Attack, Teardrop Attack,  
      and etc.";   
  
    leaf ping-of-death-insp {  
      type boolean;  
      description  
        "Additional Inspection of  
        Ping of Death Attack.";   
    }  
  
    leaf teardrop-insp {  
      type boolean;  
      description  
        "Additional Inspection of  
        Teardrop Attack.";   
    }  
  }  
}  
  
container special-packet-attack {  
  description  
    "special Packet Attack.";  
  container special-packet-attack-types {  
    description  
      "Special packet attack types:  
      Oversized ICMP Attack, Tracert Attack,  
      and etc.";   
  
    leaf oversized-icmp-insp {  
      type boolean;  
      description  
        "Additional Inspection of  
        Oversize ICMP Attack.";   
    }  
  
    leaf tracert-insp {  
      type boolean;  
      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.";

            leaf ingress-action-type {
                type ingress-action;
                description
                    "Ingress-action-type: permit, deny,
                    and mirror.";
            }
        }

        case egress-action {
            description
                "The egress actions consist of
                mirror and etc.";

            leaf egress-action-type {
                type egress-action;
            }
        }
    }
}
```



```
    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.";
        }
      }
    }
  }
}
```


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-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-01 (work in progress), March 2017.
- [i2rs-rib-data-model] Wang, L., Ananthakrishnan, H., Chen, M., Dass, A., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.
- [supa-policy-info-model] Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-03 (work in progress), May 2017.
- [i2nsf-framework] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-05 (work in progress), May 2017.

Appendix A. Changes from
draft-kim-i2nsf-nsf-facing-interface-data-model-01

The following changes are made from
draft-kim-i2nsf-nsf-facing-interface-data-model-01:

- o Event components are revised by using enumeration according to the draft about Information Model of NSFs Capabilities [i2nsf-nsf-cap-im].

- o Action components such as ingress-action and egress-action are revised by using enumeration.
- o Action components such as apply-profile-action are revised to allow for the execution of advanced NSFs.

Authors' Addresses

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

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

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

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

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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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 17, 2018

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei
July 16, 2017

Information model for Client-Facing Interface to Security Controller
draft-kumar-i2nsf-client-facing-interface-im-03

Abstract

This document defines information model for Client-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.

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 17, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
2. Conventions Used in this Document	3
3. Information Model for Multi Tenancy	4
3.1. Policy-Domain	4
3.2. Policy-Tenant	5
3.3. Policy-Role	5
3.4. Policy-User	5
3.5. Policy-Management-Authentication-Method	6
4. Information Model for Policy Endpoint Groups	6
4.1. Metadata-Source	7
4.2. User-Group	7
4.3. Device-Group	8
4.4. Application-Group	8
4.5. Location-Group	9
5. Information Model for Threat Prevention	9
5.1. Threat-Feed	9
5.2. Custom-List	10
5.3. Malware-Scan-Group	10
5.4. Event-Map-Group	11
6. Information Model for Telemetry Data	11
6.1. Telemetry-Data	11
6.2. Telemetry-Source	12
6.3. Telemetry-Destination	13
7. Information Model for Policy Instance	13
7.1. Policy-Calendar	13
7.2. Policy-Action	14
7.3. Policy-Rule	14
7.4. Policy-Instance	15
8. Security Considerations	16
9. IANA Considerations	16
10. Acknowledgements	16
11. Informative References	16
Authors' Addresses	17

1. Introduction

The Security Controller's Client-Facing interfaces would be built using a set of objects, with each object capturing a unique set of information from Security Admin 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 draft is in accordance with interface requirements as defined in [I-D.ietf-i2nsf-client-facing-interface-req].

The [RFC3444] explains differences between an information and data model. This draft use those guidelines to define information model for Client-Facing interface in this draft. A data model, that represents an implementation of the proposed information model in a specific data representation language, will be defined in a separate draft.

2. Conventions Used in this Document

BSS: Business Support System

CLI: Command Line Interface

CMDB: Configuration Management Database

Controller: Used interchangeably with Service Provider 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-terminology]

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. 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 HR, Finance, 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.

3.1. 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 this account was created or last modified

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

3.2. 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 manages 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 this account was created or last modified

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

3.3. Policy-Role

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

The Policy-Role object SHALL have following information:

Name: This field identifies name of the role

Date: Date this role was created or last modified

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

3.4. 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 do policy management. A user may be an individual, system, or application requiring access to Security Controller.

The Policy-User object SHALL have following information:

Name: Name of user

Date: Date this user was created or last modified

Password: User password for basic authentication

Email: E-mail address of user

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

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

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

3.5. Policy-Management-Authentication-Method

This object represents authentication schemes supported by Security Controller.

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

Name: This field identifies name of this object

Date: Date 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

4. Information Model for Policy Endpoint Groups

The Policy Endpoint Group is 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 Policy Endpoint Group. This section lists these objects and relationship among these objects.

4.1. Metadata-Source

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

Metadata-Source object SHALL have following information:

Name: This field identifies name of this object

Date: Date this object was created or last modified

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

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

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

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

4.2. User-Group

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

The User-Group object SHALL have following information:

Name: This field identifies the name of this object

Date: Date 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 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

4.3. Device-Group

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

The Device-Group object SHALL have following information:

Name: This field identifies the name of this object

Date: Date 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

Metadata-Server: This field should be reference to a Metadata-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

4.4. Application-Group

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

The Application-Group object SHALL have following information:

Name: This field identifies the name of this object

Date: Date 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

Metadata-Server: This field should be reference to a Metadata-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

4.5. Location-Group

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

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

Name: This field identifies the name of this object

Date: Date 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

Metadata-Server: This field should be reference to a Metadata-
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

5. Information Model for Threat Prevention

The threat prevention plays an important part in the overall security
posture by reducing the attack surface. 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.

5.1. Threat-Feed

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

The Threat-Feed object SHALL have following information:

Name: This field identifies the name of this object

Date: Date 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

5.2. Custom-List

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

The Custom-List object SHALL have following information:

Name: This field identifies the name of this object

Date: Date 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.

5.3. 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 following information:

Name: This field identifies the name of this object

Date: Date 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 list of file types needed to be scanned for the virus

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

5.4. 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: Date 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

6. Information Model for Telemetry Data

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

6.1. Telemetry-Data

This object contains information collected for telemetry.

The Telemetry-Data object SHALL have following information:

Name: This field identifies the name of this object

Date: Date 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 NSF

6.2. Telemetry-Source

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

The Telemetry-Source object SHALL have following information:

Name: This field identifies the name of this object

Date: Date this object was created or last modified

Source-Type: This field contains 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 username and password to authenticate with the NSF

Collection-Interval: This field contains time in milliseconds between each data collection. For example, a value of 5000 means data is streamed to collector every 5 seconds. Value of 0 means data streaming is event-based.

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

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

QoS-Marking: This field contains DSCP value source MUST mark on its generated telemetry packets

6.3. 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 following information:

Name: This field identifies the name of this object

Date: Date 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 the username and password for 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.

7. Information Model for Policy Instance

In order to express a Security Policy, a policy instance must have complete information such as where and when a policy need to be applied. This is done by defining a set of managed objects and relationship among them. A policy may be related segmentation, threat mitigation or telemetry data collection from NSF in the network.

7.1. Policy-Calendar

This object contains information related to scheduling a policy. The policy could be activated based on a time calendar or security event including threat level changes.

The Policy-Calendar object SHALL have following information:

Name: This field identifies the name of this object

Date: Date this object was created or last modified

Enforecement-Type: This field identifies whether the policy enforcement is "ADMIN-ENFORCED", "TIME-ENFORCED" or "EVENT-ENFORCED"

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

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

7.2. Policy-Action

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

The Policy-Action object SHALL have following information:

Name: This field identifies the name of this object

Date: Date this object was created or last modified

Primary-Action: This field identifies the action when a rule is matched by NSF. The action could be one of "PERMIT", "DENY", "REDIRECT", "RATE-LIMIT", "TRAFFIC-CLASS", "AUTHENTICATE-SESSION", "IPS", "APP-FIREWALL", or "COLLECT"

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

7.3. Policy-Rule

This object represents rules that a Security Admin want to define in order to express its business objectives in a Security Policy.

The Policy-Rule object SHALL have following information:

Name: This field identifies the name of this object

Date: Date this object was created or last modified

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

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

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

Match-Direction: This field identifies if the match criteria is to evaluated for both direction of the traffic or only in one direction with default of allowing in the other direction for stateful match conditions. This is optional and by default rule should apply in both directions

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

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

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

7.4. Policy-Instance

This object represents a mechanism to express a Security Policy by Security Admin using Security Controller Client-Facing interface; the policy would be enforced on a NSF.

The Policy-Instance object SHALL have following information:

Name: This field identifies the name of this object

Date: Date this object was created or last modified

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

Scheduling-Type: This field specifies when this policy should be scheduled. The policy could be scheduled based on time calendar or event-map

Scheduling-Information: This field contains reference to Policy-Calendar or Event-Map-Group based on Schedule-Type'

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

8. Security Considerations

Information model provides mechanism to protect Client-Facing interface to Security controller. One of the specified mechanism must be used to protect Enterprise network, data and all resources from external attacks. This model mandates that interface must have proper authentication and authorization with Role Based Access Controls to address multi-tenancy requirement. The draft does not mandate that a particular mechanism be used as different organization may have different needs based on their deployment.

9. IANA Considerations

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

10. Acknowledgements

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

11. Informative References

[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-02 (work in progress), July 2017.

[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-16 (work in progress), May 2017.

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-04 (work in progress), July 2017.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <<http://www.rfc-editor.org/info/rfc3444>>.

Authors' Addresses

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

Email: rakeshkumarccloud@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

Interface to Network Security Functions (I2NSF)
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2018

L. Xia
Q. Lin
Huawei
July 2, 2017

Policy Object for Interface to Network Security Functions (I2NSF)
draft-xia-i2nsf-security-policy-object-01

Abstract

This document describes policy object used in the Interface to Network Security Functions (I2NSF) policy rules to provide re-usability and defines essential attributes for each policy object.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. Terminology	4
4. Policy Object	4
4.1. Address Object	5
4.1.1. The addressName Attribute	5
4.1.2. The addressRange Attribute	5
4.2. Address Group Object	6
4.2.1. The addressGroupName Attribute	6
4.2.2. The addressReference Attribute	6
4.2.3. The addressRange Attribute	6
4.3. Service Object	7
4.3.1. The serviceName Attribute	7
4.3.2. The serviceList Attribute	7
4.3.2.1. The serviceProtocol Attribute	7
4.3.2.2. The serviceProtocolNumber Attribute	8
4.3.2.3. The serviceICMPType Attribute	8
4.3.2.4. The serviceICMPCode Attribute	8
4.3.2.5. The serviceSourcePort Attribute	8
4.3.2.6. The serviceDestinationPort Attribute	8
4.4. Service Group Object	8
4.4.1. The serviceGroupName Attribute	9
4.4.2. The serviceReference Attribute	9
4.5. Application Object	9
4.5.1. The applicationName Attribute	9
4.5.2. The applicationCategory Attribute	9
4.5.3. The applicationSubCategory Attribute	9
4.5.4. The applicationTransmissionModel Attribute	10
4.5.5. The applicationVulnerability Attribute	10
4.5.6. The applicationRiskLevel Attribute	10
4.6. Application Group Object	10
4.6.1. The applicationGroupName Attribute	10
4.6.2. The applicationReference Attribute	10
4.7. Schedule Object	10
4.7.1. The scheduleName Attribute	11
4.7.2. The scheduleList Attribute	11
4.7.2.1. The scheduleType Attribute	11
4.7.2.2. The scheduleStartTime Attribute	11
4.7.2.3. The scheduleEndTime Attribute	11
4.7.2.4. The scheduleWeekDay Attribute	11
4.8. User Object	11
4.8.1. The userName Attribute	12
4.8.2. The userParentGroup Attribute	12
4.8.3. The userSecurityGroup Attribute	12
4.8.4. The userDomain Attribute	12
4.8.5. The userPassword Attribute	13

4.8.6. The userExpirationTime Attribute	13
4.9. User Group Object	13
4.9.1. The userGroupName Attribute	13
4.9.2. The userGroupParentGroup Attribute	13
4.9.3. The userGroupDomain Attribute	13
4.9.4. The userGroupReference Attribute	13
4.10. Security Group Object	13
4.10.1. The securityGroupName Attribute	14
4.10.2. The securityGroupParentGroup Attribute	14
4.10.3. The securityGroupDomain Attribute	14
4.10.4. The securityGroupType Attribute	14
4.10.5. The securityGroupReference Attribute	14
4.10.6. The securityGroupFilters Attribute	14
5. Acknowledgements	14
6. IANA Considerations	14
7. Security Considerations	14
8. References	15
8.1. Normative References	15
8.2. Informative References	15
Appendix A. Application Attributes	15
A.1. Category and Subcategory	15
A.2. Data Transmission Model	16
A.3. Vulnerability	17
Appendix B. Example of Application Scenario for Policy Object	17
B.1. Security Policy Control for Marketing Departments	20
B.2. Security Policy Control for R&D Departments	20
B.3. Security Policy Control for Server Access of Internet Users	21
Authors' Addresses	21

1. Introduction

I2NSF policy consists of policy rules that are used to provision NSF instances. The I2NSF policy rule is defined by using "Event-Condition-Action" (ECA) model described in I2NSF framework draft [I-D.ietf-i2nsf-framework]. In the ECA model, a condition is used to determine whether or not the predefined actions should be executed. A condition usually consists of several attributes. Information Model of NSFs Capabilities [I-D.ietf-i2nsf-capability] describes attributes of different condition subclasses. When configuring policy rules by attributes, it is common to see that the same attribute or the same set of several attributes are configured for several times or more. And modifications of the policy rules are also very tedious and time-consuming.

To facilitate the provisioning of NSF instances, this document describes a set of policy objects which are reusable and can be referenced by variable I2NSF policy rules. A policy object consists

of a name attribute that identifies itself and one or several attributes that are typically used together to represent a certain condition. For example, protocol type and port number are usually used together to represent a certain service. Each policy object is predefined and named in order to be used in I2NSF policy rules. By defining policy objects, the creation and maintenance of policy rules are greatly simplified.

- o A policy object can be referenced in different policy rules as required to provide re-usability. And a policy rule can reference several policy objects.
- o The modification of a policy object will be propagated to the I2NSF policy rules that reference this object. No modification should be made to the related policy rules.

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

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Terminology

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

4. Policy Object

IP addresses, port numbers, protocol types, services, applications, user accounts are commonly used attributes to determine whether a certain condition occurs. In real-world deployment, these attributes are often configured for many times. The definition of policy objects could help to minimize the configuration effort and provide simplicity.

Figure 1 shows the policy objects defined in this document and their relationships.

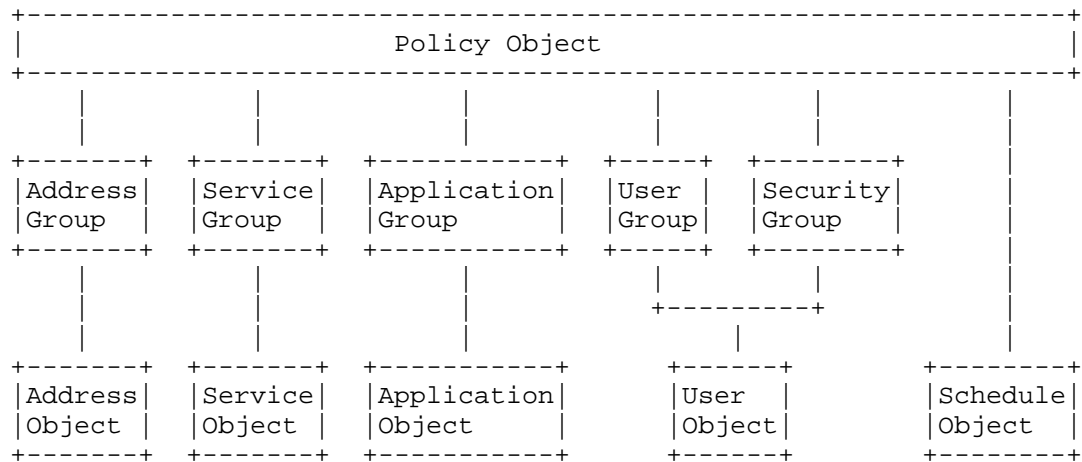


Figure 1: The Policy Objects Overview

4.1. Address Object

A of IPv4/IPv6 addresses or MAC addresses can be defined as an address object, which may belongs to an address group object. An address object consists of the following attributes:

4.1.1. The addressName Attribute

This attribute defines a unique name for the address object.

4.1.2. The addressRange Attribute

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

An IPv4 address range can be defined by one of the following representations:

- o IPv4 address with wildcard mask, e.g., 10.10.1.2\0.0.0.255.
- o IPv4 address with subnet mask (subnet mask address or length of the subnet mask), e.g., 10.10.1.2/255.255.255.0 or 10.10.1.2/32.
- o Start address and end address of the IPv4 address range, e.g., 10.10.1.2-10.10.1.254.

An IPv6 address range can be defined by one of the following representations:

- o IPv6 address with length of the prefix, e.g., a234::120/120.
- o Start address and end address of the IPv6 address range, e.g., a231::a237-b231::b237.

4.2. Address Group Object

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

4.2.1. The addressGroupName Attribute

This attribute defines a unique name for the address group object.

4.2.2. The addressReference Attribute

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

4.2.3. The addressRange Attribute

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

An IPv4 address range can be defined by one of the following representations:

- o IPv4 address with wildcard mask, e.g., 10.10.1.2\0.0.0.255.
- o IPv4 address with subnet mask (subnet mask address or length of the subnet mask), e.g., 10.10.1.2/255.255.255.0 or 10.10.1.2/32.
- o Start address and end address of the IPv4 address range, e.g., 10.10.1.2-10.10.1.254.

An IPv6 address range can be defined by one of the following representations:

- o IPv6 address with length of the prefix, e.g., a234::120/120.
- o Start address and end address of the IPv6 address range, e.g., a231::a237-b231::b237.

4.3. Service Object

A service object can be a single service based on IP, or ICMP, or UDP, or TCP, or SCTP and it can also contain a set of services. To identify services based on different protocols, different attributes should be specified (see Section 4.3.2 The serviceList Attribute).

- o IP based service is recognized by the value of the protocol field in IP packet header.
- o ICMP or ICMPv6 based service is recognized by two header fields in the ICMP or ICMPv6 packets: type field and code field.
- o UDP, TCP, or SCTP based service is recognized by port number. The source port number and destination port number are used to identify the sending and receiving service respectively.

A set of well-known services should be predefined by NSFs as service objects to support direct reference. A service object consists of the following attributes:

4.3.1. The serviceName Attribute

This attribute defines a unique name for the service object.

4.3.2. The serviceList Attribute

This attribute defined a set of services. Each service can be defined by a subset of the following sub-attributes, according to the protocol on which the service is based.

- o For IP based service, the serviceProtocolNumber attribute should be specified.
- o For ICMP or ICMPv6 based service, the serviceICMPType attribute and serviceICMPCode attribute should be specified.
- o For UDP, TCP, or SCTP based service, the serviceSourcePort attribute and serviceDestinationPort attribute should be specified.

4.3.2.1. The serviceProtocol Attribute

This attribute defines the protocol type on which the service is based, IP, ICMP, ICMPv6, TCP, UDP, or SCTP.

4.3.2.2. The serviceProtocolNumber Attribute

This attribute defines the protocol number for IP based service. The protocol number is the value of protocol field in IP packet header which identifies the corresponding upper layer protocol. For example, to define a service object for IPsec Encapsulating Security Payload, this attribute should be set to 50.

4.3.2.3. The serviceICMPType Attribute

This attribute defines the ICMP/ICMPv6 type number for ICMP/ICMPv6 based service. This attribute shall be used together with serviceICMPCode attribute. For example, to define a service object for IPv4 ping request, this attribute should be set to 8 and serviceICMPCode attribute should be set to 0.

4.3.2.4. The serviceICMPCode Attribute

This attribute defines the ICMP/ICMPv6 message code for ICMP/ICMPv6 based service. This attribute shall be used together with serviceICMPType attribute. For example, to define a service object for IPv6 ping request, this attribute should be set to 0 and serviceICMPCode should be set to 128.

4.3.2.5. The serviceSourcePort Attribute

This attribute defines the source port number for service based on TCP, UDP or SCTP. The value could be a single port number which identifies a single service, or a range of port numbers which identify a family of services or several services in consecutive port numbers. For example, to define a service object using port number greater or equal to 1024 and enforce security policy on the traffic that this object sends out, this attribute should be set as a port range, 1024-65535.

4.3.2.6. The serviceDestinationPort Attribute

This attribute defines the destination port number for service based on TCP, UDP or SCTP. The value could be a single port number or a range of port numbers. For example, to define a service object for HTTP and enforce security policy on the traffic that communicates with this service object, this attribute should be set to 80.

4.4. Service Group Object

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

4.4.1. The serviceGroupName Attribute

This attribute defines a unique name for the service group object.

4.4.2. The serviceReference Attribute

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

4.5. Application Object

Due to the diversity and large amount of applications, it is not able to identify a certain application based on protocol type and port number. For example, there are many web applications with different risk levels run on ports 80 and 443 using HTTP and HTTPS, such as web gaming application and web chat application. Protocol type and port number could not distinguish applications using the same application protocol. In this document, category, subcategory, data transmission model, vulnerability, and risk level are used to describe an application. A set of well-known application objects should be predefined in NSFs to support direct reference. For a newly created application object, the rules for NSFs to identify this application in the traffic should be configured. In this document, the configuration of these rules is out of scope. An application object consists of the following attributes:

4.5.1. The applicationName Attribute

This attribute defines a unique name for the application object.

4.5.2. The applicationCategory Attribute

This attribute defines the category for the application. The value of this attribute is selected from a predefined set of categories, e.g., general category, network category. Values of this attribute are defined in Appendix A.1. Each category is broken down into several subcategories.

4.5.3. The applicationSubCategory Attribute

This attribute defines the subcategory for the application. The value of this attribute is selected from the predefined subcategories of a category. For example, the entertainment category has seven subcategories, and Facebook application belongs to social networking subcategory. (See Appendix A.1 for details about subcategory and examples of applications belong to each subcategory.)

4.5.4. The applicationTransmissionModel Attribute

This attribute defines the data transmission model of the application. Four types of data transmission model are defined in this document: client/server, browser-based, network protocol, peer-to-peer. (See Appendix A.2 for more details.)

4.5.5. The applicationVulnerability Attribute

This attribute describes a set of possible threats for the application. The values of this attribute are selected from a predefined set of vulnerabilities, e.g., exploitable, bandwidth consuming. (See Appendix A.3 for more details.)

4.5.6. The applicationRiskLevel Attribute

This attribute defines a risk level for the application. The value of this attribute is selected from a predefined number of risk levels, e.g., 5 risk levels. The risk level is determined by the vulnerabilities of this application object.

4.6. Application Group Object

An application group object is a collection of application objects that will be processed according to the same security policy. It consists of the following attributes:

4.6.1. The applicationGroupName Attribute

This attribute defines a unique name for the application group object.

4.6.2. The applicationReference Attribute

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

4.7. Schedule Object

A schedule object is a set of time ranges. There are two kinds of time ranges: periodic time range and absolute time range. A periodic time range occurs every week. An absolute time range occurs only once. A schedule object consists of the following attributes:

4.7.1. The scheduleName Attribute

This attribute defines a unique name for the schedule object.

4.7.2. The scheduleList Attribute

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

- o For a periodic time range, the start and end time in a day, and the days in a week that it takes effect, should be specified.
- o For an absolute time range, the start time and date, and the end time and date, should be specified.

4.7.2.1. The scheduleType Attribute

This attribute defines the type of a time range, periodic, absolute.

4.7.2.2. The scheduleStartTime Attribute

For a periodic time range, this attribute defines the start time in a week day, such as 9:00 am. For an absolute time range, this attribute defines the start time and start date, such as 00:00 am 2017-07-03.

4.7.2.3. The scheduleEndTime Attribute

For a periodic time range, this attribute defines the end time in a week day, such as 18:00 pm. For an absolute time range, this attribute defines the end time and end date, such as 23:59 pm 2017-07-03.

4.7.2.4. The scheduleWeekDay Attribute

This attribute defines the days in a week that the periodic time range takes effect. For example, to define working hours in a week, the scheduleStartTime can be set to 9:00 am, the scheduleEndTime can be set to 18:00 pm, and this attribute should contain five days, from Monday to Friday.

4.8. User Object

A user object identifies a person who may access network resources. It is the basis of implementing user-based policy control. The user objects may be created locally on the NSFs, or be imported from third parties, such as authentication servers. User objects that require the same policy enforcement are grouped as user group objects or

security group objects. The user group objects are organized as a hierarchical structure, See Figure 2. A security group object consists of user objects from different user group objects that require the same policy enforcement.

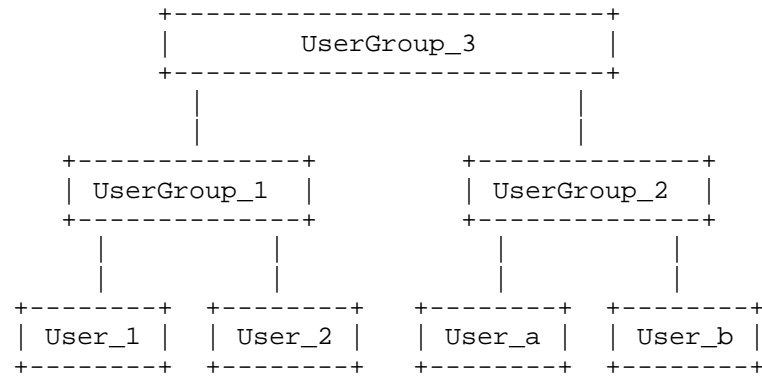


Figure 2: Hierarchical Structure of User Group

A user object consists of the following attributes:

4.8.1. The userName Attribute

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

4.8.2. The userParentGroup Attribute

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

4.8.3. The userSecurityGroup Attribute

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

4.8.4. The userDomain Attribute

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

4.8.5. The userPassword Attribute

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

4.8.6. The userExpirationTime Attribute

This attribute defines when will this user object expire.

4.9. User Group Object

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

4.9.1. The userGroupName Attribute

This attribute defines a unique name for the user group object.

4.9.2. The userGroupParentGroup Attribute

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

4.9.3. The userGroupDomain Attribute

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

4.9.4. The userGroupReference Attribute

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

4.10. Security Group Object

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

4.10.1. The securityGroupName Attribute

This attribute defines a unique name for the security group object.

4.10.2. The securityGroupParentGroup Attribute

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

4.10.3. The securityGroupDomain Attribute

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

4.10.4. The securityGroupType Attribute

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

4.10.5. The securityGroupReference Attribute

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

4.10.6. The securityGroupFilters Attribute

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

5. Acknowledgements

6. IANA Considerations

This document requires no IANA actions.

7. Security Considerations

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

8. References

8.1. Normative References

- [I-D.ietf-i2nsf-capability]
Xia, L., Strassner, J., Basile, C., and D. Lopez,
"Information Model of NSFs Capabilities", 2017,
<<https://tools.ietf.org/pdf/draft-xibassnez-i2nsf-capability-01.pdf>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

8.2. Informative References

- [I-D.ietf-i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", 2017, <<https://tools.ietf.org/pdf/draft-ietf-i2nsf-framework-05.pdf>>.
- [I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", 2017, <<https://tools.ietf.org/pdf/draft-ietf-i2nsf-terminology-03.pdf>>.

Appendix A. Application Attributes

An application object is described by five items, category, subcategory, data transmission model, vulnerability and risk level. This appendix illustrates the possible values of applicationCategory attribute, applicationSubCategory attribute, applicationTransmissionModel attribute and applicationVulnerability attribute.

A.1. Category and Subcategory

This section lists the possible values for applicationCategory attribute and applicationSubCategory attribute.

Category	Subcategory	Example
General	General_TCP General_UDP Other	TCP-based applications UDP-based applications Error_Packets
Network	IP_Protocol Encrypted_Tunnel Infrastructure Proxy Network_Admin	ICMP, IGMP, OSPF GRE, L2TP, IKEv2 FTP, HTTP, DNS HTTP_Proxy Syslog
General_Internet	Search_Engine Web_Content_Aggregate Utility Web_Desktop Browser_Plugin File_Sharing FileShare_P2P Network_Storage App_Download Software_Update Web_Browsing	www.google.com FeedReader Google Earth Zimbra Desktop Adobe XDCC BT, Thunder DBank AndroidMarket WindowsUpdate OperaMobile
Entertainment	Social_Networking Instant_Messaging Media_Sharing Peer_Casting Web_Video Game VoIP	Facebook, Twitter QQ, MSN RayV QQLive YouKu, YouTube QQGame Skype
Business_Systems	Electronic_Business Remote_Access Database Finance Enterprise_Application Internet_Conferencing Data_Backup Email	Taobao Radmin Oracle DaZhiHui, Fix LotusNotes NetMeeting Rsync GMail

A.2. Data Transmission Model

This section lists four types of models for applicationTransmissionModel attribute.

Model	Description
Client/Server	One or more client applications communicate with a communicate with a sever
Browser-Based	Applications run on web browser
Network Protocol	Applications that is used for system-to-system communication
Peer-to-Peer	Applications directly communicate with each other

A.3. Vulnerability

This section lists five types of possible risks for applicationVulnerability attribute.

Vulnerability	Description
Exploitable	Has known vulnerabilities
Evasive	Used to evade the original purpose and traverse the firewall, for example, a proxy software
Data Loss	Used for transferring files or uploading texts, may cause information leaks
Used by Malware	Used by malware for propagation, attack, or data theft, or distributed with malware
Bandwidth Consuming	Consume large bandwidths

Appendix B. Example of Application Scenario for Policy Object

This appendix describes the utilization of policy objects in policy rules for enterprise scenario.

NSFs are key components to protect security in enterprise network. For the typical architecture of an enterprise network, NSFs are deployed on-premise at network perimeter. The inbound and outbound traffic of the enterprise network are processed according to the predefined security policies rules.

Figure 3 demonstrates an example of enterprise network topology. Firewall is a typical NSF that used at the network perimeter to protect enterprise intranet. Assuming that the firewall should be provisioned to provide different network access controls for marketing departments and R&D departments.

- o Marketing departments are allowed to access the Internet website but could not use entertainment applications such as online games, instant messaging software, in work day.
- o R&D departments are not allowed to access the Internet. But managers of R&D departments have Internet access.

For Internet users who want to access the public website of this enterprise, they are only allowed to access the servers deployed in DMZ.

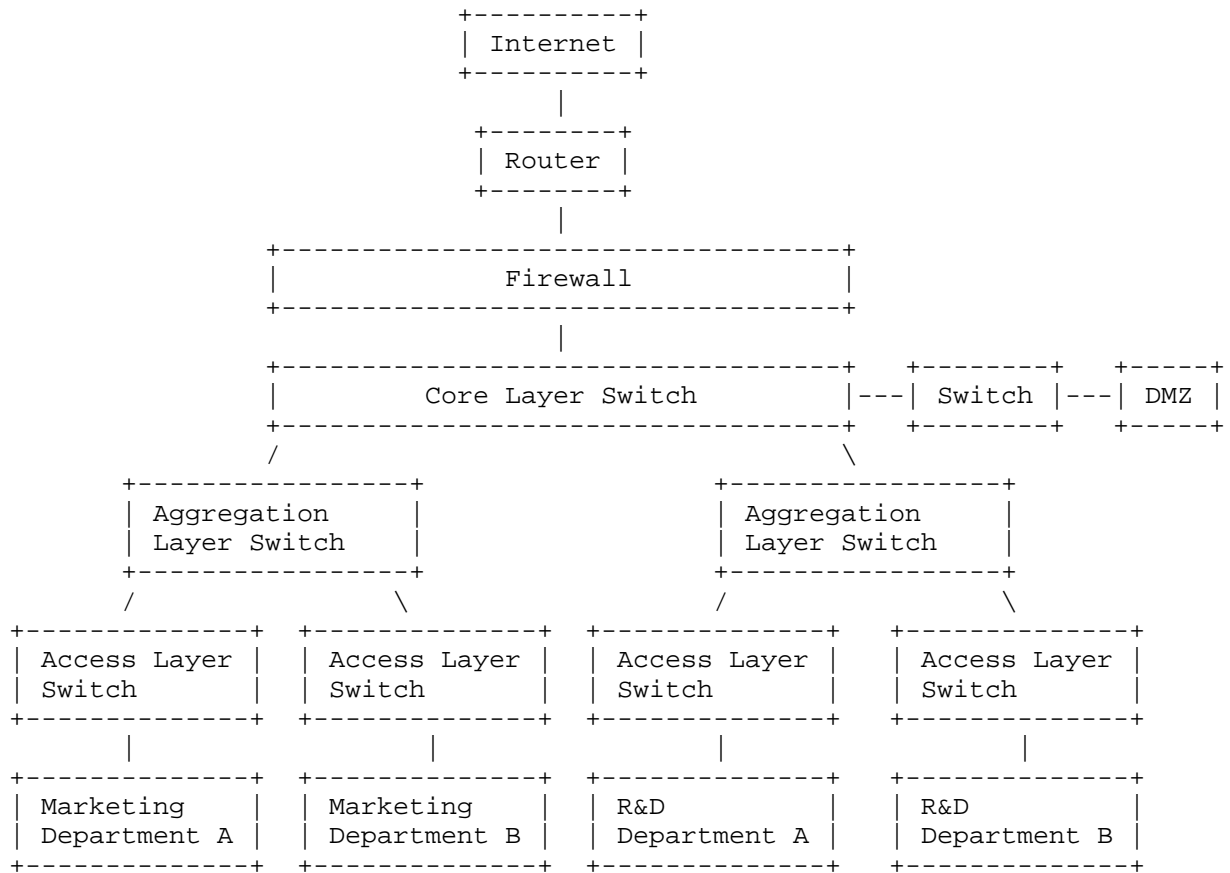


Figure 3: A Typical Architecture of Enterprise Network

To set security policy rules for this scenario, the following policy objects should be created.

Policy Object Name	Description
Marketing_A	User group object for Marketing Department A
Marketing_B	User group object for Marketing Department B
R&D_A	User group object for R&D Department A
R&D_B	User group object for R&D Department B
R&D_Manager	Security group object for managers of R&D Department A and R&D Department B
Entertainment_App	Application group object for all recognized entertainment applications
Server_Address	Address object for servers in DMZ
Web_Service	Service object for HTTP, HTTPS protocols
Work_Day	Schedule object for five week days

B.1. Security Policy Control for Marketing Departments

For traffic from marketing departments to Internet, the following policy objects can be used as conditions to filter traffic.

Policy Objects used in Condition	Action
User Group: Marketing_A, Marketing_B Application Group: Entertainment_App Schedule: Work_Day	Deny
User Group: Marketing_A, Marketing_B Service: Web_Service	Permit

B.2. Security Policy Control for R&D Departments

For traffic from R&D departments to Internet, the following policy objects can be used as conditions to filter traffic.

Policy Objects used in Condition	Action
Security Group: R&D_Manager	Permit
User Group: R&D_A, R&D_B	Deny

B.3. Security Policy Control for Server Access of Internet Users

For traffic from Internet to web servers deployed in DMZ, the following policy objects can be used as conditions to filter traffic.

Policy Objects used in Condition	Action
Address: Server_Address	Permit

Authors' Addresses

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

Email: Frank.xialiang@huawei.com

Qiushi Lin
 Huawei
 Huawei Industrial Base
 Shenzhen, Guangdong 518129
 China

Email: linqiushi@huawei.com

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

L. Xia
J. Strassner
Huawei
C. Basile
PoliTO
D. Lopez
TID
July 3, 2017

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

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on January 5, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
2. Conventions used in this document	5
2.1. Acronyms	5
3. Capability Information Model Design	6
3.1. Design Principles and ECA Policy Model Overview	6
3.2. Relation with the External Information Model	8
3.3. I2NSF Capability Information Model Theory of Operation ...	10
3.3.1. I2NSF Condition Clause Operator Types	11
3.3.2. Capability Selection and Usage	12
3.3.3. Capability Algebra	13
3.4. Initial NSFs Capability Categories	16
3.4.1. Network Security Capabilities	16
3.4.2. Content Security Capabilities	17
3.4.3. Attack Mitigation Capabilities	17
4. Information Sub-Model for Network Security Capabilities	18
4.1. Information Sub-Model for Network Security	18
4.1.1. Network Security Policy Rule Extensions	19
4.1.2. Network Security Policy Rule Operation	20
4.1.3. Network Security Event Sub-Model	22
4.1.4. Network Security Condition Sub-Model	23
4.1.5. Network Security Action Sub-Model	25
4.2. Information Model for I2NSF Capabilities	26
4.3. Information Model for Content Security Capabilities	27
4.4. Information Model for Attack Mitigation Capabilities	28
5. Security Considerations	29
6. IANA Considerations	29
7. Contributors	29
8. References	29
8.1. Normative References	29
8.2. Informative References	30
Appendix A. Network Security Capability Policy Rule Definitions ..	32
A.1. AuthenticationECAPolicyRule Class Definition	32
A.2. AuthorizationECAPolicyRuleClass Definition	34
A.3. AccountingECAPolicyRuleClass Definition	35
A.4. TrafficInspectionECAPolicyRuleClass Definition	37
A.5. ApplyProfileECAPolicyRuleClass Definition	38
A.6. ApplySignatureECAPolicyRuleClass Definition	40
Appendix B. Network Security Event Class Definitions	42
B.1. UserSecurityEvent Class Description	42
B.1.1. The usrSecEventContent Attribute	42
B.1.2. The usrSecEventFormat Attribute	42
B.1.3. The usrSecEventType Attribute	42
B.2. DeviceSecurityEvent Class Description	43
B.2.1. The devSecEventContent Attribute	43
B.2.2. The devSecEventFormat Attribute	43
B.2.3. The devSecEventType Attribute	44
B.2.4. The devSecEventTypeInfo[0..n] Attribute	44
B.2.5. The devSecEventTypeSeverity Attribute	44

Table of Contents (continued)

B.3. SystemSecurityEvent Class Description	44
B.3.1. The sysSecEventContent Attribute	45
B.3.2. The sysSecEventFormat Attribute	45
B.3.3. The sysSecEventType Attribute	45
B.4. TimeSecurityEvent Class Description	45
B.4.1. The timeSecEventPeriodBegin Attribute	46
B.4.2. The timeSecEventPeriodEnd Attribute	46
B.4.3. The timeSecEventTimeZone Attribute	46
Appendix C. Network Security Condition Class Definitions	47
C.1. PacketSecurityCondition	47
C.1.1. PacketSecurityMACCondition	47
C.1.1.1. The pktSecCondMACDest Attribute	47
C.1.1.2. The pktSecCondMACSrc Attribute	47
C.1.1.3. The pktSecCondMAC8021Q Attribute	48
C.1.1.4. The pktSecCondMACEtherType Attribute	48
C.1.1.5. The pktSecCondMACTCI Attribute	48
C.1.2. PacketSecurityIPv4Condition	48
C.1.2.1. The pktSecCondIPv4SrcAddr Attribute	48
C.1.2.2. The pktSecCondIPv4DestAddr Attribute	48
C.1.2.3. The pktSecCondIPv4ProtocolUsed Attribute	48
C.1.2.4. The pktSecCondIPv4DSCP Attribute	48
C.1.2.5. The pktSecCondIPv4ECN Attribute	48
C.1.2.6. The pktSecCondIPv4TotalLength Attribute	49
C.1.2.7. The pktSecCondIPv4TTL Attribute	49
C.1.3. PacketSecurityIPv6Condition	49
C.1.3.1. The pktSecCondIPv6SrcAddr Attribute	49
C.1.3.2. The pktSecCondIPv6DestAddr Attribute	49
C.1.3.3. The pktSecCondIPv6DSCP Attribute	49
C.1.3.4. The pktSecCondIPv6ECN Attribute	49
C.1.3.5. The pktSecCondIPv6FlowLabel Attribute	49
C.1.3.6. The pktSecCondIPv6PayloadLength Attribute	49
C.1.3.7. The pktSecCondIPv6NextHeader Attribute	50
C.1.3.8. The pktSecCondIPv6HopLimit Attribute	50
C.1.4. PacketSecurityTCPCondition	50
C.1.4.1. The pktSecCondTCPSrcPort Attribute	50
C.1.4.2. The pktSecCondTCPDestPort Attribute	50
C.1.4.3. The pktSecCondTCPSeqNum Attribute	50
C.1.4.4. The pktSecCondTCPFlags Attribute	50
C.1.5. PacketSecurityUDPCondition	50
C.1.5.1.1. The pktSecCondUDPSrcPort Attribute	50
C.1.5.1.2. The pktSecCondUDPDestPort Attribute	51
C.1.5.1.3. The pktSecCondUDPLength Attribute	51
C.2. PacketPayloadSecurityCondition	51
C.3. TargetSecurityCondition	51
C.4. UserSecurityCondition	51
C.5. SecurityContextCondition	52
C.6. GenericContextSecurityCondition	52

Table of Contents (continued)

Appendix D. Network Security Action Class Definitions	53
D.1. IngressAction	53
D.2. EgressAction	53
D.3. ApplyProfileAction	53
Appendix E. Geometric Model	54
Authors' Addresses	57

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:

- o 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:

- o 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].
- o 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.
- o 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).

- o 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.
- o 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-supra-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-supra-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:

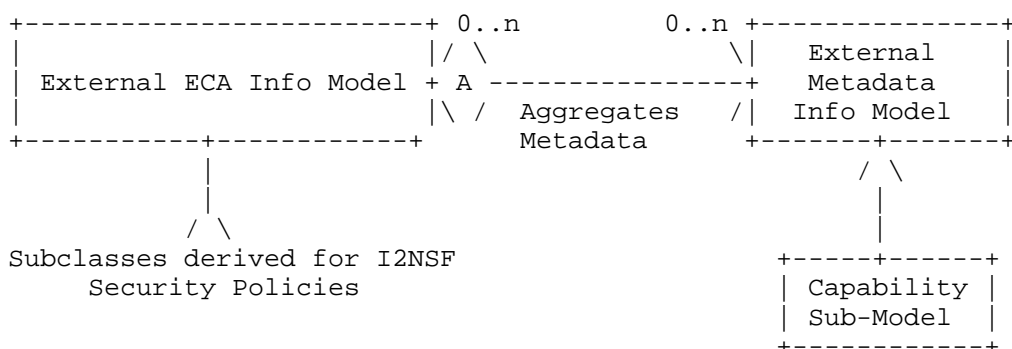


Figure 1. The Overall I2NSF Information Model Design

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-sup-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:
* $cap1+cap2 = \{Ac1 \cup Ac2, Cc1 \cup Cc2, Ec1 \cup Ec2, RSc1, Dc1\}$
- * o Capability subtraction:
* $cap1-cap2 = \{Ac1 \setminus Ac2, Cc1 \setminus Cc2, Ec1 \setminus Ec2, RSc1, Dc1\}$

* 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 NSF's 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:

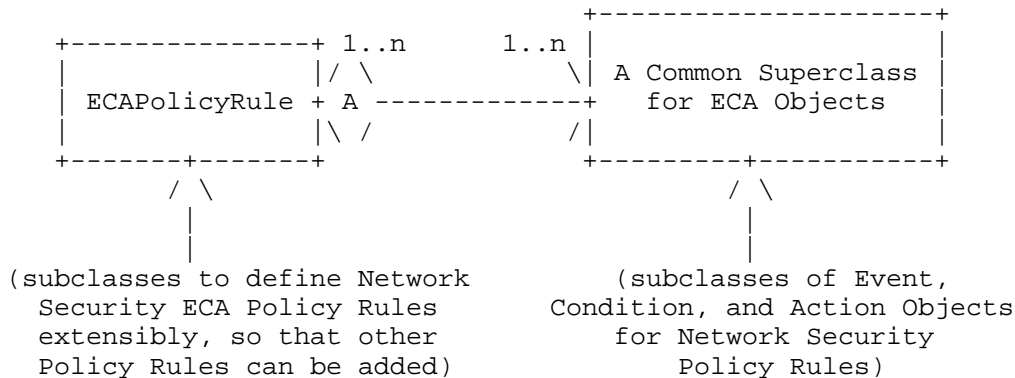


Figure 2. Network Security Information Sub-Model Overview

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.

4.1.1.1. 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.

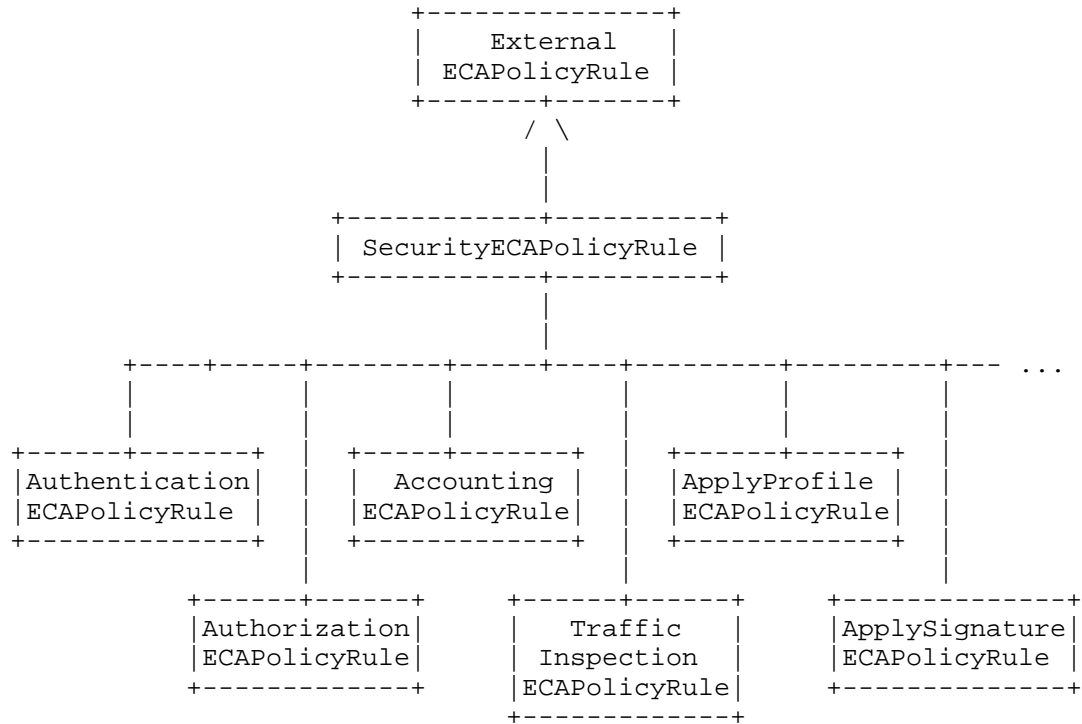


Figure 3. Network Security Info Sub-Model ECAPolicyRule Extensions

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.

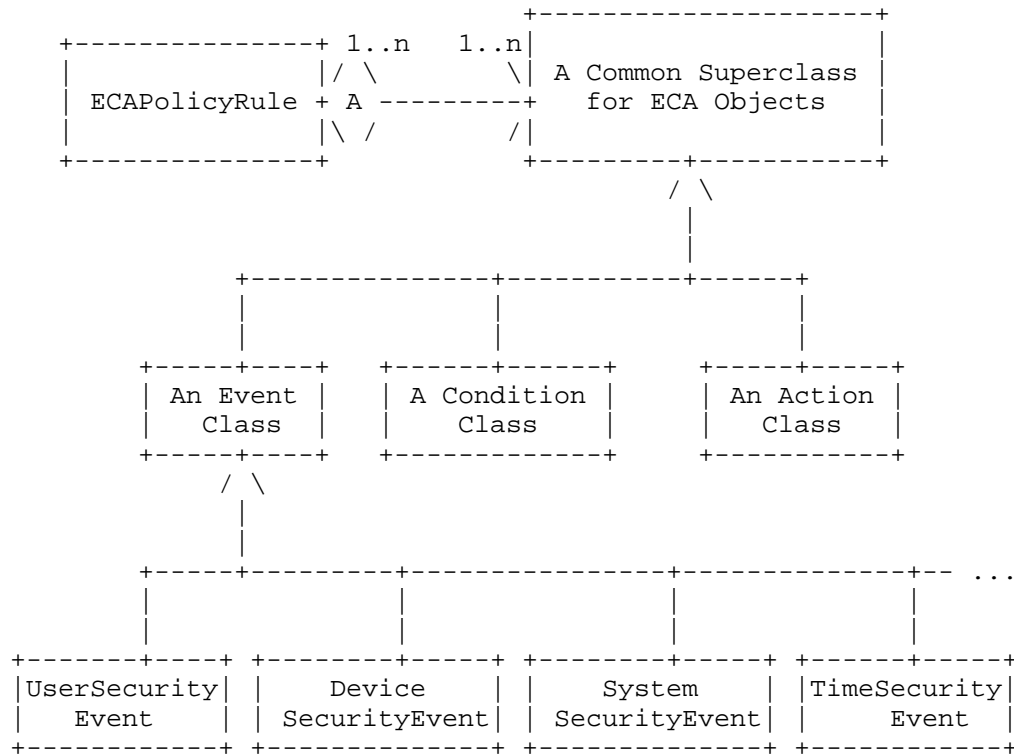


Figure 4. Network Security Info Sub-Model Event Class Extensions

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.

4.1.1.4. 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 the WG regarding this.

*

*** End of Note to WG

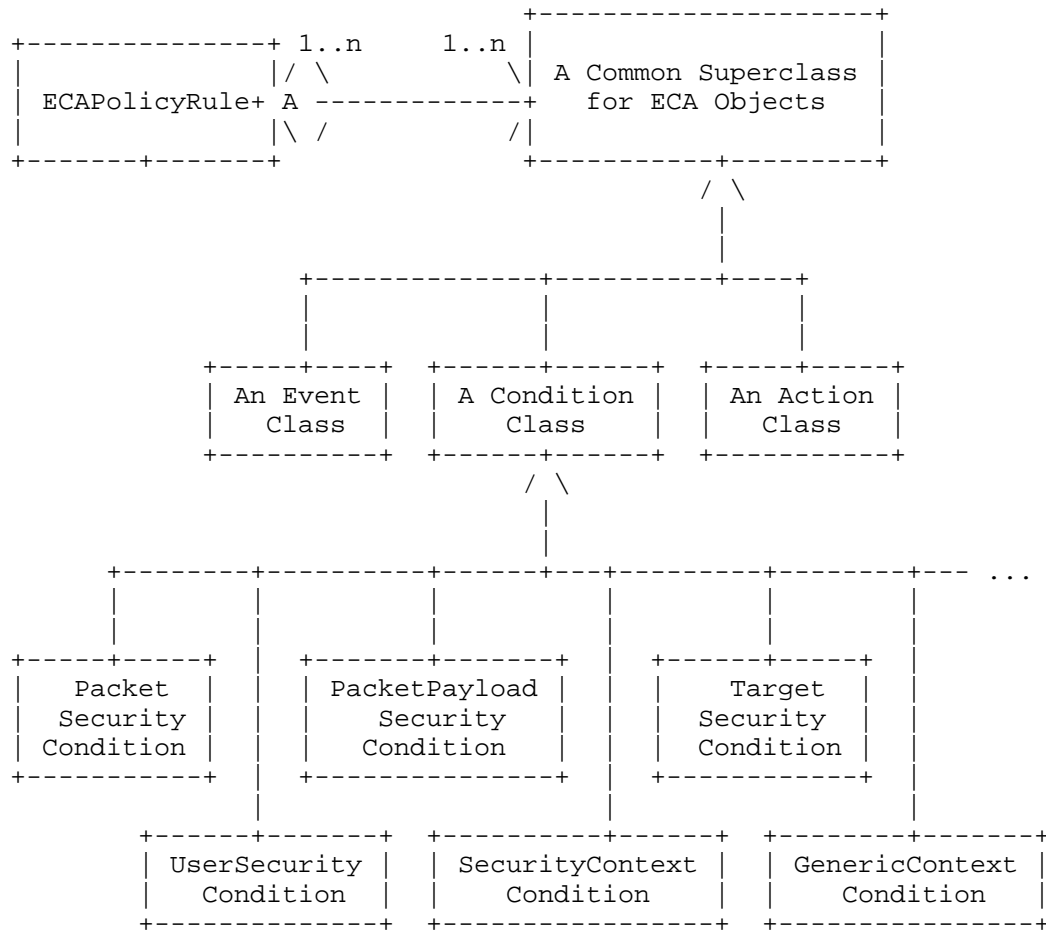


Figure 5. Network Security Info Sub-Model Condition Class Extensions

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.

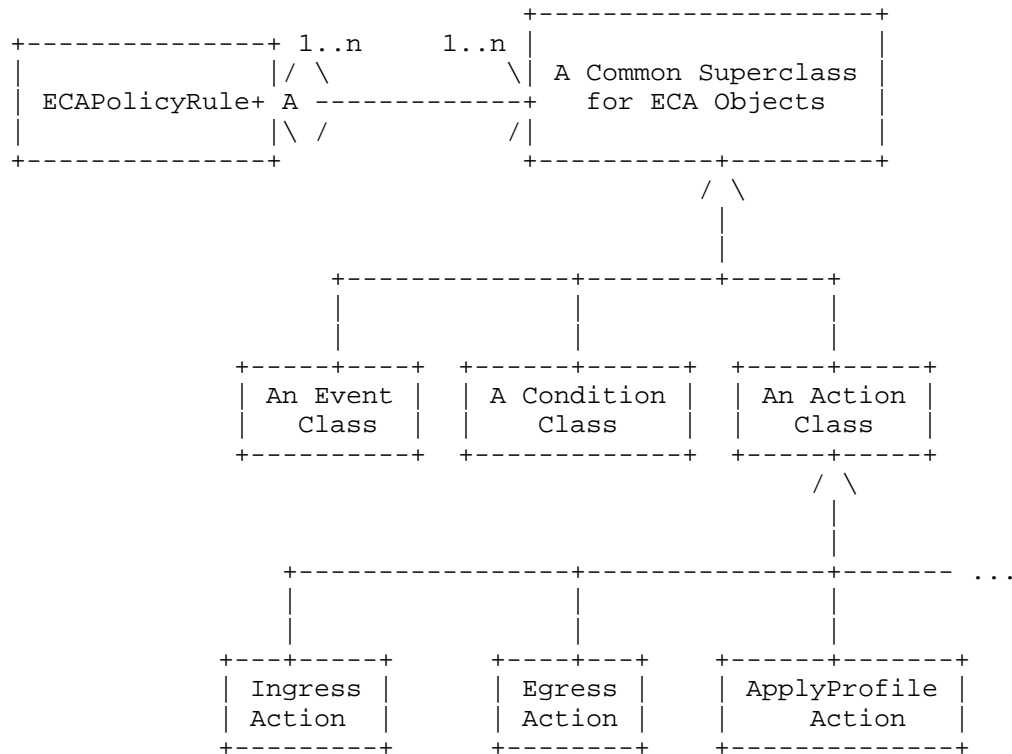


Figure 6. Network Security Info Sub-Model Action Extensions

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.

Figure 8 shows exemplary types of the content security GNSF.

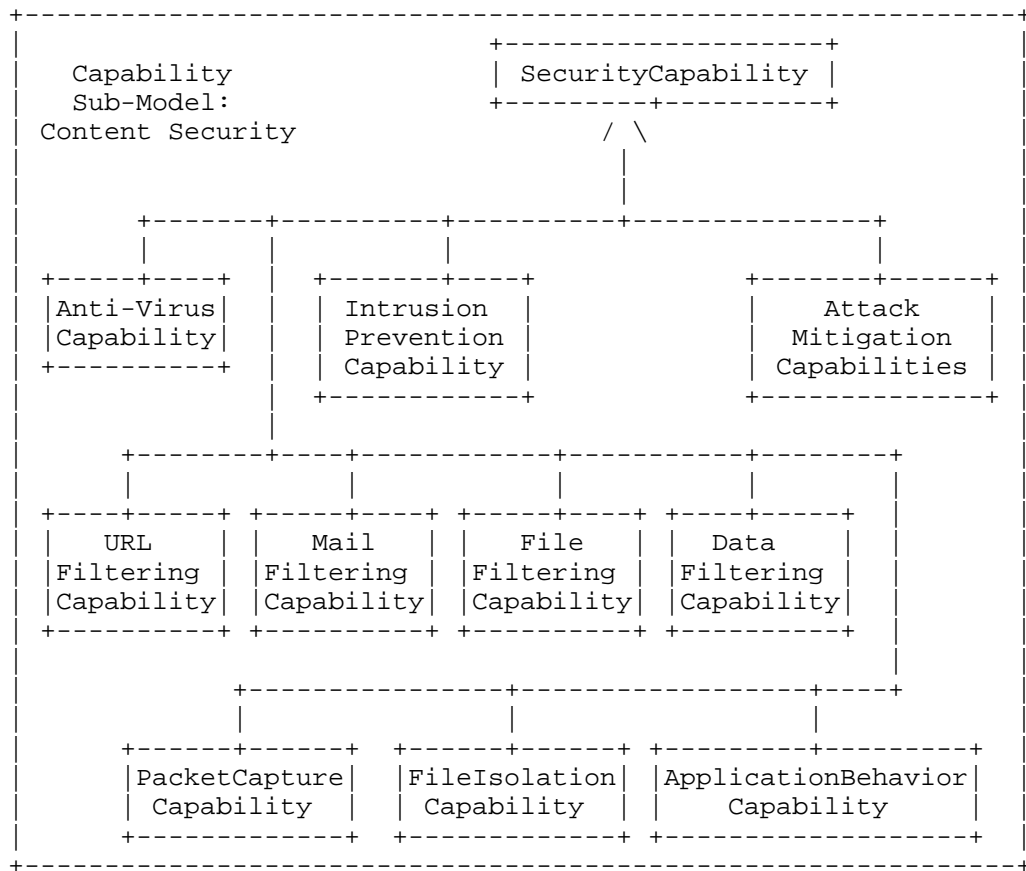


Figure 8. Network Security Capability Information Model

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.

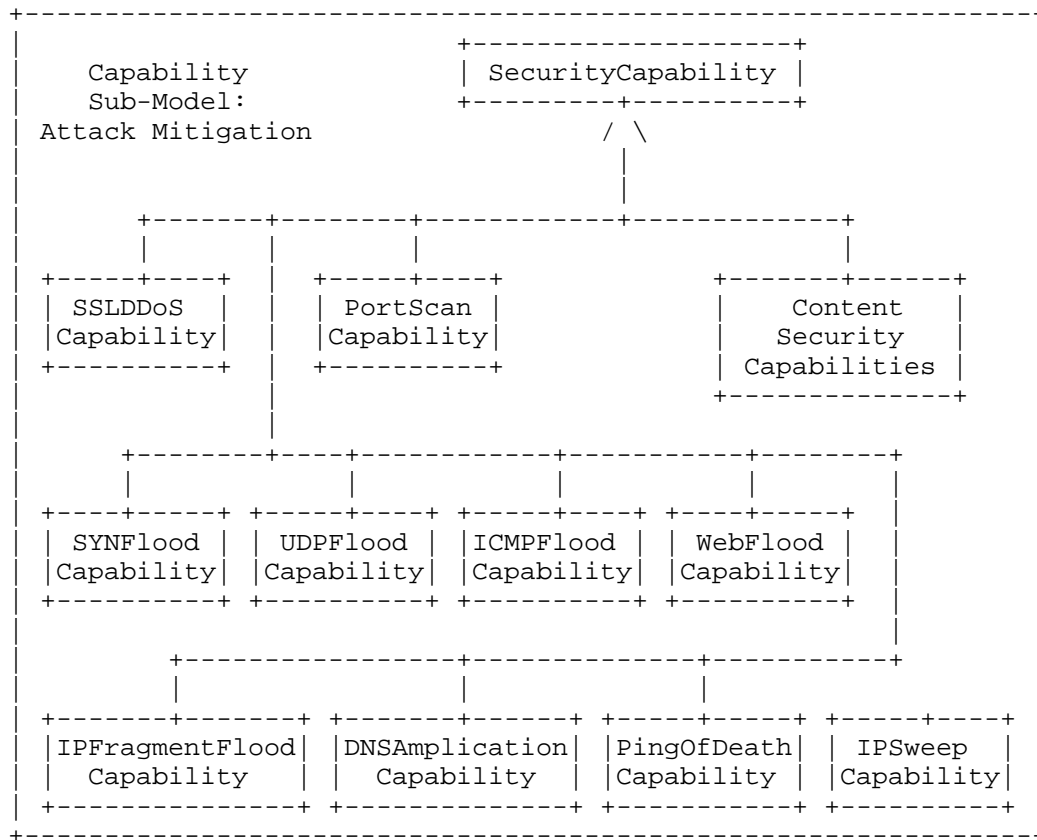


Figure 9. Attack Mitigation Capability Information Model

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 Liroy (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-supra-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-supra-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. Liroy, "Network-Level Access Control Policy Analysis and Transformation", 2012.
- [Bas15]
Basile, C. and Liroy, 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.

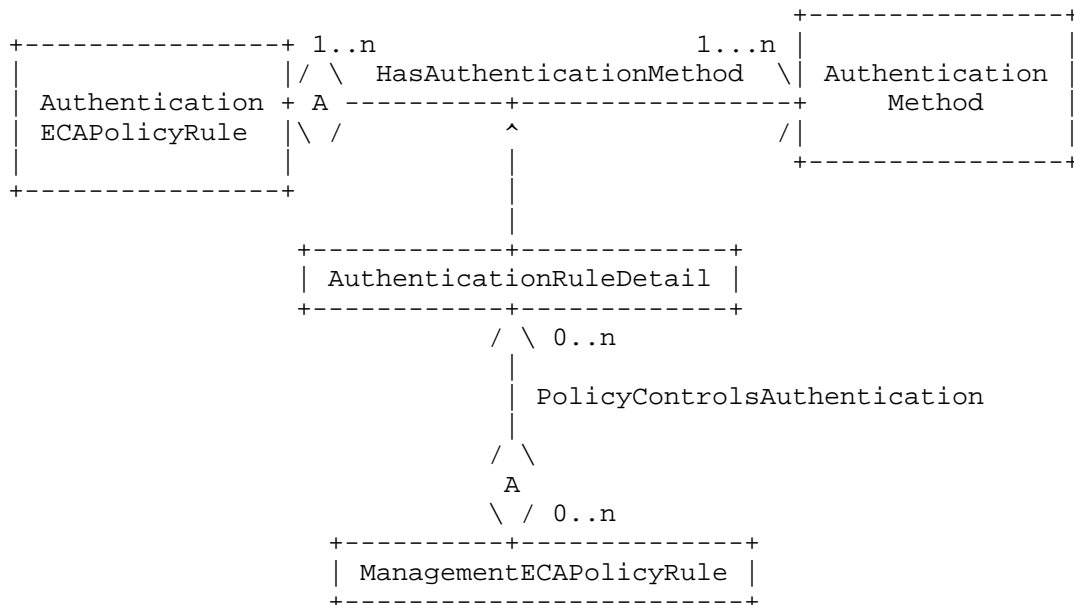


Figure 10. Modeling Authentication Mechanisms

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:

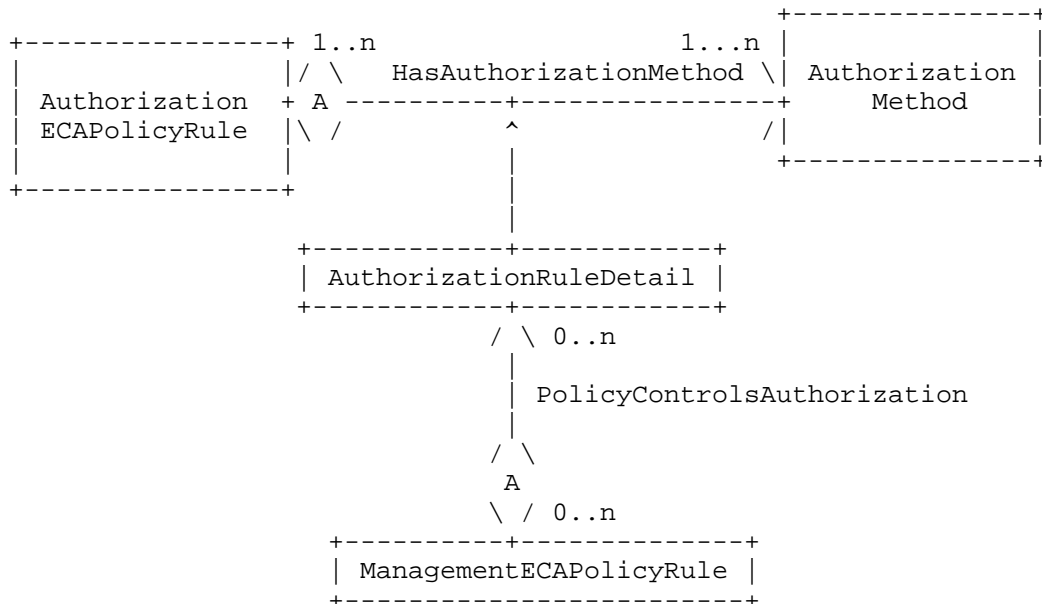


Figure 11. Modeling Authorization Mechanisms

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:

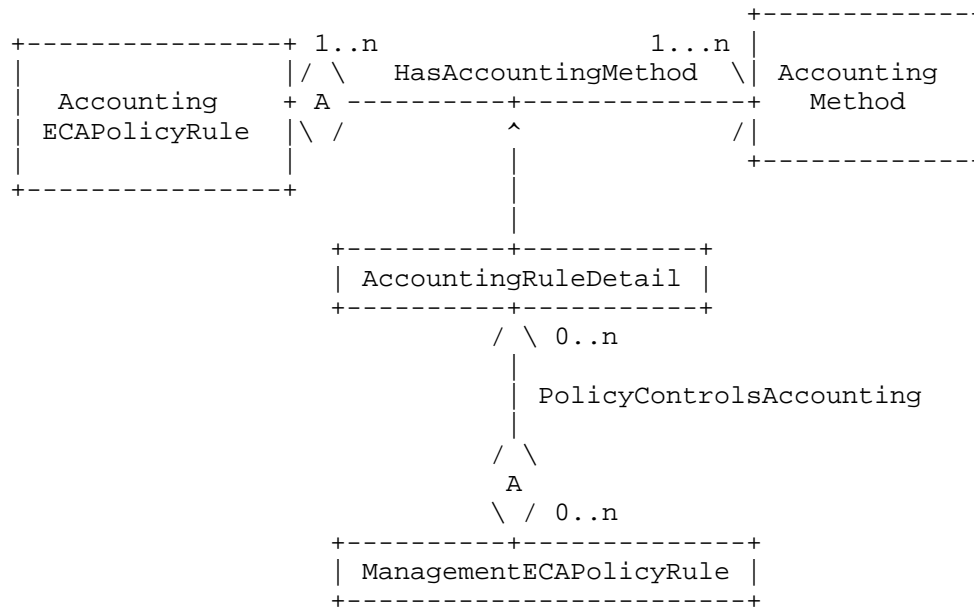


Figure 12. Modeling Accounting Mechanisms

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:

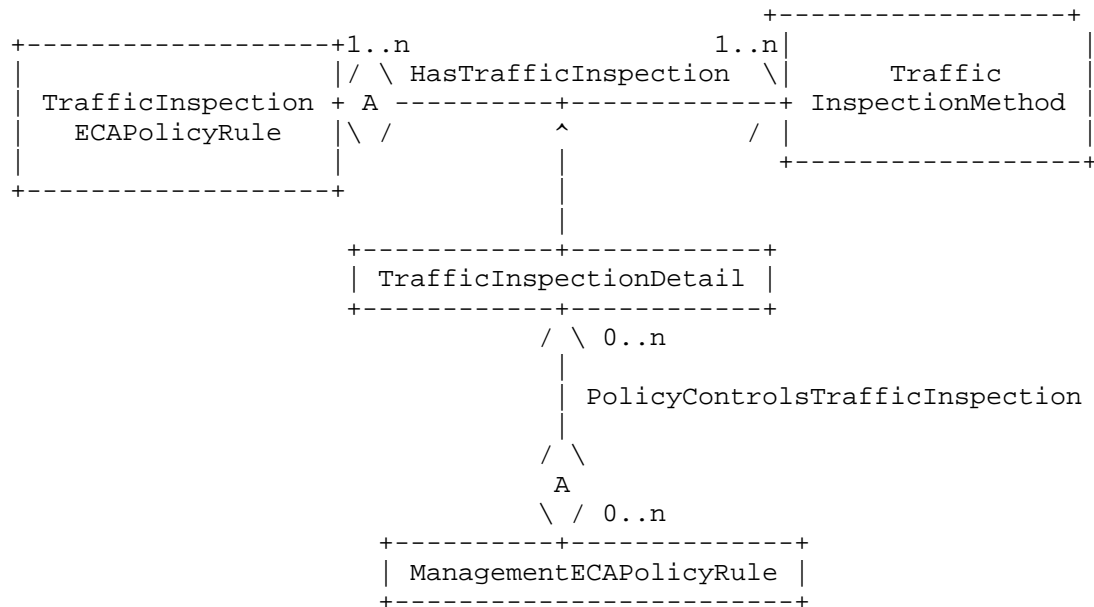


Figure 13. Modeling Traffic Inspection Mechanisms

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:

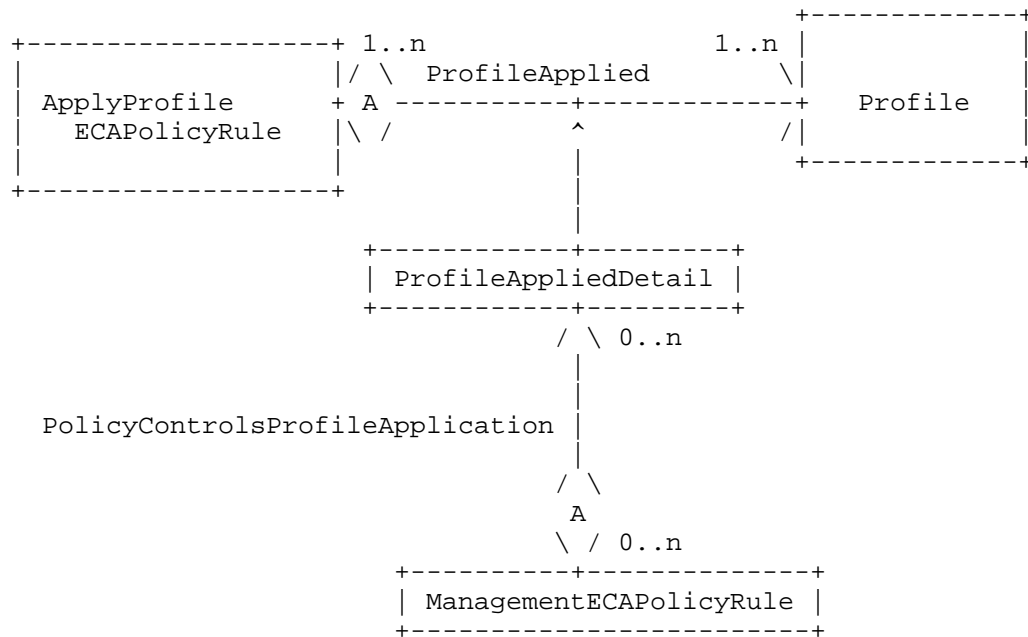


Figure 14. Modeling Profile ApplicationMechanisms

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 `ApplyProfileECAPolicyRule` class, 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. `ApplySignatureECAPolicyRule` Class 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 a URL 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`.

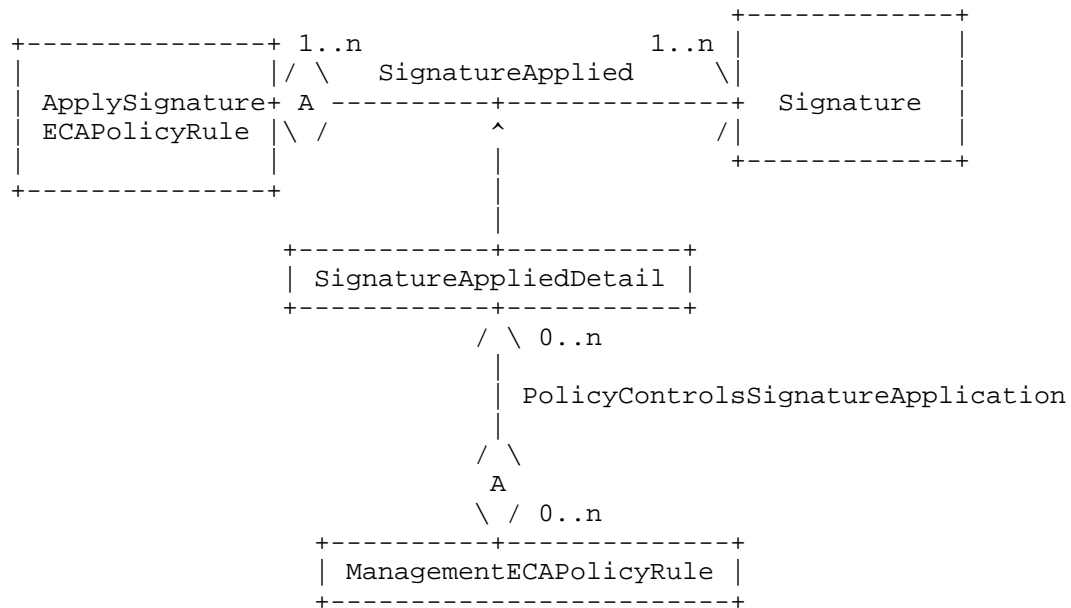


Figure 15. Modeling Sginature Application Mechanisms

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 occurrence, 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.

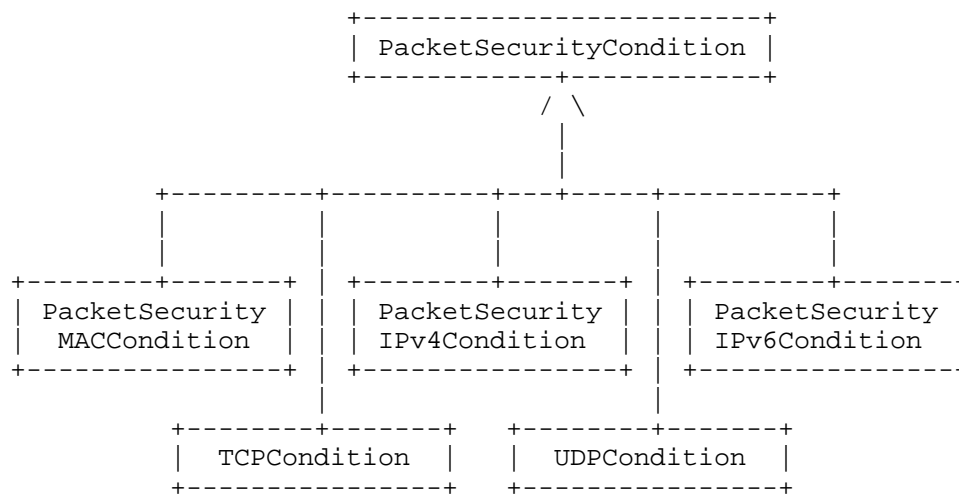


Figure 16. Network Security Info Sub-Model PacketSecurityCondition Class Extensions

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.1.1.1.3. 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.1.1.1.4. 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.1.1.1.5. 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 $A = \{\text{Allow}, \text{Deny}\}$. 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 $s1 \leq S1$ (read as $s1$ subset of or equal to $S1$) and $s2 \leq S2$ ($s2$ 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 $S1, S2, \dots, Sm$ (where m is the number of Selectors available at the security control we want to model), its selection space is:

$$S = S1 \times S2 \times \dots \times Sm$$

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.

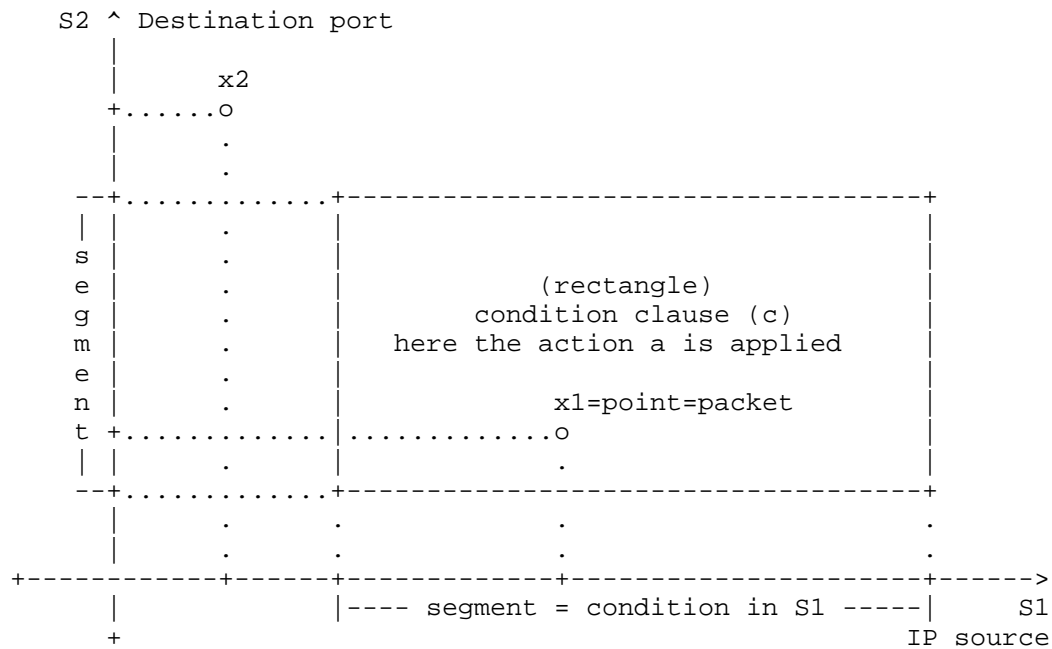


Figure 17: Geometric representation of a rule $r=(c,a)$ that matches x_1 , but does not match x_2 .

Accordingly, the condition clause c is a subset of S :

$$c = s_1 \times s_2 \times \dots \times s_m \leq S_1 \times S_2 \times \dots \times S_m = S$$

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 $c \leq S$ (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 x_1 but not x_2 .

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

$$RS: Pow(R) \rightarrow A$$

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 $e(.)$:

$$e(ri) = (ri, f1(ri), f2(ri), \dots)$$

where $E=\{fj:R \rightarrow Xj\}$ ($j=1,2,\dots$) 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 $p: S \rightarrow A$ 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:

$$p(x)=RS(\text{match}\{R(x)\}).$$

Therefore, in the geometric model, a policy is completely defined by the 4-tuple (R,RS,E,d) : the rule set R , the resolution function RS , the set E of mappings to the external attributes, and the default action d .

Note that, the geometric model also supports ECA paradigms by simply modeling events like an additional selector.

Authors' Addresses

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

John Strassner
Huawei
Email: John.sc.Strassner@huawei.com

Cataldo Basile
Politecnico di Torino
Corso Duca degli Abruzzi, 34
Torino, 10129
Italy
Email: cataldo.basile@polito.it

Diego R. Lopez
Telefonica I+D
Zurbaran, 12
Madrid, 28010
Spain
Phone: +34 913 129 041
Email: diego.r.lopez@telefonica.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 5, 2018

L. Xia
D. Zhang
Huawei
Y. Wu
Aliababa Group
R. Kumar
A. Lohiya
Juniper Networks
H. Birkholz
Fraunhofer SIT
July 04, 2017

An Information Model for the Monitoring of Network Security Functions
(NSF)
draft-zhang-i2nsf-info-model-monitoring-04

Abstract

The Network Security Functions (NSF) NSF-facing interface exists between the Service Provider's management system (or Security Controller) and the NSFs to enforce the security policy provisioning and network security status monitoring. This document focuses on the monitoring part of it and proposes the information model for it.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 5, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	4
2.1. Key Words	4
2.2. Definition of Terms	4
3. Use cases for NSF Monitoring Data	4
4. Classification of NSF Monitoring Data	4
4.1. Retention and Emission	5
4.2. Notifications and Events	6
4.3. Unsolicited Poll and Solicited Push	7
4.4. I2NSF Monitoring Terminology for Retained Information	8
5. Conveyance of NSF Monitoring Information	8
6. Basic Information Model for All Monitoring Data	9
7. Extended Information Model for Monitoring Data	10
7.1. System Alarm	10
7.1.1. Memory Alarm	10
7.1.2. CPU Alarm	11
7.1.3. Disk Alarm	11
7.1.4. Hardware Alarm	11
7.1.5. Interface Alarm	12
7.2. System Events	12
7.2.1. Access Violation	12
7.2.2. Configuration Change	12
7.3. System Log	13
7.3.1. Access Logs	13
7.3.2. Resource Utilization Logs	13
7.3.3. User Activity Logs	14
7.4. System Counters	14
7.4.1. Interface counters	14
7.5. NSF Events	15
7.5.1. DDoS Event	15
7.5.2. Session Table Event	16
7.5.3. Virus Event	16
7.5.4. Intrusion Event	17
7.5.5. Botnet Event	18
7.5.6. Web Attack Event	19
7.6. NSF Logs	20

7.6.1.	DDoS Logs	20
7.6.2.	Virus Logs	20
7.6.3.	Intrusion Logs	21
7.6.4.	Botnet Logs	21
7.6.5.	DPI Logs	21
7.6.6.	Vulnerability Scanning Logs	22
7.6.7.	Web Attack Logs	23
7.7.	NSF Counters	23
7.7.1.	Firewall counters	23
7.7.2.	Policy Hit Counters	24
8.	IANA Considerations	25
9.	Security Considerations	25
10.	References	26
10.1.	Normative References	26
10.2.	Informative References	26
	Acknowledgements	27
	Authors' Addresses	27

1. Introduction

According to [I-D.ietf-i2nsf-terminology], the interface provided by a NSF (e.g., FW, IPS, Anti-DDOS, or Anti-Virus function) to administrative entities (e.g., NMS, security controller) for configuring security function in the NSF and monitoring the NSF is referred to as a "I2NSF NSF-Facing Interface". The monitoring part of it is meant to acquire vital information about the NSF via, e.g. notifications, events, records, counters. The monitoring of the NSF plays a very important role in the overall security framework if done in a timely and comprehensive way. The monitoring information generated by a NSF can very well be an early indication of malicious activity, or anomalous behavior, or a potential sign of denial of service attacks.

This draft proposes a comprehensive NSF monitoring information model that provides visibility into NSFs. This document will not go into the design details of NSF-Facing Interfaces. Instead, this draft is focused on specifying the information and illustrates the methods that enable a NSF to provide the information required in order to be monitored in a scalable and efficient way via the NSF-Facing Interface. The information model for monitoring presented in this document is a complement to the information model for the security policy provisioning part of the NSF-Facing Interface specified in [I-D.xibassnez-i2nsf-capability].

2. Terminology

2.1. Key Words

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

2.2. Definition of Terms

This document uses the terms defined in [I-D.ietf-i2nsf-terminology].

3. Use cases for NSF Monitoring Data

As mentioned earlier, monitoring plays a very critical role in the overall security framework. The monitoring of the NSF provides very valuable information to the security controller in maintaining the provisioned security posture. Besides this, there are various other reasons to monitor the NSFs as listed below:

- o The security administrator can configure a policy that is triggered on a specific event happened in the NSF or the network. The security controller would monitor for the specified event and once it happens, it configures additional security functions as per the policy.
- o The events triggered by NSFs as a result of security policy violation can be used by SIEM to detect any suspicious activity.
- o The events and activity logs from NSFs can be used to build advanced analytics such as behavior and predictive to improve the security posture.
- o 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.
- o The events and activity logs from the NSF can aid in debugging and root cause analysis of an operational issue.
- o The activity logs from the NSF can be used to build historical data for operational and business reasons.

4. Classification of NSF Monitoring Data

In order to maintain a strong security posture, it is not only necessary to configure NSF security policies but also to continuously monitor NSF by consuming acquirable observable information. This

enables security admins to assess what is happening in the network timely. It is not possible to block all the internal and external threats based on static security posture. To achieve this goal, a very dynamic posture with constant visibility is required. This draft defines a set of information elements (and their scope) that can be acquired from NSF and can be used as monitoring information. In essence, these types of monitoring information can be leveraged to support constant visibility on multiple levels of granularity and can be consumed by corresponding functions.

Three basic domains about the monitoring of information originating from a system entity [RFC4949] or a NSF are highlighted in this document.

- o Retention and Emission
- o Notifications and Events
- o 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:

- o a system entity that creates the information
- o a system entity that retains an aggregation of records
- o an I2NSF Component that includes the capabilities of using standardized interfaces provided by other system entities that are not I2NSF Components
- o 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 importance is yet to be assessed. Semantically, a notification is not an I2NSF Event in the context of I2NSF, although they can potentially use the exact same information or data model. In respect to [RFC3877], a Notification is a specific subset of events, because they convey information about "something that happens which may be of interest". In consequence, Notifications can contain information with very low expressiveness or relevance. Hence, additional post-processing functions, such as aggregation, correlation or simple anomaly detection, might have to be employed to satisfy a level of expressiveness that is required for an event-clause of an I2NSF Policy Rule.

It is important to note that the consumer of a notification (the observer) assesses the importance of a notification and not the producer. The producer can include metadata in a notification that supports the observer in assessing the importance (even metadata about severity), but the deciding entity is an I2NSF Agent.

4.3. Unsolicited Poll and Solicited Push

The freshness of the monitored information depends on the acquisition method. Ideally, an I2NSF Agent is accessing every relevant information about the I2NSF Component and is emitting I2NSF Events to a monitoring NSF timely. Publication of events via a pubsub/broker model, peer-2-peer meshes, or static defined channels are only a few examples on how a solicited push of I2NSF Events can be facilitated. The actual mechanic implemented by an I2NSF Component is out of the scope of this document.

Often, corresponding management interfaces have to be queried in intervals or on-demand if required by an I2NSF Policy rule. In some cases, a collection of information has to be conducted via login mechanics provided by a system entity. Accessing records of information via this kind of unsolicited polls can introduce a significant latency in regard to the freshness of the monitored information. The actual definition of intervals implemented by an I2NSF Component is also out of scope of this document.

4.4. I2NSF Monitoring Terminology for Retained Information

Records: Unlike information emitted via notifications and events, records do not require immediate attention from an analyst but may be useful for visibility and retroactive cyber forensic. Depending on the record format, there are different qualities in regard to structure and detail. Records are typically stored in logfiles or databases on a system entity or NSF. Records in the form of logfiles usually include less structures but potentially more detailed information in regard to changes of an system entity's characteristics. In contrast, databases often use more strict schemas or data models, therefore enforcing a better structure, but inhibit storing information that do not match those models ('closed world assumption'). Records can be continuously processed by I2NSF Agents that act as I2NSF Producer and emit events via functions specifically tailored to a certain type of record. Typically, records are information generated by NSF or system entity about their operational and informational data, or various changes in system characteristics, such as user activities, network/traffic status, network activity, etc. They are important for debugging, auditing and security forensic.

Counters: A specific representation of continuous value changes of information elements that potentially occur in high frequency. A prominent example are network interface counters, e.g. PDU amount or byte amount, drop counters, error counters etc. Counters are useful in debugging and visibility into operational behavior of the NSF. An I2NSF Agent that observes the progression of counters can act as an I2NSF Producer and emit events in respect to I2NSF Policy Rules.

5. Conveyance of NSF Monitoring Information

As per the use cases of NSF monitoring data, information needs to be conveyed to various I2NSF Consumers based on requirements imposed by I2NSF Capabilities and work-flows. There are multiple aspects to be considered in regard to emission of monitoring information to requesting parties as listed below:

- o **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.

- o 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.
- o 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.
- o 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.
- o 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.
- o Data Model and Interaction Model for Data in Motion: There are a lot of
- o transport mechanisms such as IP, UDP, TCP. There are also open source implementations for specific set of data such as systems counter, e.g. IPFIX [RFC7011] or NetFlow [RFC3954]. The I2NSF does not mandate any specific method for a given data set, it is up to each implementation.

6. Basic Information Model for All Monitoring Data

As explained in the above section, there is a wealth of data available from the NSF that can be monitored. Firstly, there must be some general information with each monitoring message sent from an NSF that helps consumer in identifying meta data with that message, which are listed as below:

- o message_version: Indicate the version of the data format and is a two-digit decimal numeral starting from 01
- o message_type: Event, Alert, Alarm, Log, Counter, etc
- o time_stamp: Indicate the time when the message is generated
- o vendor_name: The name of the NSF vendor
- o NSF_name: The name (or IP) of the NSF generating the message
- o Module_name: The module name outputting the message
- o Severity: Indicates the level of the logs. There are total eight levels, from 0 to 7. The smaller the numeral is, the higher the severity is.

7. Extended Information Model for Monitoring Data

This section covers the additional information associated with the system messages. The extended information model is only for the structured data such as alarm. Any unstructured data is specified with basic information model only.

[Editors' note]: This section remains the same as -02 version, although the classification of the monitoring data has been changed from -02 version. The new inconsistency will be addressed in next version.

7.1. System Alarm

7.1.1. Memory Alarm

The following information should be included in a Memory Alarm:

- o event_name: 'MEM_USAGE_ALARM'
- o module_name: Indicate the NSF module responsible for generating this alarm
- o usage: specifies the amount of memory used
- o threshold: The threshold triggering the alarm
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: 'The memory usage exceeded the threshold'

7.1.2. CPU Alarm

The following information should be included in a CPU Alarm:

- o event_name: 'CPU_USAGE_ALARM'
- o usage: Specifies the amount of CPU used
- o threshold: The threshold triggering the event
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: 'The CPU usage exceeded the threshold'

7.1.3. Disk Alarm

The following information should be included in a Disk Alarm:

- o event_name: 'DISK_USAGE_ALARM'
- o usage: Specifies the amount of disk space used
- o threshold: The threshold triggering the event
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: 'The disk usage exceeded the threshold'

7.1.4. Hardware Alarm

The following information should be included in a Hardware Alarm:

- o event_name: 'HW_FAILURE_ALARM'
- o component_name: Indicate the HW component responsible for generating this alarm
- o threshold: The threshold triggering the alarm
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: 'The HW component has failed or degraded'

7.1.5. Interface Alarm

The following information should be included in a Interface Alarm:

- o event_name: 'IFNET_STATE_ALARM'
- o interface_Name: The name of interface
- o interface_state: 'UP', 'DOWN', 'CONGESTED'
- o threshold: The threshold triggering the event
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: 'Current interface state'

7.2. System Events

7.2.1. Access Violation

The following information should be included in this event:

- o event_name: 'ACCESS_DENIED'
- o user: Name of a user
- o group: Group to which a user belongs
- o login_ip_address: Login IP address of a user
- o authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- o message: 'access denied'

7.2.2. Configuration Change

The following information should be included in this event:

- o event_name: 'CONFIG_CHANGE'
- o user: Name of a user
- o group: Group to which a user belongs
- o login_ip_address: Login IP address of a user

- o authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- o message: 'Configuration modified'

7.3. System Log

7.3.1. Access Logs

Access logs record administrators' login, logout, and operations on the device. By analyzing them, security vulnerabilities can be identified. The following information should be included in operation report:

- o Administrator: Administrator that operates on the device
- o login_ip_address: IP address used by an administrator to log in
- o login_mode: Specifies the administrator logs in mode e.g. root, user
- o operation_type: The operation type that the administrator execute, e.g., login, logout, configuration, etc
- o result: Command execution result
- o 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:

- o system_status: The current system's running status
- o CPU_usage: Specifies the CPU usage
- o memory_usage: Specifies the memory usage
- o disk_usage: Specifies the disk usage
- o disk_left: Specifies the available disk space left
- o session_number: Specifies total concurrent sessions
- o process_number: Specifies total number of system processes

- o `in_traffic_rate`: The total inbound traffic rate in pps
- o `out_traffic_rate`: The total outbound traffic rate in pps
- o `in_traffic_speed`: The total inbound traffic speed in bps
- o `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.

- o `user`: Name of a user
- o `group`: Group to which a user belongs
- o `login_ip_address`: Login IP address of a user
- o `authentication_mode`: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- o `access_mode`: User access mode. e.g., PPP, SVN, LOCAL
- o `online_duration`: Online duration
- o `lockout_duration`: Lockout duration
- o `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
- o `cause`: Cause of a failed user activity

7.4. System Counters

7.4.1. Interface counters

Interface counters provide visibility into traffic into and out of NSF, bandwidth usage.

- o `interface_name`: Network interface name configured in NSF
- o `in_total_traffic_pkts`: Total inbound packets

- o out_total_traffic_pkts: Total outbound packets
- o in_total_traffic_bytes: Total inbound bytes
- o out_total_traffic_bytes: Total outbound bytes
- o in_drop_traffic_pkts: Total inbound drop packets
- o out_drop_traffic_pkts: Total outbound drop packets
- o in_drop_traffic_bytes: Total inbound drop bytes
- o out_drop_traffic_bytes: Total outbound drop bytes
- o in_traffic_ave_rate: Inbound traffic average rate in pps
- o in_traffic_peak_rate: Inbound traffic peak rate in pps
- o in_traffic_ave_speed: Inbound traffic average speed in bps
- o in_traffic_peak_speed: Inbound traffic peak speed in bps
- o out_traffic_ave_rate: Outbound traffic average rate in pps
- o out_traffic_peak_rate: Outbound traffic peak rate in pps
- o out_traffic_ave_speed: Outbound traffic average speed in bps
- o out_traffic_peak_speed: Outbound traffic peak speed in bps.

7.5. NSF Events

7.5.1. DDoS Event

The following information should be included in a DDoS Event:

- o event_name: 'SEC_EVENT_DDoS'
- o 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.
- o dst_ip: The IP address of a victim under attack
- o dst_port: The port numbers that the attrack traffic aims at.
- o start_time: The time stamp indicating when the attack started

- o end_time: The time stamp indicating when the attack ended. If the attack is still undergoing when sending out the alarm, this field can be empty.
- o attack_rate: The PPS of attack traffic
- o attack_speed: the bps of attack traffic
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches.

7.5.2. Session Table Event

The following information should be included in a Session Table Event:

- o event_name: 'SESSION_USAGE_HIGH'
- o current: The number of concurrent sessions
- o max: The maximum number of sessions that the session table can support
- o threshold: The threshold triggering the event
- o message: 'The number of session table exceeded the threshold'

7.5.3. Virus Event

The following information should be included in a Virus Event:

- o event_Name: 'SEC_EVENT_VIRUS'
- o virus_type: Type of the virus, e.g., trojan, worm, macro Virus type
- o virus_name
- o dst_ip: The destination IP address of the packet where the virus is found
- o src_ip: The source IP address of the packet where the virus is found
- o src_port: The source port of the packet where the virus is found

- o `dst_port`: The destination port of the packet where the virus is found
- o `src_zone`: The source security zone of the packet where the virus is found
- o `dst_zone`: The destination security zone of the packet where the virus is found
- o `file_type`: The type of the file where the virus is hided within
- o `file_name`: The name of the file where the virus is hided within
- o `virus_info`: The brief introduction of virus
- o `raw_info`: The information describing the packet triggering the event.
- o `rule_id`: The ID of the rule being triggered
- o `rule_name`: The name of the rule being triggered
- o `profile`: Security profile that traffic matches.

7.5.4. Intrusion Event

The following information should be included in a Intrusion Event:

- o `event_name`: The name of event: 'SEC_EVENT_Intrusion'
- o `sub_attack_type`: Attack type, e.g., brutal force, buffer overflow
- o `src_ip`: The source IP address of the packet
- o `dst_ip`: The destination IP address of the packet
- o `src_port`:The source port number of the packet
- o `dst_port`: The destination port number of the packet
- o `src_zone`: The source security zone of the packet
- o `dst_zone`: The destination security zone of the packet
- o `protocol`: The employed transport layer protocol, e.g.,TCP, UDP
- o `app`: The employed application layer protocol, e.g.,HTTP, FTP

- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches
- o intrusion_info: Simple description of intrusion
- o 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:

- o event_name: the name of event: 'SEC_EVENT_Botnet'
- o botnet_name: The name of the detected botnet
- o src_ip: The source IP address of the packet
- o dst_ip: The destination IP address of the packet
- o src_port: The source port number of the packet
- o dst_port: The destination port number of the packet
- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o protocol: The employed transport layer protocol, e.g., TCP, UDP
- o app: The employed application layer protocol, e.g., HTTP, FTP
- o 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

- o botnet_info: Simple description of Botnet
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches
- o 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:

- o event_name: the name of event: 'SEC_EVENT_WebAttack'
- o sub_attack_type: Concret web attack type, e.g., sql injection, command injection, XSS, CSRF
- o src_ip: The source IP address of the packet
- o dst_ip: The destination IP address of the packet
- o src_port: The source port number of the packet
- o dst_port: The destination port number of the packet
- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o req_method: The method of requirement. For instance, 'PUT' or 'GET' in HTTP
- o req_url: Requested URL
- o url_category: Matched URL category
- o filtering_type: URL filtering type, e.g., Blacklist, Whitelist, User-Defined, Predefined, Malicious Category, Unknown
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered

- o profile: Security profile that traffic matches.

7.6. NSF Logs

7.6.1. DDoS Logs

Besides the fields in an DDoS Alarm, the following information should be included in a DDoS Logs:

- o attack_type: DDoS
- o attack_ave_rate: The average pps of the attack traffic within the recorded time
- o attack_ave_speed: The average bps of the attack traffic within the recorded time
- o attack_pkt_num: The number attack packets within the recorded time
- o 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.
- o 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:

- o attack_type: Virus
- o protocol: The transport layer protocol
- o app: The name of the application layer protocol
- o times: The time of detecting the virus
- o action: The actions dealing with the virus, e.g., alert, block
- o 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:

- o attack_type: Intrusion
- o times: The times of intrusions happened in the recorded time
- o os: The OS that the intrusion will affect, e.g., all, android, ios, unix, windows
- o action: The actions dealing with the intrusions, e.g., e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service
- o attack_rate: NUM the pps of attack traffic
- o 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:

- o attack_type: Botnet
- o botnet_pkt_num: The number of the packets sent to or from the detected botnet
- o action: The actions dealing with the detected packets, e.g., Allow, Alert, Block, Discard, Declare, Block-ip, Block-service, etc
- o 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.

- o type: DPI action types. e.g., File Blocking, Data Filtering, Application Behavior Control
- o file_name: The file name

- o file_type: The file type
- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic
- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o action: Action defined in the file blocking rule, data filtering rule, or application behavior control rule that traffic matches.

7.6.6. Vulnerability 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:

- o victim_ip: IP address of the victim host which has vulnerabilities
- o vulnerability_id: The vulnerability id
- o vulnerability_level: The vulnerability level. e.g., high, middle, low
- o OS: The operating system of the victim host
- o service: The service which has vulnerability in the victim host

- o protocol: The protocol type. e.g., TCP, UDP
- o port: The port number
- o vulnerability_info: The information about the vulnerability
- o 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:

- o attack_type: Web Attack
- o rsp_code: Response code
- o req_clientapp: The client application
- o req_cookies: Cookies
- o req_host: The domain name of the requested host
- o raw_info: The information describing the packet triggering the event.

7.7. NSF Counters

7.7.1. Firewall counters

Firewall counters provide visibility into traffic signatures, bandwidth usage, and how the configured security and bandwidth policies have been applied.

- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic

- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic
- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o in_interface: Inbound interface of traffic
- o out_interface: Outbound interface of traffic
- o total_traffic: Total traffic volume
- o in_traffic_ave_rate: Inbound traffic average rate in pps
- o in_traffic_peak_rate: Inbound traffic peak rate in pps
- o in_traffic_ave_speed: Inbound traffic average speed in bps
- o in_traffic_peak_speed: Inbound traffic peak speed in bps
- o out_traffic_ave_rate: Outbound traffic average rate in pps
- o out_traffic_peak_rate: Outbound traffic peak rate in pps
- o out_traffic_ave_speed: Outbound traffic average speed in bps
- o 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.

- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic

- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic
- o policy_id: Security policy id that traffic matches
- o 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 undesirable 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 appropriately to provide the above security functions.

In addition, to defend against the DDoS attack caused by a lot of NSFs sending massive monitoring information to the security controller, the rate limiting or similar mechanisms should be considered in NSF and security controller, whether in advance or just in the process of DDoS attack.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <<http://www.rfc-editor.org/info/rfc3877>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<http://www.rfc-editor.org/info/rfc4949>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<http://www.rfc-editor.org/info/rfc5424>>.
- [RFC6587] Gerhards, R. and C. Lonvick, "Transmission of Syslog Messages over TCP", RFC 6587, DOI 10.17487/RFC6587, April 2012, <<http://www.rfc-editor.org/info/rfc6587>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<http://www.rfc-editor.org/info/rfc7011>>.

10.2. Informative References

- [I-D.ietf-i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-03 (work in progress), March 2017.
- [I-D.xibassnez-i2nsf-capability] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-xibassnez-i2nsf-capability-02 (work in progress), July 2017.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <<http://www.rfc-editor.org/info/rfc3954>>.

Acknowledgements

Authors' Addresses

Liang Xia
Huawei

Email: frank.xialiang@huawei.com

Dacheng Zhang
Huawei

Email: dacheng.zhang@huawei.com

Yi Wu
Aliababa Group

Email: anren.wy@alibaba-inc.com

Rakesh Kumar
Juniper Networks

Email: rkkumar@juniper.net

Anil Lohiya
Juniper Networks

Email: alohiya@juniper.net

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de