I2NSF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

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

March 11, 2019

I2NSF Consumer-Facing Interface YANG Data Model
draft-ietf-i2nsf-consumer-facing-interface-dm-03

Abstract

This document describes an information model and 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 information model defines various managed objects and relationship among these objects needed to build the interface. The information model is organized based on the "Event-condition-Event" (ECA) policy model defined by a capability information model for Interface to Network Security Functions (I2NSF)[draft-ietf-i2nsf-capability], and the data model is defined for enabling different users of a given I2NSF system to define, manage, and monitor security policies for specific flows within an administrative domain.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 12, 2019.

Table of Contents

1. Introduction

In I2NSF framework, each vendor can register their NSFs using the Vendor Management System (VMS). Assuming that vendors also provide the front-end web applications registered with the I2NSF provider, the Consumer-Facing Interface is required because the web applications developed by each vendor need to have a standard interface specifying the data types used when I2NSF user and security controller communicate using this interface. Therefore, this document specifies the required information, their data types, and encoding schemes so that any data (security policy) transferred through the Consumer-Facing Interface can easily be translated by the security controller into low-level security policies. The translated policies are delivered to Network Security Functions (NSFs) according to their respective security capabilities for securiy enforcement.

The Consumer-Facing Interface would be built using a set of objects, with each object capturing a unique set of information from Security Admin (i.e., I2NSF User [RFC8329]) needed to express a Security Policy. An object may have relationship with various other objects to express a complete set of requirement. An information model captures the managed objects and relationship among these objects. The information model proposed in this document is structured in accordance with the "Event-Condition-Event" (ECA) policy model.

An NSF Capability model is proposed in [draft-ietf-i2nsf-capability] as the basic model for both the NSF-Facing interface and Consumer-Facing Interface security policy model of this document.

[RFC3444] explains differences between an information and data model. This document use the guidelines in [RFC3444] to define both the information and data model for Consumer-Facing Interface. Figure 1 shows a high-level abstraction of Consumer-Facing Interface. A data model, which represents an implementation of the information model in a specific data representation language, is also defined in the later sections of this document (see section 10).

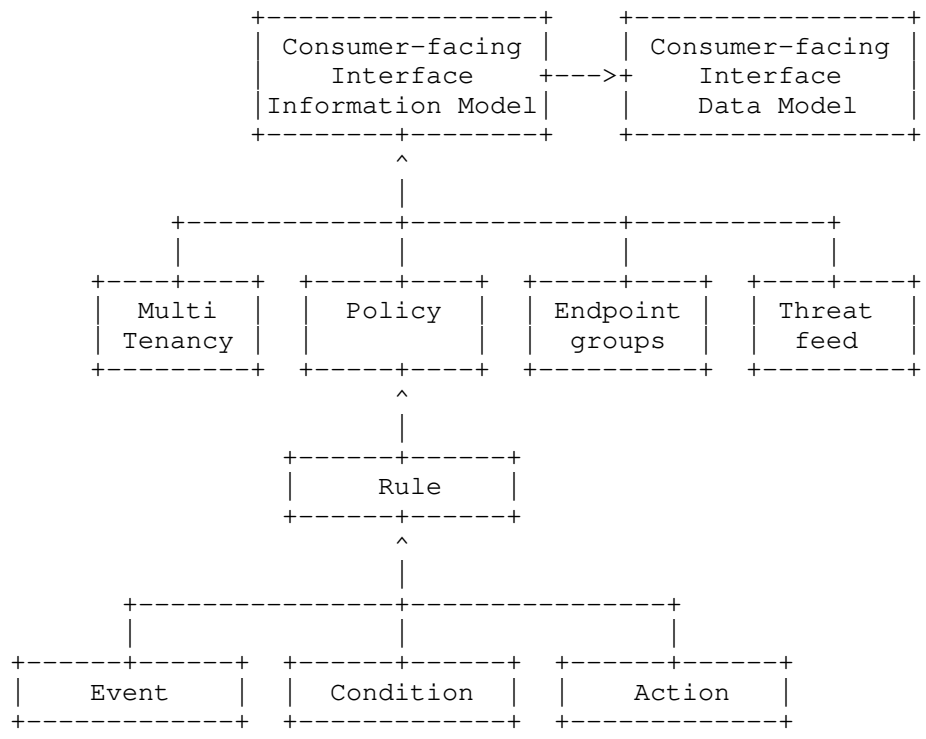

Figure 1: Diagram for High-level Abstraction of Consumer-Facing Interface

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] RFC8174 [RFC8174].

3. Terminology

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

This document follows the guidelines of [RFC6087], uses the common YANG types defined in [RFC6991], and adopts the Network Management Datastore Architecture (NMDA). The meaning of the symbols in tree diagrams is defined in [RFC8340].

4. Information Model for Policy

A Policy object represents a mechanism to express a Security Policy by Security Admin (i.e., I2NSF User) using Consumer-Facing Interface toward Security Controller; the policy would be enforced on an NSF. Figure 2 shows the XML instance of the Policy object. The Policy object SHALL have following information:

Name: This field identifies the name of this object.

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

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.

```
+--rw policy
   +--rw policy-name?              string
   +--rw rule* [rule-name]
   +--rw event
   +--rw condition
   +--rw action
```

Figure 2: Policy YANG Data tree

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

network, which will be specified as the sub-model of the policy model in the subsequent sections. Figure 3 shows the XML instance of the Rule object. The rule object SHALL have the following information:

Name: This field identifies the name of this object.

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

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

Condition: This field contains all the checking conditions to apply to the objective traffic. See details in Section 4.2.

Action: This field identifies the action taken when a rule is matched. There is always an implicit action to drop traffic if no rule is matched for a traffic type. See details in Section 4.3.

Owner: This field contains the onwer of the rule. For example, the person who created it, and eligible for modifying it.

```
     +--rw rule* [rule-name]
        +--rw rule-name    string
        +--rw date?        yang:date-and-time
        +--rw event* [name]
        +--rw condition
        +--rw action
        +--rw owner?       string
```

Figure 3: YANG Data tree for Rule

4.1. Event Sub-Model

The Event Object contains information related to scheduling a Rule. The Rule could be activated based on a time calendar or security event including threat level changes. Figure 4 shows the XML instance of the Event object. Event object SHALL have following information:

Name: This field identifies the name of this object.

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

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

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

```
  +--rw event
     +--rw name?                      string
     +--rw date?                      yang:date-and-time
     +--rw event-type                 enumeration
     +--rw time-information
        +--rw time
        |  +--rw begin-time           begin-time-type
        |  +--rw end-time             end-time-type
        +--rw recursive
           +--rw recur                boolean
           +--rw recursive-type?      enumeration
```

Figure 4: Event sub-model YANG data tree

4.2. Condition Sub-Model

This object represents Conditions that Security Admin wants to apply the checking on the traffic in order to determine whether the set of actions in the Rule can be executed or not. The condition sub-model consists of 3 different types of three containers each representing different cases, such as general firewall and ddos-mitigation cases, and a case when the condition is based on the payload strings of packets. Each containers have source-target and destination-target to represent the source and destination for each case. Figure 5 shows the XML instance of the Condition object.The Condition submodel SHALL have following information:

Firewall-condition: This field represents the general firewall case, where a security admin can set up firewall conditions using the information present in this field. The source and destination is represented as source-target and destination-target, each referring to the IP-address-based groups defined in the endpoint-group.

DDoS-condition: This field represents the condition for DDoS mitigation, where a security admin can set up DDoS mitigation conditions using the information present in this field. The source and destination is represented as

source-target and destination-target, each referring to the device-groups defined and registered in the endpoint-group.

Custom-condition: This field contains the payload string information. This information is useful when security rule condition is based on the string contents of incoming or outgoing packets. The source and destination is represented as source-target and destination-target, each referring to the payload-groups defined and registered in the endpoint-group.

```
      +--rw condition
        +--rw firewall-condition
        |  +--rw source-target
        |  |  +--rw src-target?   -> /policy
        |  |                         /endpoint-group
        |  |                         /user-group
        |  |                         /name
        |  +--rw destination-target
        |  |  +--rw dest-target*  -> /policy
        |  |                         /endpoint-group
        |  |                         /user-group
        |  |                         /name
        |  +--rw ddos-condition
        |  |  +--rw source-target
        |  |  |  +--rw src-target*   -> /policy
        |  |  |                         /endpoint-group
        |  |  |                         /device-group
        |  |  |                         /name
        |  |  +--rw destination-target
        |  |  |  +--rw dest-target*  -> /policy
        |  |  |                         /endpoint-group
        |  |  |                         /device-group
        |  |  |                         /name
        |  |  +--rw rate-limit
        |  |     +--rw packet-per-second?   uint8
        |  +--rw custom-condition
        |  |  +--rw source-target
        |  |  |  +--rw src-target*   -> /policy
        |  |  |                         /threat-prevention
        |  |  |                         /payload-content
        |  |  |                         /name
        |  |  +--rw destination-target
        |  |  |  +--rw dest-target?  -> /policy
        |  |  |                         /threat-prevention
        |  |  |                         /payload-content
        |  |  |                         /name
        |  +--rw threat-feed-condition
```

```
                +--rw source-target
                |  +--rw src-target*   -> /policy
                |                         /threat-prevention
                |                         /threat-feed-list
                |                         /name
                +--rw destination-target
                   +--rw dest-target?  -> /policy
                                          /threat-prevention
                                          /threat-feed-list
                                          /name
```

Figure 5: Condition sub-model YANG data tree

4.3. Action Sub-Model

This object represents actions that Security Admin wants to perform based on certain traffic class. Figure 6 shows the XML instance of the Action object. The Action object SHALL have following information:

Name: This field identifies the name of this object.

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

Action: This field identifies the action when a rule is matched by an NSF. The action could be one of "PASS", "DROP", "ALERT", "MIRROR", and "LOG".

```
          +--rw action
             +--rw name            string
             +--rw date            yang:date-and-time
             +--rw action          string
```

Figure 6: Action sub-model YANG data tree

5. Information Model for Multi-Tenancy

Multi-tenancy is an important aspect of any application that enables multiple administrative domains in order to manage application resources. An Enterprise organization may have multiple tenants or departments such as Human Resources (HR), Finance, and Legal, with each tenant having a need to manage their own Security Policies. In a Service Provider, a tenant could represent a Customer that wants to manage its own Security Policies. There are multiple managed objects that constitute multi-tenancy aspects as shown in Figure 7. This

section lists these objects and any relationship among these objects. Below diagram shows an example of multi-tenancy in an Enterprise domain.

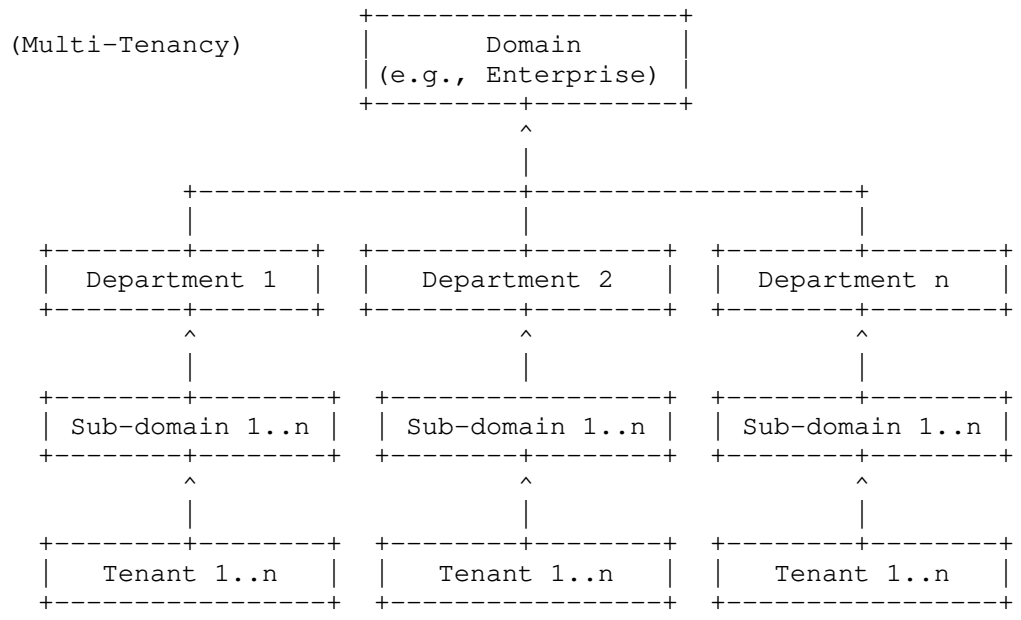

Figure 7: Multi-tenancy Diagram

5.1. 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. Figure 8 shows the XML instance of the Policy-domain object. Multi-tenancy model should be able to work in all such environments. The Policy-Domain object SHALL have following information:

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

Address: Address of the organization or customer.

Contact: Contact information of the organization or customer.

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

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

```
   +--rw policy-domain* [name]
      +--rw name                       string
      +--rw date?                      yang:date-and-time
      +--rw address?                   string
      +--rw contact?                   string
      +--rw policy-tenant* [name]
      +--rw authentication-method?     -> /policy
                                          /multi-tenancy
                                          /policy-mgnt-auth-method
                                          /name
         ...
         ...
```

Figure 8: Policy domain YANG data tree

5.2. Policy-Tenant

This object defines an entity within an organization. The entity could be a department or business unit within an Enterprise organization that would like to manage its own Policies due to regulatory compliance or business reasons. Figure 9 shows the XML instance of the Policy-tenant object. The Policy-Tenant object SHALL have following information:

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

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

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

```
+--rw policy-tenant* [name]
   +--rw name      string
   +--rw date?     yang:date-and-time
   +--rw domain?   -> /policy
                      /multi-tenancy
                      /policy-domain
                      /name
```

Figure 9: Policy tenant YANG data tree

5.3. Policy-Role

This object defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. It provides a convenient way to assign policy users to a job function or a set of permissions within the organization. Figure 10 shows the XML instance of the Policy-role object. The Policy-Role object SHALL have the following information:

Name: This field identifies the name of the role.

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

Access-Profile: This field identifies the access profile for the role. The profile grants or denies the permissions to access Endpoint Groups for the purpose of policy management or may restrict certain operations related to policy managements. There are two permission types, read-only and read-and-write, to choose from for each access-profile.

```
+--rw policy-role
|  +--rw name?              string
|  +--rw date?              yang:date-and-time
|  +--rw access-profile* [name]
|     +--rw name               string
|     +--rw date?              yang:date-and-time
|     +--rw permission-type?   identityref
```

Figure 10: Policy role YANG data tree

5.4. Policy-User

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

application requiring access to Security Controller. Figure 11 shows the XML instance of the Policy-user object. The Policy-User object SHALL have the following information:

Name: Name of a user.

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

Password: User password for basic authentication.

Email: E-mail address of the user.

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

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

```
+--rw policy-user* [name]
|  +--rw name          string
|  +--rw date?         yang:date-and-time
|  +--rw password?     string
|  +--rw email?        string
|  +--rw scope-type?   identityref
|  +--rw role?         -> /policy
                          /multi-tenancy
                          /policy-role
                          /access-profile
                          /name
```

Figure 11: Policy user YANG data tree

5.5. Policy Management Authentication Method

This object represents authentication schemes supported by Security Controller. Figure 12 shows the XML instance of the Policy Management Authentication Method onject. This Policy-Management-Authentication-Method object SHALL have the following information:

Name: This field identifies name of this object.

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

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

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

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

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

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

```
   +--rw policy-mgnt-auth-method* [name]
      +--rw name                       string
      +--rw date?                      yang:date-and-time
      +--rw mutual-authentication?     boolean
      +--rw password
      |  +--rw password?               password-type
      +--rw token
      |  +--rw token?                  string
      |  +--rw token-server?           inet:ipv4-address
      +--rw certificate
      |  +--rw certificate?            certificate-type
      |  +--rw certificate-server?     inet:ipv4-address
      +--rw single-sign-on
         +--rw credential?             certificate-type
         +--rw certificate-server?     inet:ipv4-address
```

Figure 12: Policy management authentication method YANG data tree

6. Information Model for Policy Endpoint Groups

The Policy Endpoint Group is a very important part of building User-construct based policies. Security Admin would create and use these objects to represent a logical entity in their business environment, where a Security Policy is to be applied. There are multiple managed objects that constitute a Policy Endpoint Group as shown in Figure 13. Figure 14 shows the XML instance of the endpoint-group object. shows the XML instance of the User-group object.. This section lists these objects and relationship among them.

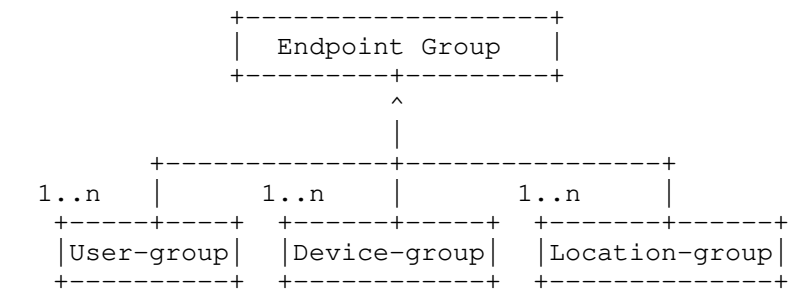

Figure 13: Endpoint Group Diagram

```
+--rw endpoint-group
   +--rw user-group* [name]
   |  ...
   +--rw device-group* [name]
   |  ...
   +--rw location-group* [name]
      ...
```

Figure 14: Endpoint Group YANG data tree

6.1. User Group

This object represents a User-group. Figure 15 shows the XML instance of the User-group object.The User-Group object SHALL have the following information:

Name: This field identifies the name of this object.

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

IP-Address: This field identifies the IP address of a user.

Range-IP-Address: This field is a range of IP addresses of users.

```
      +--rw user-group* [name]
         +--rw name                          string
         +--rw date?                         yang:date-and-time
         +--rw (match-type)?
            +--:(exact-match)
            |  +--rw ip-address*             inet:ipv4-address
            +--:(range-match)
               +--rw range-ip-address* [start-ip-address end-ip-address]
                  +--rw start-ip-address    inet:ipv4-address
                  +--rw end-ip-address      inet:ip-address
```

Figure 15: User group YANG data tree

6.2. Device-Group

This object represents a Device-group. Figure 16 shows the XML instance of the Device-group object.The Device-Group object SHALL have the following information:

Name: This field identifies the name of this object.

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

IP-Address: This field identifies the IP address of a device.

Range-IP-Address: This field is a range of IP addresses of devices.

```
       +--rw device-group* [name]
          +--rw name                          string
          +--rw date?                         yang:date-and-time
          +--rw (match-type)?
            +--:(exact-match)
            |  +--rw ip-address*             inet:ipv4-address
            +--:(range-match)
               +--rw range-ip-address* [start-ip-address end-ip-address]
                  +--rw start-ip-address    inet:ipv4-address
                  +--rw end-ip-address      inet:ip-address
```

Figure 16: Device group YANG data tree

6.3. Location-Group

This object represents a location group based on either tag or other information. Figure 17 shows the XML instance of the Location-group

object. The Location-group object SHALL have the following information:

Name: This field identifies the name of this object.

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

continent: to identify which continent the location group member is based at.

```
  +--rw location-group* [name]
     +--rw name         string
     +--rw date?        yang:date-and-time
     +--rw continent?   identityref
```

Figure 17: Location group YANG data tree

7. Information Model for Threat Prevention

The threat prevention plays an important part in the overall security posture by reducing the attack surfaces. This information could come from various threat feeds (i.e., sources for obtaining the threat information), such as EmergingThreats.com or AlienVault.com. There are multiple managed objects that constitute this category. This section lists these objects and relationship among them. Figure 19 shows the XML instance of a Threat-prevention object.

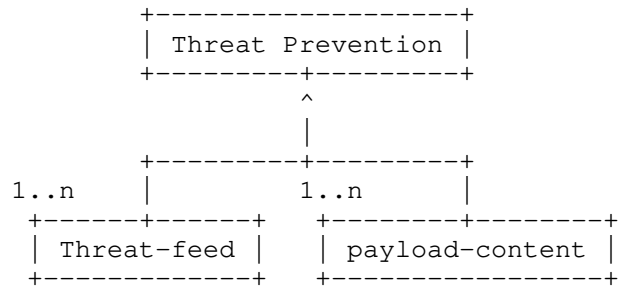

Figure 18: Threat Prevention Diagram

```
      +--rw threat-prevention
      |  +--rw threat-feed-list* [name]
      |   ...
      |  +--rw payload-content* [name]
      |   ...
```

Figure 19: Threat Prevention YANG data tree

7.1. Threat-Feed

This object represents a threat feed which provides signatures of malicious activities. Figure 20 shows the XML instance of a Threat-feed-list. The Threat-Feed object SHALL have the following information:

Name: This field identifies the name of this object.

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

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

Threat-file-types: This field identifies the information about the file types identified and reported by the threat-feed.

signatures: This field contains the signatures of malicious programs or activities provided by the threat-feed.

```
   +--rw threat-feed-list* [name]
      +--rw name                              string
      +--rw date?                             yang:date-and-time
      +--rw threat-feed-server
      |  +--rw (match-type)?
      |  |  +--:(exact-match)
      |  |  |  +--rw ip-address*            inet:ipv4-address
      |  |  +--:(range-match)
      |  |     +--rw range-ip-address* [start-ip-address end-ip-address]
      |  |        +--rw start-ip-address    inet:ipv4-address
      |  |        +--rw end-ip-address      inet:ip-address
      |  +--rw threat-feed-description?     string
      +--rw threat-file-types*                identityref
      +--rw signatures*                       string
```

Figure 20: Threat feed YANG data tree

7.2. Payload-content

This object represents a custom list created for the purpose of defining exception to threat feeds. Figure 21 shows the XML instance of a Payload-content list. The Payload-content object SHALL have the following information:

Name: This field identifies the name of this object.

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

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

```
+--rw payload-content*  [name]
|  +--rw name           string
|  +--rw date?          yang:date-and-time
|  +--rw content*       string
```

Figure 21: Payload-content in YANG data tree

8. Role-Based Acess Control (RBAC)

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

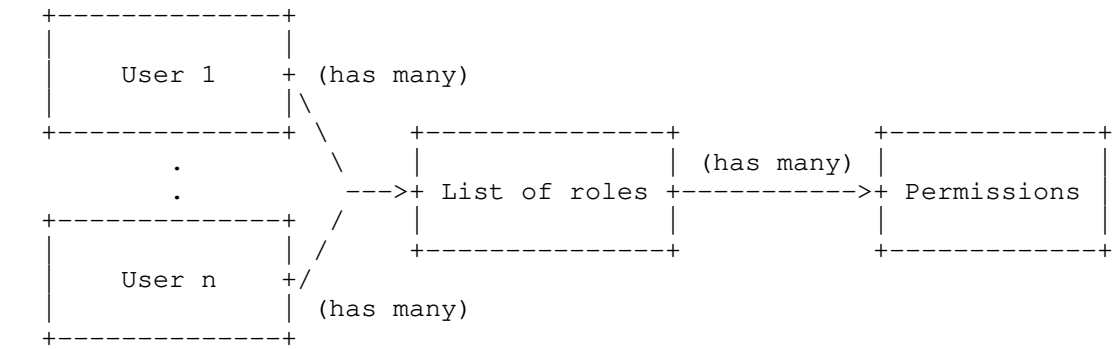

Figure 22: RBAC Diagram

As shown in Figure 22, a role represents a collection of permissions (e.g., accessing a file server or other particular resources). A

role may be assigned to one or multiple users. Both roles and permissions can be organized in a hirarchy. A role may consists of other roles and permissions.

Following are the steps required to build RBAC:

1. Defining roles and permissions.

2. Establishing relations among roles and permissions.

3. Defining users.

4. Associating rules with roles and permissions.

5. assigning roles to users.

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

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

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

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

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-cfi-policy.yang"
module ietf-i2nsf-cfi-policy {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy";
  prefix
    cfi-policy;
```

```
  import ietf-yang-types{
    prefix yang;
    reference
    "Section 3 of RFC 6991";
  }

  import ietf-inet-types{
    prefix inet;
    reference
    "Section 4 of RFC 6991";
  }

  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 is a YANG module for Consumer-Facing Interface.
   Copyright (c) 2018 IETF Trust and the persons identified as
   authors of the code. All rights reserved.
   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject
   to the license terms contained in, the Simplified BSD License
   set forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (http://trustee.ietf.org/license-info).
   This version of this YANG module is part of RFC XXXX; see
   the RFC itself for full legal notices.";


  revision "2019-03-11"{
    description "latest revision";
    reference
      "draft-ietf-consumer-facing-interface-dm-02";
  }
```

```
  identity permission-type {
    description
      "Base identity for the permission types.";
  }

  identity read-only {
    base permission-type;
    description
      "Identity for read-only permission.";
  }
  identity read-and-write {
    base permission-type;
    description
      "Identity for read permission.";
  }

  identity scope-type {
    description
      "Base Identity for scope-type.";
  }
  identity tenant-wide {
    base scope-type;
    description
      "Base Identity for tenant-wide scope type.";
  }
  identity domain-wide {
    base scope-type;
    description
      "Base Identity for domain-wide scope type.";
  }

  identity malware-file-type {
    description
      "Base identity for malware file types.";
  }
  identity executable-file {
    base malware-file-type;
    description
      "Identity for executable file types.";
  }
  identity doc-file {
    base malware-file-type;
    description
      "Identity for Microsoft document file types.";
  }
  identity html-app-file {
    base malware-file-type;
    description
```

```
      "Identity for html application file types.";
  }
  identity javascript-file {
    base malware-file-type;
    description
      "Identity for Javascript file types.";
  }
  identity pdf-file {
    base malware-file-type;
    description
      "Identity for pdf file types.";
  }
  identity dll-file {
    base malware-file-type;
    description
      "Identity for dll file types.";
  }
  identity msi-file {
    base malware-file-type;
    description
      "Identity for Microsoft installer file types.";
  }

  identity security-event-type {
    description
      "Base identity for security event types.";
  }
  identity ddos {
    base malware-file-type;
    description
      "Identity for DDoS event types.";
  }
  identity spyware {
    base malware-file-type;
    description
      "Identity for spyware event types.";
  }
  identity trojan {
    base malware-file-type;
    description
      "Identity for Trojan infection event types.";
  }
  identity ransomeware {
    base malware-file-type;
    description
      "Identity for ransomeware infection event types.";
  }
```

```
  identity continent {
    description
    "Base Identity for continent types.";
  }

  identity africa {
    base continent;
    description
    "Identity for africa.";
  }
  identity asia {
    base continent;
    description
    "Identity for asia.";
  }
  identity europe {
    base continent;
    description
    "Identity for europe.";
  }
  identity north-america {
    base continent;
    description
    "Identity for north-america.";
  }
  identity south-america {
    base continent;
    description
    "Identity for south-america.";
  }
  identity oceania {
    base continent;
    description
    "Identity for Oceania";
  }
  typedef certificate-type {

    type enumeration {
      enum cer {
        description
           "The extension type is '.cer'.";
      }
      enum crt {
        description
           "The extension type is '.crt'.";
      }
      enum key {
        description
```

```
            "The extension type is '.key'.";
        }
      }
      description
        "CRT certificate extension, which is used for certificates.
        The certificates may be encoded as binary DER or as ASCII PEM.
        The CER and CRT extensions are nearly synonymous.  Most common
        among *nix systems. CER certificate extension, which is an
        alternate form of .crt (Microsoft Convention) You can use MS to
        convert .crt to .cer (.both DER encoded .cer, or base64[PEM]
        encoded .cer). The KEY extension is used both for public and
        private PKCS#8 keys. The keys may be encoded as binary DER or
        as ASCII PEM.";
    }

  grouping meta {
    description
      "The purpose of this grouping is to avoid repetition of same fields, such as
   'name' and 'date'.";
    leaf name {
      type string;
      description "This is the name for an entity.";
    }
    leaf date {
      type yang:date-and-time;
      description "This is the date when the entity is created or modified.";
    }
  }

  grouping ip-address {
    description
      "There are two types to configure a security policy
      for IPv4 address, such as exact match and range match.";
      choice match-type {
        description
          "User can choose between 'exact match' and 'range match'.";
        case exact-match {
          leaf-list ip-address {
            type inet:ipv4-address;
            description
            "Exactly matches the IP address specified.";
          }
        }
        case range-match {
          list range-ip-address {
            key "start-ip-address end-ip-address";
            leaf start-ip-address {
              type inet:ipv4-address;
              description
```

```
              "Start IP address for a range match.";
          }
          leaf end-ip-address {
            type inet:ip-address;
            description
              "End IP address for a range match.";
          }
          description
            "Range match for an IP-address.";
        }
      }
    }
}

grouping user-group {
  description
    "This grouping is to remove repetition of
    'name' and 'ip-address' fields.";
  uses meta;
  uses ip-address;
}

grouping device-group {
  description
    "This grouping is to remove repetition of
    'name', 'ip-address', and 'protocol' fields.";
  uses meta;
  uses ip-address;
  leaf-list protocol {
    type string;
    description
    "This represents the port numbers of devices.";
  }
}

grouping location-group {
  description
    "This grouping is to remove repetition of
    'name' and 'continent' fields.";
  uses meta;
  leaf continent {
    type identityref {
      base continent;
        }
    description
      "location-group-based on geo-ip of
      respective continent.";
  }
```

```
}

grouping payload-string {
  description
    "This grouping is to remove repetition of
    'name' and 'content' fields.";
  uses meta;
  leaf-list content {
    type string;
    description
    "This represents the payload string content.";
  }
}

container policy {
    leaf policy-name {
      type string;
      description
        "The name which identifies the policy.";
    }
    description
    "There can be a multiple number of security rules in
    a policy object. This object is a policy instance to
    have complete information such as where and when a
    policy need to be applied.";

    list rule {
      leaf rule-name {
        type string;
        mandatory true;
        description
        "This represents the name for rules.";
      }
      key "rule-name";
      description
        "There can be a single or multiple number of rules.";

      leaf date {
       type yang:date-and-time;
        description
         "Date this object was created or last
          modified";
      }
      list event {
        uses meta;
        key "name";
          description
          "This represents the event map group name.";
```

```
        leaf security-event {
          type identityref {
            base security-event-type;
          }
          description
            "This contains the description of security events.";
        }
        leaf enforce-type {
          type enumeration{
            enum admin-enforced {
              description
                "The enforcement type is admin-enforced.";
            }
            enum time-enforced {
              description
                "The enforcement type is time-enforced.";
            }
            enum event-enforced {
              description
                "The enforcement type is event-enforced.";
            }
          }
          description
            "This field identifies the event of
            policy enforcement trigger type.";
        }
       container time-information {
        description
          "The container for time-information.";
           leaf begin-time {
             type string;
             description
             "This is start time for time zone";
           }
           leaf end-time {
             type string;
             description
             "This is end time for time zone";
           }
           }
       container recursive {
        description
          "The container to represent the recursiveness
          of the rule.";
         leaf recur {
           type boolean;
           description
           "recursive enforcement";
```

```
        }
        leaf recursive-type{
          type enumeration{
            enum daily {
             description
               "The recursive type is daily.";
            }
            enum weekly {
             description
               "The recursive type is weekly.";
            }
            enum monthly {
             description
               "The recursive type is monthly.";
            }
          }
          description
           "This leaf identifies the recursive type.";
        }
      }
    }
    container condition {
      description
        "The conditions for general security policies.";
          container firewall-condition {
            description
              "The general firewall condition.";
            container source-target {
              description
                "This represents the source.";
              leaf src-target {
                type leafref {
                  path "/policy/endpoint-group/user-group/name";
                }
                description
                 "This describes the paths to
                 the source reference.";
              }
            }
            container destination-target {
              description
                "This represents the destination.";
                leaf-list dest-target {
                  type leafref {
                    path "/policy/endpoint-group/user-group/name";
                  }
                  description
                    "This describes the paths to the
```

```
                    destination target reference.";
                }
              }
            }
            container ddos-condition {
              description
                "The condition for DDoS mitigation.";
              container source-target {
                description
                  "This represents the source.";
                leaf-list src-target {
                    type leafref {
                      path "/policy/endpoint-group/device-group/name";
                    }
                    description
                      "This describes the path to the
                      source target references.";
                }
              }
              container destination-target {
                description
                  "This represents the target.";
                leaf-list dest-target {
                  type leafref {
                    path "/policy/endpoint-group/device-group/name";
                  }
                  description
                    "This describes the path to the
                    destination target references.";
                }
              }
              container rate-limit {
                description "This describes the rate-limit.";
                leaf packet-per-second {
                  type uint8;
                  description
                  "The rate-limit limits the amount of incoming packets.";
                }
              }
            }
            container custom-condition {
              description
                "The condition based on packet contents.";
              container source-target {
                description
                  "This represents the source.";
                leaf-list src-target {
                  type leafref {
```

```
                    path "/policy/threat-prevention/payload-content/name";
                  }
                  description
                   "Describes the payload string
                   content condition source.";
                }
              }
              container destination-target {
                description
                  "This represents the destination.";
                leaf dest-target {
                  type leafref {
                    path "/policy/threat-prevention/payload-content/name";
                  }
                  description
                   "Describes the payload string
                  content condition destination.";
                }
              }
            }
            container threat-feed-condition {
              description
                "The condition based on the threat-feed information.";
              container source-target {
                description
                  "This represents the source.";
                leaf-list src-target {
                  type leafref {
                    path "/policy/threat-prevention/threat-feed-list/name";
                  }
                  description "Describes the threat-feed
                  condition source.";
                }
              }
              container destination-target {
                description
                  "This represents the destination.";
                leaf dest-target {
                  type leafref {
                    path "/policy/threat-prevention/threat-feed-list/name";
                  }
                  description "Describes the threat-feed
                  condition destination.";
                }
              }
            }
      }
      container action {
```

```
        description
          "This is the action container.";
        leaf primary-action {
          type string;
          mandatory true;
          description
            "This field identifies the action when a rule
            is matched by NSF. The action could be one of
            'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
            'AUTHENTICATE-SESSION', 'IPS, 'APP-FIREWALL', etc.";
        }
        leaf secondary-action {
          type string;
          description
            "This field identifies additional actions if
            a rule is matched. This could be one of 'LOG',
            'SYSLOG', 'SESSION-LOG', etc.";
        }
      }
      leaf owner {
        type string;
          description
           "This field defines the owner of this
           policy. Only the owner is authorized to
           modify the contents of the policy.";
      }
    }

container multi-tenancy {
    description
      "The multi-tenant environment information
       in which the policy is applied. The Rules
       in the Policy can refer to sub-objects
       (e.g., domain, tenant, role, and user) of it.";

    list policy-domain {
      uses meta;
      key "name";
      leaf address {
        type string;
        description
          "The address details of the organization
          or customer.";
      }
      leaf contact {
        type string;
        description
          "contact information of the organization
```

```
           or customer.";
      }
      list policy-tenant {
        uses meta;
        key "name";
        description
        "This represents the list of tenants";
        leaf domain {
          type leafref {
            path "/policy/multi-tenancy/policy-domain/name";
          }
          description
          "This field identifies the domain to which this
          tenant belongs. This should be reference to a
          'Policy-Domain' object.";
        }
      }
      leaf authentication-method {
        type leafref {
          path "/policy/multi-tenancy/policy-mgnt-auth-method/name";
        }
        description
        "Authentication method to be used for this domain.
        It should be a reference to a 'policy-mgmt-auth-method'
        object.";
      }
      description
        "This represents the list of policy domains.";
    }
    container policy-role {
      uses meta;
      description
      "This represents the list of policy roles.";
      list access-profile {
        uses meta;
        key "name";
        description
          "This field identifies the access profile for the
          role. The profile grants or denies access to policy
          objects.";
        leaf permission-type {
          type identityref {
            base permission-type;
          }
          default read-only;
          description
            "Permission type for access-profile: read-only
            or read-and-write.";
```

```
        }
      }
    }
    list policy-user {
      uses meta;
      key "name";
      description
      "This represents the policy users.";
      leaf password {
        type string;
        description
          "User password for basic authentication";
      }
      leaf email {
        type string;
        description
          "The email account of a user";
      }
      leaf scope-type {
        type identityref {
          base scope-type;
        }
        default tenant-wide;
        description
          "identifies whether a user has domain-wide
          or tenant-wide privileges";
      }
      leaf role {
        type leafref {
          path "/policy/multi-tenancy/policy-role/access-profile/name";
        }
        description
          "This represents the reference to the
          access-profiles.";
      }
    }
    list policy-mgnt-auth-method {
      uses meta;
      key "name";
      leaf mutual-authentication {
        type boolean;
          description
          "To identify whether the authentication
          is mutual.";
      }
      container password {
        leaf password {
          type string;
```

```
          description
            "This should be defined using the
            regular expression.";
        }
        description
          "This represents the password method.";
      }
      container token {
        leaf token {
          type string;
          description
            "This should be defined according to
            the token scheme.";
        }
        description
          "This represents the token method.";
        leaf token-server {
          type inet:ipv4-address;
            description
              "The token-server information if the
              authentication method is token-based";
          }
        }
      container certificate {
        description
          "This represents the certificate method.";
        leaf certificate {
          type certificate-type;
          description
            "This represents the certificate-type.";
        }
        leaf certificate-server {
          type inet:ipv4-address;
            description
              "The certificate-server information if
               the authentication method is
               certificate-based";
        }
      }
      container single-sign-on {
        description
          "This represents the authentication method
          for single-sing-on.";
        leaf credential {
          type certificate-type;
          description
            "This represents the authentication
            using user credentials.";
```

```
        }
        leaf certificate-server {
          type inet:ipv4-address;
            description
              "The certificate-server information if
              the authentication method is
              certificate-based";
        }
      }
    description
      "This represents the policy managegement method.";
  }
}
container endpoint-group {
        description
          "A logical entity in their business
          environment, where a security policy
          is to be applied.";
        list user-group {
          uses user-group;
          key "name";
          description
            "This represents the user group.";
        }
        list device-group {
          uses device-group;
          key "name";
          description
            "This represents the device group.";
        }
        list location-group{
          uses location-group;
          key "name";
          description
            "This represents the location group.";
        }

}
container threat-prevention {
      description
      "this describes the list of threat-prevention.";
      list threat-feed-list {
        uses meta;
        key "name";
          description
          "This represents the threat feed list.";
        container threat-feed-server {
          uses ip-address;
```

```
          description
            "This describes the threat-feed server.";
          leaf threat-feed-description {
            type string;
            description
            "This object containes threat-feed
            description.";
          }
        }
        leaf-list threat-file-types {
          type identityref {
            base malware-file-type;
          }
          default executable-file;
          description
            "This contains a list of file types needed to
            be scanned for the virus.";
        }
        leaf-list signatures {
          type string;
          description
            "This contains a list of signatures or hash
            of the threats.";
        }
      }
      list payload-content {
          uses payload-string;
          key "name";
          description
            "This represents the payload-string group.";
      }
    }
  }
}
<CODE ENDS>
```

Figure 23: YANG for policy-general

10. Example XML Output for Various Scenarios

This section describes the XML instances for different policies examples that are delivered through Consumer-Facing Interface. The considered use cases are: VoIP/VoLTE security service, DDoS-attack mitigation, time-based firewall as a web-filter.

10.1. DB registration: Information of positions and devices (Endpoint group)

In order to create a rule of a security policy, it is essential to first register data (those which are used to form such rule) to the database. For example, The endpoint group consists of three different groups: user-group, device-group, and payload-group. Each of these groups have separate group members with information other than meta ("name" or "date"), such as ip-addresses or protocols used by devices. Figure 24 shows an example XML representation of the registered information for the user-group and device-group.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <ietf-i2nsf-cfi-policy:endpoint-group>
    <user-group>
      <name>employees</name>
      <range-ip-address>
        <start-ip-address>221.159.112.1</start-ip-address>
        <end-ip-address>221.159.112.90</end-ip-address>
      </range-ip-address>
    </user-group>
    <device-group>
      <name>webservers</name>
      <range-ip-address>
        <start-ip-address>221.159.112.91</start-ip-address>
        <end-ip-address>221.159.112.97</end-ip-address>
      </range-ip-address>
      <protocol>http</protocol>
      <protocol>https</protocol>
    </device-group>
  </ietf-i2nsf-cfi-policy:endpoint-group>
```

Figure 24: Registering user-group and device-group information

10.2. Scenario 1: Block SNS access during business hours

The first example scenario is to "block SNS access during business hours" using a time-based firewall policy. In this scenario, all users registered as "employee" in the user-group list are unable to access Social Networking Services (SNS) during the office hours. The XML instance is described below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ietf-i2nsf-cfi-policy:policy>
  <policy-name>security_policy_for_blocking_sns</policy-name>
  <rule>
    <rule-name>block_access_to_sns_during_office_hours</rule-name>
    <event>
      <time-information>
        <begin-time>09:00</begin-time>
        <end-time>18:00</end-time>
      </time-information>
    </event>
    <condition>
      <firewall-condition>
        <source-target>
          <src-target>employees</src-target>
        </source-target>
      </firewall-condition>
      <custom-condition>
        <destination-target>
          <dest-target>sns-websites</dest-target>
        </destination-target>
      </custom-condition>
    </condition>
    <action>
      <primary-action>drop</primary-action>
    </action>
  </rule>
</ietf-i2nsf-cfi-policy:policy>
```

Figure 25: An XML Example for Time-based Firewall

Time-based-condition Firewall

1. The policy name is "security_policy_for_blocking_sns".

2. The rule name is "block_access_to_sns_during_office_hours".

3. The Source-target is "employees".

4. The destination target is "sns-websites". "sns-websites" is the key which represents the list containing the information, such as URL, about sns-websites.

5. The action required is to "drop" any attempt to connect to websites related to Social networking.

10.3. Scenario 2: Block malicious VoIP/VoLTE packets coming to the company

The second example scenario is to "block malicious VoIP/VoLTE packets coming to the company" using a VoIP policy. In this scenario, the calls comming from from VOIP and/or VOLTE sources with VOLTE IDs that are classified as malicious are dropped. The IP addresses of the employees and malicious VOIP IDs should be blocked are stored in the database or datastore of the enterprise. Here and the rest of the cases assume that the security administrators or someone responsible for the existing and newly generated policies, are not aware of which and/or how many NSFs are needed to meet the security requirements. Figure 26 represents the XML document generated from YANG discussed in previous sections. Once a high-level seucurity policy is created by a security admin, it is delivered by the Consumer-Facing Interface, through RESTCONF server, to the security controller. The XML instance is described below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ietf-i2nsf-cfi-policy:policy>
  <policy-name>security_policy_for_blocking_malicious_voip_packets</policy-name>
  <rule>
    <rule-name>Block_malicious_voip_and_volte_packets</rule-name>
    <condition>
      <custom-condition>
        <source-target>
          <src-target>malicious-id</src-target>
        </source-target>
      </custom-condition>
      <firewall-condition>
        <destination-target>
          <dest-target>employees</dest-target>
        </destination-target>
      </firewall-condition>
    </condition>
    <action>
      <primary-action>drop</primary-action>
    </action>
  </rule>
</ietf-i2nsf-cfi-policy:policy>
```

Figure 26: An XML Example for VoIP Security Service

Custom-condition Firewall

1. The policy name is "security_policy_for_blocking_malicious_voip_packets".

2. The rule name is "Block_malicious_voip_and_volte_packets".

3. The Source-target is "malicious-id". This can be a single ID or a list of IDs, depending on how the ID are stored in the database. The "malicious-id" is the key so that the security admin can read every stored malicious VOIP IDs that are named as "malicious-id".

4. The destination target is "employees". "employees" is the key which represents the list containing information about employees, such as IP addresses.

5. The action required is "drop" when any incoming packets are from "malicious-id".

10.4. Scenario 3: Mitigate HTTP and HTTPS flood attacks on a company web Server

The third example scenario is to "Mitigate HTTP and HTTPS flood attacks on a company web Server" using a DDoS-attack mitigation policy. Here, the time information is not set because the service provided by the network should be maintained at all times. If the packets sent by any sources are more than the set threshold, then the admin can set the percentage of the packets to be dropped to safely maintain the service. In this scenario, the source is set as "any" to block any sources which send abnormal amount of packets. The destination is set as "web_server01". Once the rule is set and delivered and enforced to the nsfs by the securiy controller, the NSFs will monitor the incoming packet amounts and the destination to act according to the rule set. The XML instance is described below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ietf-i2nsf-cfi-policy:policy>
  <policy-name>security_policy_for_ddos_attacks</policy-name>
  <rule>
    <rule-name>100_packets_per_second</rule-name>
    <condition>
      <ddos-condition>
        <destination-target>
          <dest-target>webservers</dest-target>
        </destination-target>
        <rate-limit>
          <packet-per-second>100</packet-per-second>
        </rate-limit>
      </ddos-condition>
    </condition>
    <action>
      <primary-action>drop</primary-action>
    </action>
  </rule>
</ietf-i2nsf-cfi-policy:policy>
```

Figure 27: An XML Example for DDoS-attack Mitigation

DDoS-condition Firewall

1. The policy name is "security_policy_for_ddos_attacks".

2. The rule name is "100_packets_per_second".

3. The destination target is "webservers". "webservers" is the key which represents the list containing information, such as IP addresses and ports, about web-servers.

4. The rate limit exists to limit the incoming amount of packets per second. In this case the rate limit is "100" packets per second. This amount depends on the packet receiving capacity of the server devices.

5. The Source-target is all sources which send abnormal amount of packets.

6. The action required is to "drop" packet reception is more than 100 packets per second.

11. 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.

12. IANA Considerations

This document requests IANA to register the following URI in the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy
Registrant Contact: The I2NSF.
XML: N/A; the requested URI is an XML namespace.
```

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [RFC7950].

```
name: ietf-i2nsf-cfi-policy
namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy
prefix: cfi-policy
reference: RFC 7950
```

13. References

13.1. Normative References

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

13.2. Informative References

[client-facing-inf-im]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Information model for Client-Facing Interface to Security Controller", draft-kumar-i2nsf-client-facing-interface-im-07 (work in progress), July 2018.

[client-facing-inf-req]
Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", draft-ietf-i2nsf-client-facing-interface-req-05 (work in progress), May 2018.

[draft-ietf-i2nsf-capability]
Xia, L., Strassner, J., Huawei, Basile, C., PoliTo, Lopez, D., and TID, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-04 (work in progress), October 2018.

[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-07 (work in progress), January 2019.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

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

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <https://www.rfc-editor.org/info/rfc6087>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Changes from draft-ietf-i2nsf-consumer-facing-interface-dm-02

The following changes have been made from draft-ietf-i2nsf-consumer-facing-interface-dm-02:

- In this version of the WG draft, we merged the [client-facing-inf-im] and draft-ietf-i2nsf-consumer-facing-interface-dm-02 drafts. In sections 4 to 9, we describe the information model for the security policies delivered through the Consumer-Facing Interface. In sections 10 to 12, we provide and discuss the YANG data model and example XML outputs of security policies for various use cases.

- In Section 10, the following changes have been made: For "time-information" container in event sub-module list, the enforcement type is defined into three different types (admin-enforced, time-enforced, and event-enforced). Also, begin-time and end-time type has been defined seperately. The security policies can now be set recursively (daily, weekly, and monthly).

- "policy-role" now has the access-profile container, and previlege can be set separately per profile.

- "policy-user" information, such as email and password is newly defined by regular expressions.

- "authentication-method" in "policy-mgnt-auth-method" has been modified. More specifically, the authentication-method type has been changed from string to choice so that one can choose between password, token, and certificate. If not selected, password is used as a default.

- "Certificate-type" has been re-defined to include common certificate extensions, such as ".CRT", "CER", and "KEY".

- Used groupings to represent the groups in the Endpoint groups.

- Added examples for registering information (i.e., endpoint-groups, threat-prevention, and multi-tenancy.)

Appendix B. 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).

Appendix C. Contributors

This document is made by the group effort of I2NSF working group. Many people actively contributed to this document, such as Mahdi F. Dachmehchi and Daeyoung Hyun. The authors sincerely appreciate their contributions.

The following are co-authors of this document:

Hyoungshick Kim
Department of Software
2066 Seo-ro Jangan-gu
Suwon, Gyeonggi-do 16419
Republic of Korea

EMail: hyoung@skku.edu

Seungjin Lee
Department of Electrical and Computer Engineering
2066 Seo-ro Jangan-gu
Suwon, Gyeonggi-do 16419
Republic of Korea

EMail: jine33@skku.edu

Jinyong Tim Kim
Department of Electrical and Computer Engineering
2066 Seo-ro Jangan-gu
Suwon, Gyeonggi-do 16419
Republic of Korea

EMail: timkim@skku.edu

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

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