I2NSF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 25, 2020

J. Jeong
E. Kim
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei
July 24, 2019

I2NSF Consumer-Facing Interface YANG Data Model
draft-ietf-i2nsf-consumer-facing-interface-dm-06

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 types of managed objects and the relationship among them needed to build the interface. The information model is organized based on the "Event-Condition-Action" (ECA) policy model defined by a capability information model for I2NSF [i2nsf-capability-im], 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

Copyright Notice

Table of Contents

1. Introduction

In a framework of Interface to Network Security Functions (I2NSF), each vendor can register their NSFs using a Developer's Management System (DMS). Assuming that vendors also provide the front-end web applications registered with an I2NSF User, 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 the I2NSF User and Security Controller communicate using this interface. Therefore, this document specifies the required information, their data types, and encoding schemes so that high-level security policies (or configuration information for security policies) can be transferred to the Security Controller through the Consumer-Facing Interface. These policies can easily be translated by the Security Controller into low-level security policies. The Security Controller delivers the translated policies to Network Security Functions (NSFs) according to their respective security capabilities for the required 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 Administrator (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 requirements. 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-Action" (ECA) policy model.

An NSF Capability model is proposed in [i2nsf-capability-im] 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 uses 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 this document.

```
                       +-----------------+    +-----------------+
                       | Consumer-Facing |    | Consumer-Facing |
                       |    Interface    +--->+    Interface    |
                       |Information Model|    |    Data Model   |
                       +--------+--------+    +-----------------+
                                ^
                                |
                  +-------------+-------------+------------+
                  |             |             |            |
             +----+----+  +-----+----+  +-----+----+  +----+----+
             |  Multi  |  |  Policy  |  | Endpoint |  | Threat  |
             | Tenancy |  |          |  |  groups  |  |  feed   |
             +---------+  +-----+----+  +----------+  +---------+
                                ^
                                |
                         +------+------+
                         |     Rule    |
                         +------+------+
                                ^
                                |
               +----------------+----------------+
               |                |                |
        +------+------+  +------+------+  +------+------+
        |    Event    |  |  Condition  |  |    Action   |
        +-------------+  +-------------+  +-------------+
```

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 from a single information model.

The efficient and flexible provisioning of network functions by a Network Functions Virtualization (NFV) system leads to a rapid advance in the network industry. As practical applications, Network Security Functions (NSFs), such as firewall, Intrusion Detection System (IDS)/Intrusion Prevention System (IPS), and attack mitigation, can also be provided as Virtual Network Functions (VNF) in the NFV system. By the efficient virtualization 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-req].

This document follows the guidelines of [RFC8407], 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 Administrator (i.e., I2NSF User) using Consumer-Facing Interface toward Security Controller; the policy would be enforced on an NSF. Figure 2 shows the YANG tree of the Policy object. The Policy 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.

Rules: This field contains a list of rules. These rules are defined for 1) communication between two Endpoint Groups, 2) for preventing communication with externally or internally identified threats, and 3) for implementing business requirement such as controlling access to internal or external resources for meeting regulatory compliance or business objectives. An organization may restrict certain communication between a set of user and applications for example. The threats may be from threat feeds obtained from external sources or dynamically identified by using specialty devices in the network. Rule conflict analysis should be triggered by the monitoring service to perform an exhaustive detection of anomalies among the configuration rules installed into the security functions.

```
        +--rw i2nsf-cfi-policy* [policy-name]
           +--rw policy-name          string
           +--rw rule* [rule-name]
           +--rw multi-tenancy
           +--rw endpoint-group
           +--rw threat-prevention
```

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 YANG tree of the Rule object. The rule object SHALL have the following information:

Name: This field identifies the name of this object.

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.

IPsec-Method: This field contains the information about IPsec method type. There are two types such as IPsec-IKE and IPsec-IKEless [i2nsf-ipsec].

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 event
              +--rw (condition)?
              +--rw action
              +--rw ipsec-method
              +--rw owner                            identityref
```

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 set time or security event. Figure 4 shows the YANG tree of the Event object. Event object SHALL have following information:

Security-event: This field identifies for which security event the policy is enforced. The examples of security events are: "DDOS", "spyware", "trojan", and "ransomware".

Enforce-type: This field identifies whether the event of triggering policy enforcement is "Admin" or "Time".

Admin: This represents the enforcement type based on admin's decision.

Time: This represents the security rule is enforced based on begin-time and end-time information.

Frequency: This represents how frequent the rule should be enforced. There are four options: "only-once", "daily", "weekly" and "monthly".

```
                +--rw event
                   +--rw security-event            identityref
                   +--rw (enforce-type)?
                   |  +--:(admin)
                   |  |  +--rw admin?              identityref
                   |  +--:(time)
                   |     +--rw time-information
                   |        +--rw begin-time?   yang:date-and-time
                   |        +--rw end-time?     yang:date-and-time
                   +--rw frequency?                enumeration
```

Figure 4: Event Sub-model YANG Data Tree

4.2. Condition Sub-model

This object represents Conditions that Security Administrator 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 three different types of 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 YANG tree of the Condition object. The Condition Sub-model SHALL have following information:

Case (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 firewall-source and firewall-destination, 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 ddos-source and ddos-destination, 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 custon-source and custom-destination, each referring to the payload-groups defined and registered in the endpoint-group.

Threat-feed-condition: This field contains the information obtained from threat-feeds (e.g., Palo-Alto, or RSA-netwitness). This information is useful when security rule condition is based on the existing threat reports gathered by other sources. The source and destination is represented as threat-feed-source and threat-feed-destination. For clarity, threat-feed-source/destination represent the source/destination of a target security threat, not the information source/destination of a threat-feed.

```
        +--rw (condition)?
           +--:(firewall-condition)
           |  +--rw firewall-source
           |  |  +--rw src-target    -> /../../user-group/name
           |  +--rw firewall-destination
           |     +--rw dest-target*   -> /../../user-group/name
           +--:(ddos-condition)
           |  +--rw ddos-source
           |  |  +--rw src-target*   -> /../../device-group/name
           |  +--rw ddos-destination
           |  |  +--rw dest-target*   -> /../../device-group/name
           |  +--rw rate-limit
           |     +--rw packet-per-second?   uint16
           +--:(custom-condition)
           |  +--rw custon-source
           |  |  +--rw src-target*   -> /../../payload-content/name
           |  +--rw custom-destination
           |     +--rw dest-target    -> /../../payload-content/name
           +--:(threat-feed-condition)
              +--rw threat-feed-source
              |  +--rw src-target*   -> /../../threat-feed-list/feed-name
              +--rw threat-feed-destination
                 +--rw dest-target    -> /../../threat-feed-list/feed-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 YANG tree of the Action object. The Action object SHALL have following information:

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

Secondary-action: This field identifies the action when a rule is matched by an NSF. The action could be one of "log", "syslog", "session-log".

```
                  +--rw action
                     +--rw primary-action      identityref
                     +--rw secondary-action?   identityref
```

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 the relationship among these objects. Below diagram shows an example of multi-tenancy in an Enterprise domain.

```
                             +-------------------+
       (Multi-Tenancy)       |       Domain      |
                             |(e.g., Enterprise) |
                             +---------+---------+
                                       ^
                                       |
                  +--------------------+--------------------+
                  |                    |                    |
         +--------+-------+  +---------+--------+  +--------+--------+
         |  Department 1  |  |   Department 2   |  |  Department n   |
         +--------+-------+  +---------+--------+  +--------+--------+
                  ^                    ^                    ^
                  |                    |                    |
         +--------+--------+  +-----------------+  +--------+--------+
         | Sub-domain 1..n |  | Sub-domain 1..n |  | Sub-domain 1..n |
         +--------+--------+  +--------+--------+  +--------+--------+
                  ^                    ^                    ^
                  |                    |                    |
         +--------+--------+  +--------+--------+  +--------+--------+
         |   Tenant 1..n   |  |   Tenant 1..n   |  |   Tenant 1..n   |
         +-----------------+  +-----------------+  +-----------------+
```

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 YANG tree of the Policy-Domain object. Multi-tenancy model should be able to work in all such environments. The Policy-Domain object SHALL have the following information:

Domain-name: Name of the domain of an organization or enterprise.

Address: Address information of the organization or enterprise.

Contact: Contact information of the organization or enterprise.

```
                +--rw policy-domain*         [domain-name]
                   +--rw domain-name         identityref
                   +--rw address?            string
                   +--rw contact?            string
```

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 YANG tree of the Policy-Tenant object. The Policy-Tenant object SHALL have the following information:

Tenant-type: This field represents the type of tenant within a domain. In an enterprise, the examples of tenants could be the departments or divisions, such as HR department and Finance department.

```
               +--rw policy-tenant*                 [tenant-name]
                  +--rw tenant-type                 identityref
```

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 YANG tree of the Policy-Role object. The Policy-Role object SHALL have the following information:

Role-type: "This represent the roles within the tenants, in order to distinguish who may or may not have access to policies. The role types include "user", "group", "other", and "all". "user" "represents an individual where as group represents a group of users. "All" means both the individual and the group members, whereas "other" denotes anyone who is not a specific individual or a member of a specific group.

```
+--rw policy-role*                [role-name]
   +--rw role-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 YANG tree of the Policy-User object. The Policy-User object SHALL have the following information:

Name: Name of a user.

Password: User password for basic authentication. The crypto-hash mechanism for this entry is ianach:crypt-hash.

Email: E-mail address of the user.

Access-profile: This represents the access profile for the user. The access-profile is based on the permission-type and the scope type defined. The permission-types include "no-permission", read", "write", "execute", "read-and-write", "read-and-execute", and "write-and-execute"

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

```
              +--rw policy-user*              [name]
                 +--rw name                   string
                 +--rw password?              ianach:crypt-hash
                 +--rw email?                 string
                 +--rw access-profile*        [permission-type scope-type]
                    +--rw permission-type     identityref
                    +--rw scope-type          identityref
```

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 YANG tree of the Policy Management Authentication Method onject. This Policy-Management-Authentication-Method object SHALL have the following information:

Policy-mgmt-auth-method-instance: This field represent the authentication instances. Each instance is based on either client authentication, server authentication or both (mutual) authentication.

Policy-mgmt-auth-method: This represents the choices of authentication methods. Each instance of authentication consists of authentication methods chosen by an entity, such as a security admin. There are "Password-based", "token-based". "certificate-based", and "IPsec" authentication methods.

Password-list: This list contains the passwords that are encrypted using crypto-has algorithm (ianach:crypt-hash).

Token-list: This list contains the information such as the access tokens and a token server.

Cert-server-list: This list contains the certification server information such as server address (IPv4 and IPv6) and certificate types.

IPsec: This list has IPsec method types based on the identities defined. There are two types such as IPsec-IKE and IPsec-IKEless.

```
                                +--rw policy-mgmt-auth-method-instance* [auth-
instance-type]
                                   +--rw auth-instance-type        identityref
                                   +--rw (policy-mgmt-auth-method)?
                                      +--:(password-based)
                                      |  +--rw password-list* [password]
                                      |     +--rw password    ianach:crypt-hash
                                      +--:(token-based)
                                      |  +--rw token-list* [token]
                                      |     +--rw token           string
                                      |     +--rw token-server?   inet:ipv4-
address
                                      +--:(certificate-based)
                                      |  +--rw cert-server-list* [cert-server-
name]
                                      |     +--rw cert-server-name    string
                                      |     +--rw cert-server-ipv4?
inet:ipv4-address
                                      |     +--rw cert-server-ipv6?
inet:ipv6-address
                                      |     +--rw certificate* [cert-type]
                                      |        +--rw cert-type    identityref
                                      +--:(ipsec)
                                         +--rw ipsec-method* [method]
                                            +--rw method    identityref
```

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. A Security Administrator 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's Endpoint Group as shown in Figure 13. Figure 14 shows the YANG tree of the Endpoint-Group object. This section lists these objects and relationship among them.

```
                      +--------------------+
                      |  Endpoint Group   |
                      +---------+---------+
                                ^
                                |
                 +--------------+----------------+
          1..n   |      1..n    |       1..n     |
           +-----+----+  +------+-----+  +-------+------+
           |User-group|  |Device-group|  |Location-group|
           +----------+  +------------+  +--------------+
```

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 YANG tree of the User-Group object. The User-Group object SHALL have the following information:

Name: This field identifies the name of this object.

IP-address: This represents the IPv4 address of a user in the user group.

range-ipv4-address: This represents the IPv4 address of a user in the user gorup.

range-ipv6-address: This represents the IPv6 address of a user in the user gorup.

```
                       +--rw user-group* [name]
                          +--rw name                          string
                          +--rw (match-type)?
                             +--:(exact-match-ipv4)
                             |  +--rw ip-address*            inet:ipv4-address
                             +--:(exact-match-ipv6)
                             |  +--rw ip-address*            inet:ipv4-address
                             +--:(range-match-ipv4)
                             |  +--rw range-ipv4-address* [start-ipv4-address
end-ipv4-address]
                             |     +--rw start-ipv4-address    inet:ipv4-address
                             |     +--rw end-ipv4-address      inet:ipv4-address
                             +--:(range-match-ipv6)
                                +--rw range-ipv6-address* [start-ipv6-vaddress
end-ipv6-address]
                                   +--rw start-ipv6-address    inet:ipv6-address
                                   +--rw end-ipv6-address      inet:ipv6-address
```

Figure 15: User Group YANG Data Tree

6.2. Device Group

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

Name: This field identifies the name of this object.

IP-address: This represents the IPv4 address of a device in the device group.

range-ipv4-address: This represents the IPv4 address of a device in the device gorup.

range-ipv6-address: This represents the IPv6 address of a device in the device gorup.

Protorol: This represents the communication protocols used by the devices. The protocols are "SSH", "FTP", "SMTP", "HTTP", "HTTPS", and etc.

```
                      +--rw device-group* [name]
                         +--rw name                        string
                         +--rw (match-type)?
                            +--:(exact-match-ipv4)
                            |  +--rw ip-address*            inet:ipv4-address
                            +--:(exact-match-ipv6)
                            |  +--rw ip-address*            inet:ipv4-address
                            +--:(range-match-ipv4)
                            |  +--rw range-ipv4-address* [start-ipv4-address
end-ipv4-address]
                            |     +--rw start-ipv4-address    inet:ipv4-address
                            |     +--rw end-ipv4-address      inet:ipv4-address
                            +--:(range-match-ipv6)
                               +--rw range-ipv6-address* [start-ipv6-vaddress
end-ipv6-address]
                                  +--rw start-ipv6-address    inet:ipv6-address
                                  +--rw end-ipv6-address      inet:ipv6-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 YANG tree of the Location-Group object. The Location-Group object SHALL have the following information:

Name: This field identifies the name of this object.

geo-ip-ipv4: This field represents the IPv4 Geo-ip of a location.

geo-ip-ipv6: This field represents the IPv6 Geo-ip of a location.

continent: This field represents the continent where the location group member is at.

```
                                +--rw location-group* [name]
                                   +--rw name           string
                                   +--rw geo-ip-ipv4    inet:ipv4-address
                                   +--rw geo-ip-ipv6    inet:ipv6-address
                                   +--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 YANG tree of a Threat-Prevention object.

```
                            +-------------------+
                            | Threat Prevention |
                            +---------+---------+
                                      ^
                                      |
                            +---------+---------+
                    1..n    |         1..n      |
                     +------+------+   +--------+--------+
                     | Threat-feed |   | payload-content |
                     +-------------+   +-----------------+
```

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 YANG tree of a Threat-feed-list. The Threat-Feed object SHALL have the following information:

Feed-name: This field identifies the name of this object.

Feed-Server-ipv4: This represents the IPv4 server address of the feed provider, it may be external or local servers.

Feed-Server-ipv6: This represents the IPv6 server address of the feed provider, it may be external or local servers.

Feed-description: This is the description of the threat feed. The descriptions should have clear indication of the security attack such as attack type (e.g., APT) and file types used (e.g., executable malware).

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. The examples of signature types are "YARA", "SURICATA", and "SNORT".

```
                             +--rw threat-prevention
                                +--rw threat-feed-list* [feed-name]
                                   +--rw feed-name              identityref
                                   +--rw feed-server-ipv4?      inet:ipv4-
address
                                   +--rw feed-server-ipv6?      inet:ipv6-
address
                                   +--rw feed-description?      string
                                   +--rw threat-file-types*     identityref
                                   +--rw signatures*            identityref
```

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 YANG tree of a Payload-content list. The Payload-Content object SHALL have the following information:

Name: This field identifies the name of this object. For example, the name "backdoor" indicates the payload content is related to backdoor attack.

payload-description: This represents the description of how the payload content is related to a security attack.

Content: This contains the payload contents, which are involed in a security attack, as strings.

```
                                  +--rw payload-content* [name]
                                     +--rw name                   string
                                     +--rw payload-description    string
                                     +--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.

```
     +--------------+
     |              |
     |    User 1    + (has many)
     |              |\
     +--------------+ \     +---------------+            +-------------+
             .         \    |               | (has many) |             |
             .          --->+ List of roles +----------->+ Permissions |
     +--------------+  /    |               |            |             |
     |              | /     +---------------+            +-------------+
     |    User n    +/
     |              | (has many)
     +--------------+
```

Figure 22: Role-based Acess Control 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 provide both an information model and the corresponding YANG data model of I2NSF Consumer-Facing Interface. This interface can be used to deliver control and management messages 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.

```
<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";
  }

  import iana-crypt-hash {
    prefix ianach;
  }

  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 "2019-07-21"{
    description "latest revision";
    reference
      "draft-ietf-consumer-facing-interface-dm-03";
```

```
  }

  identity permission-type {
    description
      "Base identity for the permission types.";
  }
  identity no-permission {
    base permission-type;
    description
      "Identity for no-permission.";
  }
  identity read {
    base permission-type;
    description
      "Identity for read permission.";
  }
  identity write {
    base permission-type;
    description
      "Identity for write permission.";
  }
  identity execute {
    base permission-type;
    description
      "Identity for execute permission.";
  }
  identity write-and-execute {
    base permission-type;
    description
      "Identity for write & execute permission.";
  }
  identity read-and-execute {
    base permission-type;
    description
      "Identity for read & execute permission.";
  }
  identity read-and-write {
    base permission-type;
    description
      "Identity for read & write 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 ransomware {
    base malware-file-type;
    description
      "Identity for ransomware infection event types.";
  }

  identity i2nsf-ipsec {
    description
      "Base identity for IPsec method types.";
  }
  identity ipsec-ike {
    base i2nsf-ipsec;
    description
      "Identity for ipsec-ike.";
  }

  identity ipsec-ikeless {
    base i2nsf-ipsec;
    description
      "Identity for ipsec-ikeless.";
  }

  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";
    }

    identity certificate-type {
      description
      "Base Identity for certificate-type.
      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.";
    }
    identity cer {
      base certificate-type;
      description
      "Identity for '.cer' certificates.";
    }
    identity crt {
```

```
    base certificate-type;
    description
    "Identity for '.crt' certificates.";
  }
  identity key {
    base certificate-type;
    description
    "Identity for '.key' certificates.";
  }

  identity enforce-type {
    description
    "This identity represents the event of
    policy enforcement trigger type.";
  }
  identity admin {
    base enforce-type;
    description
    "The identity for policy enforcement by admin.";
  }
  identity time {
    base enforce-type;
    description
    "The identity for policy enforcement based on time.";
  }

  identity protocol-type {
    description
    "This identity represents the protocol types.";
  }
  identity ftp {
    base protocol-type;
    description
    "The identity for ftp protocol.";
  }
  identity ssh {
    base protocol-type;
    description
    "The identity for ssh protocol.";
  }
  identity telnet {
    base protocol-type;
    description
    "The identity for telnet.";
  }
  identity smtp {
    base protocol-type;
    description
```

```
    "The identity for smtp.";
  }
  identity sftp {
    base protocol-type;
    description
    "The identity for sftp.";
  }
  identity http {
    base protocol-type;
    description
    "The identity for http.";
  }
  identity https {
    base protocol-type;
    description
    "The identity for https.";
  }
  identity pop3 {
    base protocol-type;
    description
    "The identity for pop3.";
  }
  identity nat {
    base protocol-type;
    description
    "The identity for nat.";
  }

  identity primary-action {
    description
    "This identity represents the primary actions, such as
    PASS, DROP, ALERT, RATE-LIMIT, and MIRROR.";
  }
  identity pass {
    base primary-action;
    description
    "The identity for pass.";
  }
  identity drop {
    base primary-action;
    description
    "The identity for drop.";
  }
  identity alert {
    base primary-action;
    description
    "The identity for alert.";
  }
```

```
  identity rate-limit {
    base primary-action;
    description
    "The identity for rate-limit.";
  }
  identity mirror {
    base primary-action;
    description
    "The identity for mirroring.";
  }

  identity secondary-action {
    description
    "This field identifies additional actions if a rule is
    matched. This could be one of 'LOG', 'SYSLOG',
    'SESSION-LOG', etc.";
  }
  identity log {
    base secondary-action;
    description
    "The identity for logging.";
  }
  identity syslog {
    base secondary-action;
    description
    "The identity for system logging.";
  }
  identity session-log {
    base secondary-action;
    description
    "The identity for session logging.";
  }

  identity role-type {
    description
    "This is the base identity for the roles.";
  }
  identity user {
    base role-type;
    description
    "This represents the identity of the user role.";
  }
  identity group {
    base role-type;
    description
    "This represents the identity of any member of the
    security policy's defined group.";
  }
```

```
  identity other {
    base role-type;
    description
    "This represents the identity of anyone else.";
  }
  identity all {
    base role-type;
    description
    "This represents the identity of everyone
    (i.e., user, group, and other).";
  }

  identity owner {
    description
    "This is the base identity for the owner";
  }
  identity dept-head {
    base owner;
    description
    "This represents the identity of the head of department.";
  }
  identity manager {
    base owner;
    description
    "This represents the identity of the manager of the department.";
  }
  identity employee {
    base owner;
    description
    "This represents the identity of department employees.";
  }
  identity sec-head {
    base owner;
    description
    "This represents the identity of the head of security.";
  }
  identity sec-admin {
    base owner;
    description
    "This represents the identity of security admin.";
  }

  identity tenant-type {
    description
    "This is the base identity for the tenants
    to represent the ownership of the security policies.";
  }
  identity human-resources {
```

```
    base tenant-type;
    description
    "This represents the identity of the human resources
    department or division.";
  }
  identity marketing {
    base tenant-type;
    description
    "This represents the identity of the marketing
    department or division.";
  }
  identity customer-service {
    base tenant-type;
    description
    "This represents the identity of customer service
    department or division.";
  }
  identity research {
    base tenant-type;
    description
    "This represents the identity of research
    department or division.";
  }
  identity finance {
    base tenant-type;
    description
    "This represents the identity of finance
    department or division.";
  }

  identity domain {
    description
    "This represents the base identity of different domains.";
  }
  identity enterprise {
    base domain;
    description
    "This represents the identity of an enterprise domain.";
  }

  identity signature-type {
    description
    "This represents the base identity for signature types.";
  }
  identity signature-yara {
    base signature-type;
    description
    "This represents the YARA signatures.";
```

```
  }
  identity signature-snort {
    base signature-type;
    description
    "This represents the SNORT signatures.";
  }
  identity signature-suricata {
    base signature-type;
    description
    "This represents the SURICATA signatures.";
  }

  identity threat-feed-type {
    description
    "This represents the base identity for threat-feed.";
  }
  identity palo-alto {
    base threat-feed-type;
    description
    "This represents Palo-Alto threat-feed.";
  }
  identity rsa-netwitness {
    base threat-feed-type;
    description
    "This represents RSA-netwitness threat-feed.";
  }
  identity fireeye {
    base threat-feed-type;
    description
    "This represents FireEye threat-feed.";
  }
  identity alienvault {
    base threat-feed-type;
    description
    "This represents Alienvault threat-feed.";
  }
  identity auth-type {
  description
  "The base identity for authentication type.";
  }
  identity auth-type-server {
    base auth-type;
    description
    "This represents the server authentication.";
  }
  identity auth-type-client {
    base auth-type;
    description
```

```
    "This represents the client authentication.";
  }
  identity auth-type-mutual {
    base auth-type;
    description
    "This represents the both server and client
    authentication.";
  }

  identity auth-method-type {
    description
    "Base idendity for authentication-methods";
  }

  identity password-based {
    base auth-method-type;
    description
    "This is the identity for the password-based authetication type.";
  }
  identity token-based {
    base auth-method-type;
    description
    "This is the identity for the token-based authetication type.";
  }
  identity certificate-based {
    base auth-method-type;
    description
    "This is the identity for the certificate-based authetication type.";
  }

/*
 * Groupings
 */

grouping ipv4-list {
  description
  "Grouping for ipv4 based ip-addresses.";
  leaf-list ipv4 {
    type inet:ipv4-address;
    description
    "This is the entry for the ipv4 ip-addresses.";
  }
}

grouping ipv6-list {
  description
  "Grouping for ipv6 based ip-addresses.";
  leaf-list ipv6 {
```

```
    type inet:ipv6-address;
    description
    "This is the entry for the ipv6 ip-addresses.";
  }
}

grouping ipv4 {
  description
  "Grouping for ipv4 based ip-address.";
  leaf ipv4 {
    type inet:ipv4-address;
    description
    "This is the entry for the ipv4 ip-address.";
  }
}

grouping ipv6 {
  description
  "Grouping for ipv6 based ip-address.";
  leaf ipv6 {
    type inet:ipv6-address;
    description
    "This is the entry for the ipv6 ip-address.";
  }
}
grouping ip-address-info {
  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-ipv4 {
        uses ipv4;
        description
        "Exact ip-address match for ipv4 type addresses";
      }
      case exact-match-ipv6 {
        uses ipv6;
        description
        "Exact ip-address match for ipv6 type addresses";
      }
      case range-match-ipv4 {
        list range-ipv4-address {
          key "start-ipv4-address end-ipv4-address";
          leaf start-ipv4-address {
            type inet:ipv4-address;
            description
```

```
              "Start IPv4 address for a range match.";
          }
          leaf end-ipv4-address {
            type inet:ipv4-address;
            description
              "End IPv4 address for a range match.";
          }
          description
            "Range match for an IP-address.";
        }
      }
      case range-match-ipv6 {
        list range-ipv6-address {
          key "start-ipv6-address end-ipv6-address";
          leaf start-ipv6-address {
            type inet:ipv6-address;
            description
              "Start IPv6 address for a range match.";
          }
          leaf end-ipv6-address {
            type inet:ipv6-address;
            description
              "End IPv6 address for a range match.";
          }
          description
            "Range match for an IP-address.";
        }
      }
    }
}

grouping password-based-method {
  list password-list {
    key "auth-method";
    leaf auth-method {
      type identityref {
        base auth-method-type;
      }
      description
      "This represents the authentication method is password-based.";
    }
    leaf password {
      type ianach:crypt-hash;
      description
      "The password for this entry.";
    }
    description
    "This represents the list of
```

```
    encrypted passwords.";
  }
}

grouping certificate-based-method {
  list cert-server-list {
    key "auth-method";
    description
    "This describes the certificate-based authentication list.";

    leaf auth-mthod {
      type identityref {
        base auth-method-type;
      }
      description
      "This represents the authentication method is
      certificate based method.";
    }
    leaf cert-server-name {
      type string;
      description
      "This field represents the name of the certificate-
      server name.";
    }
    leaf cert-server-ipv4 {
      type inet:ipv4-address;
      description
      "This represents ipv4 address of a
      certificate server.";
    }
    leaf cert-server-ipv6 {
      type inet:ipv6-address;
      description
      "This represents the ipv6 address of a
      certificate server.";
    }
    list certificate {
      key "cert-type";
      description
      "This represents the certificate-types.";

      leaf cert-type {
        type identityref {
          base certificate-type;
        }
        description
        "This represents a certificate type.";
      }
```

```
    }
  }
}

grouping token-based-method {
  list token-list {
    key "auth-method";
    description
    "This represents the list of tokens.";

    leaf auth-method {
      type identityref {
        base auth-method-type;
      }
      description
      "This represents the authentication type is
      token-based method.";
    }
    leaf token {
      type string;
      description
      "This object contains a string of a token.";
    }
    leaf token-server {
      type inet:ipv4-address;
      description
      "This represents the token-server information.";
    }
  }
}
grouping ipsec-based-method {
  list ipsec-method {
    key "method";
    description
    "This represents the list of IPsec method types.";

    leaf method {
      type identityref {
        base i2nsf-ipsec;
      }
      description
      "This represents IPsec IKE and IPsec IKEless cases.";
    }
  }
}

grouping user-group {
  description
```

```
    "The grouping for user-group entities, and
    contains information such as name & ip-address.";
  leaf name {
    type string;
    description
    "This represents the name of a user.";
  }
  uses ip-address-info;
}

grouping device-group {
  description
    "This group represents device group information
    such as ip-address protocol.";
  leaf name {
    type string;
    description
    "This represents the name of a device.";
  }
  uses ip-address-info;
  leaf-list protocol {
    type identityref {
      base protocol-type;
    }
    description
    "This represents the communication protocols of devices.";
  }
}

grouping location-group {
  description
    "This group represents location-group information
    such as geo-ip and continent.";
  leaf name {
    type string;
    description
    "This represents the name of a location.";
  }
  leaf geo-ip-ipv4 {
    type inet:ipv4-address;
    description
    "This represents the IPv4 geo-ip of a location.";
  }
  leaf geo-ip-ipv6 {
    type inet:ipv6-address;
    description
    "This represents the IPv6 geo-ip of a location.";
  }
```

```
  leaf continent {
    type identityref {
      base continent;
        }
    description
      "location-group-based on geo-ip of
      respective continent.";
  }
}

grouping threat-feed-info {
  description
  "This is the grouping for the threat-feed-list";

  leaf feed-name {
    type identityref {
      base threat-feed-type;
    }
    description
    "This represents the name of the a threat-feed.";
  }
  leaf feed-server-ipv4 {
    type inet:ipv4-address;
    description
    "The IPv4 ip-address for the threat-feed server.";
  }
  leaf feed-server-ipv6 {
    type inet:ipv6-address;
    description
    "The IPv6 ip-address for the threat-feed server.";
  }
  leaf feed-description {
    type string;
    description
    "This represents the descriptions of a threat-feed.
    The description should include information, such as
    the type, related threat, method, and file type.";
  }
}

grouping payload-string {
  description
    "The grouping for payload-string content.
    It contains information such as name and string content.";
  leaf payload-description {
    type string;
    description
    "This represents the description of a payload.";
```

```
  }
  leaf-list content {
    type string;
    description
    "This represents the payload string content.";
  }
}

list i2nsf-cfi-policy {
    key "policy-name";
    description
    "This is the security policy list. Each policy in the list
    contains a list of security rules, and is a policy instance
    to have complete information such as where and when a
    policy needs to be applied.";
    leaf policy-name {
      type string;
      mandatory true;
      description
        "The name which identifies the policy.";
    }
    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.";

      container event {
        description
         "This represents the event (e.g., a security event, which a security
rule is made for.";
        leaf security-event {
          type identityref {
            base security-event-type;
          }
          mandatory true;
          description
            "This contains the description of security events.";
        }
        choice enforce-type {
          description
            "There are three different enforcement types;
            admin, and time.";
          case enforce-admin {
            leaf admin {
              type identityref {
                base enforce-type;
              }
            description
            "This represents the enforcement type based on admin's decision.";
            }
          }
          case time {
            container time-information {
              description
                "The begin-time and end-time information
                when the security rule should be applied.";
              leaf enforce-time {
                type identityref {
                  base enforce-type;
                }
                description
                "The enforcement type is time-enforced.";
              }
              leaf begin-time {
                type yang:date-and-time;
                description
                  "This is start time for time zone";
              }
              leaf end-time {
                type yang:date-and-time;
                description
                  "This is end time for time zone";
              }
            }
          }
        }
        leaf frequency {
          type enumeration {
            enum only-once {
              description
              "This represents the rule is enforced only once.";
            }
            enum daily {
              description
              "This represents the rule is enforced on a daily basis.";
            }
            enum weekly {
              description
              "This represents the rule is enforced on a weekly basis.";
            }
            enum monthly {
```

```
              description
              "This represents the rule is enforced on a monthly basis.";
            }
          }
          default only-once;
          description
          "This represents how frequent the rule should be enforced.";
        }
      }
      container condition {
        choice condition {
          description
            "The conditions for general security policies.";
              case firewall-condition {
                description
                  "The general firewall condition.";
                container firewall-source {
                  description
                  "This represents the source.";
                  leaf src-target {
                    type leafref {
                      path "/i2nsf-cfi-policy/endpoint-group/user-group/name";
                    }
                    mandatory true;
                    description
                    "This describes the paths to
                    the source reference.";
                  }
                }
                container firewall-destination {
                  description
                    "This represents the destination.";
                    leaf-list dest-target {
                      type leafref {
                        path "/i2nsf-cfi-policy/endpoint-group/user-group/
name";
                      }
                      description
                        "This describes the paths to the
                        destination target reference.";
                    }
                  }
              }
              case ddos-condition {
                description
                  "The condition for DDoS mitigation.";
                container ddos-source {
                  description
```

```
                    "This represents the source.";
```

```
                  leaf-list src-target {
                    type leafref {
                      path "/i2nsf-cfi-policy/endpoint-group/device-group/
name";
                    }
                    description
                      "This describes the path to the
                      source target references.";
                  }
                }
                container ddos-destination {
                  description
                    "This represents the target.";
                  leaf-list dest-target {
                    type leafref {
                      path "/i2nsf-cfi-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 uint16;
                    description
                    "The rate-limit limits the amount of incoming packets.";
                  }
                }
              }
              case custom-condition {
                description
                  "The condition based on packet contents.";
                container custon-source {
                  description
                    "This represents the source.";
                  leaf-list src-target {
                    type leafref {
                      path "/i2nsf-cfi-policy/threat-prevention/payload-
content/name";
                    }
                    description
                      "Describes the payload string
                      content condition source.";
                  }
                }
```

```
            container custom-destination {
              description
                "This represents the destination.";
```

```
                  leaf dest-target {
                    type leafref {
                      path "/i2nsf-cfi-policy/threat-prevention/payload-
content/name";
                    }
                    mandatory true;
                    description
                      "Describes the payload string
                    content condition destination.";
                  }
                }
              }
              case threat-feed-condition {
                description
                  "The condition based on the threat-feed information.";
                container threat-feed-source {
                  description
                    "This represents the source.";
                  leaf-list src-target {
                    type leafref {
                      path "/i2nsf-cfi-policy/threat-prevention/threat-feed-
list/feed-name";
                    }
                    description "Describes the threat-feed
                    condition source.";
                  }
                }
                container threat-feed-destination {
                  description
                    "This represents the destination.";
                  leaf dest-target {
                    type leafref {
                      path "/i2nsf-cfi-policy/threat-prevention/threat-feed-
list/feed-name";
                    }
                    mandatory true;
                    description "Describes the threat-feed
                    condition destination.";
                  }
                }
              }
        }
      }
      container action {
        description
          "This is the action container.";
        leaf primary-action {
          type identityref {
            base primary-action;
          }
          mandatory true;
          description
          "This represent the primary actions (e.g., PASS, DROP,
          ALERT, and MIRROR) to be applied a condition.";
        }
        leaf secondary-action {
          type identityref {
            base secondary-action;
          }
          description
          "This represents the secondary actions (e.g., log
          and syslog) to be applied if needed.";
        }
      }
      container ipsec-method {
        description
          "This container represents the IPsec IKE and IKEless cases.";
        leaf method {
          type leafref {
           path "/i2nsf-cfi-policy/multi-tenancy/policy-mgmt-auth-method-
instance/ipsec-method/method";
          }
          description
            "This references the IPsec method types,
             which includes IPsec IKE and IPsec IKEless cases.";
        }
      }
      leaf owner {
        type identityref {
          base owner;
        }
        mandatory true;
        description
         "This field defines the owner of this
         rule. Only the owner is authorized to
         modify the contents of the rule.";
      }
    }

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 {
      key "domain-name";
      description
      "This represents the list of policy domains.";
      leaf domain-name {
        type identityref {
          base domain;
        }
        description
        "This represents the name of a domain.";
      }
      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 {
        key "tenant-type";
        description
        "This field identifies the domain to which this
        tenant belongs. This should be reference to a
        'Policy-Domain' object.";

        leaf tenant-type{
          type identityref {
            base tenant-type;
          }
          description
          "The name of the tenant, such as HR or Finance department.";
        }
        list policy-role {
          key "role-type";
          description
          "This represent the roles within the tenants,
          in order to distinguish who may or may not
          have access to policies.";

          leaf role-type {
            type identityref {
              base role-type;
            }
            description
            "This represents the name of the role";
          }
          list policy-user {
```

```
            key "name";
            description
            "This represents the list of policy users.";

            leaf name {
              type string;
              description
              "This represents the name of the user";
            }
            leaf password {
              type ianach:crypt-hash;
              description
                "User password for basic authentication";
            }
            leaf email {
              type string;
              description
              "The email account of a user";
            }
            list access-profile {
              key "permission-type scope-type";
              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;
                }
                description
                "This represents the permission types, such as
                read, write, execute, read-and-write, and etc.";
              }
              leaf scope-type {
                type identityref {
                  base scope-type;
                }
                description
                  "identifies whether a user has domain-wide
                  or tenant-wide privileges";
              }
            }
          }
        }
      }
    }
    list policy-mgmt-auth-method-instance {
      key "auth-instance-type";
```

```
      description
      "This represents the list of instances for
      policy management authentication methods.";

      leaf auth-instance-type {
        type identityref {
          base auth-type;
        }
        description
        "This identifies whether the authentication type
        is server authentication, client authentication,
        or both.";
      }
      choice policy-mgmt-auth-method {
        description
        "This represents the choices for which
        authentication method is used.";
        case password-based {
        uses password-based-method;
        }
        case token-based {
          description
          "This represents the token-based method.";
          uses token-based-method;
        }
        case certificate-based {
          description
          "This represents the certificate-based-method.";
          uses certificate-based-method;
        }
        case ipsec {
          description
          "This repreents authentication method based on IPSEC.";
          uses ipsec-based-method;
        }
      }
    }
}
container endpoint-group {
  description
    "A logical entity in their business
    environment, where a security policy
    is to be applied.";
  list user-group {
    key "name";
    uses user-group;
    description
      "This represents the user group.";
```

```
  }
  list device-group {
    key "name";
    uses device-group;
    description
      "This represents the device group.";
  }
  list location-group{
    key "name";
    uses location-group;
    description
      "This represents the location group.";
  }
}
container threat-prevention {
      description
      "this describes the list of threat-prevention.";

      list threat-feed-list {
        key "feed-name";
        description
        "This represents the threat feed list.";
        uses threat-feed-info;

        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 identityref {
            base signature-type;
          }
          default signature-suricata;
          description
          "This contains a list of signatures or hash
          of the threats.";
        }
      }
      list payload-content {
          key "name";
          leaf name {
            type string;
            decription
```

```
            "This represents the name of payload-content".
            It should give an idea of why specific payload
            content is marked as threat. For example, the name
            "backdoor" indicates the payload content is related
            to backdoor attack.";
          }
          description
          "This represents the payload-string group.";
          uses payload-string;
      }
    }
  }
}
<CODE ENDS>
```

Figure 23: YANG for Consumer-Facing Interface

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)

If new endpoints are introduced to the network, it is necessary to first register their data to the database. For example, if new members are newly introduced in either of three different groups (i.e., user-group, device-group, and payload-group), each of them should be registered with information 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" ?>
        <endpoint-group xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-
policy">
          <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>
        </endpoint-group xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-
policy">
```

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" ?>
<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-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>
    <ipsec-method>
      <method>ipsec-ike</method>
    </ipsec-method>
  </rule>
</policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-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.

6. The IPsec method type used for nsf traffic steering is set to "ipsec-ike".

10.3. Scenario 2: Block Malicious VoIP/VoLTE Packets Coming to a Company

The second example scenario is to "block malicious VoIP/VoLTE packets coming to a 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" ?>
<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-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>
    <ipsec-method>
      <method>ipsec-ikeless</method>
    </ipsec-method>
  </rule>
</policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-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".

6. The IPsec method used for nsf traffic steering is set to "ipsec-ikeless".

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" ?>
<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-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>
    <ipsec-method>
      <method>ipsec-ikeless</method>
    </ipsec-method>
  </rule>
</policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-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.

7. The IPsec method used for nsf traffic steering is set to "ipsec-ike".

11. Security Considerations

The data model for the I2NSF Consumer-Facing Interface is based on the I2NSF framework [RFC8329], so the same security considerations with the I2NSF framework should be included in this document. The data model needs a secure communication channel to protect the Consumer-Facing Interface between the I2NSF User and 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. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[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., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

13.2. Informative References

[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.

[i2nsf-capability-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-05 (work in progress), April 2019.

[i2nsf-ipsec]
Marin-Lopez, R., Lopez-Millan, G., and F. Pereniguez-Garcia, "Software-Defined Networking (SDN)-based IPsec Flow Protection", draft-ietf-i2nsf-sdn-ipsec-flow-protection-05 (work in progress), July 2019.

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

Appendix A. Changes from draft-ietf-i2nsf-consumer-facing-interface-dm-05

The following are major changes made from draft-ietf-i2nsf-consumer-facing-interface-dm-05:

- The container policy-mgnt-auth-method uses a list, and the policy-mgmt-auth-method consists of choice-cases.
- Policy-role is changed from container to list. The access-profile in the policy-role is not removed. Instead, it is placed inside policy-user.
- Container Condition consists of choice-cases to show that it is capable of configuring different triggering conditions.
- The enforce-type in Event container use a choice-case statement. This change shows the clarity that the enforce-type is relevant to each case (i.e., enforce-type == admin or time).
- The name for container "recursive" is changed to "frequency". This container represents how frequently the rule is enforced, so the name "frequency" is more appropriate.
- The certificate based authentication method is modified so that a certificate server can handle more than one (list) of certificate types.

The minor changes are as follows:

- Typos are corrected.
- IPv6 as well as IPv4 are included.
- Some misused types are corrected (e.g., enum -> identity)
- Some descriptions that are unclear, mistaken, or shortly explained are rewritten.

Appendix B. Acknowledgments

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

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 Computer Science and Engineering
Sungkyunkwan University
2066 Seo-ro Jangan-gu
Suwon, Gyeonggi-do 16419
Republic of Korea

EMail: hyoung@skku.edu

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

EMail: jine33@skku.edu

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

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

Eunsoo Kim
Department of Electronic, 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