

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

J. Jeong
Sungkyunkwan University
S. Hyun
Chosun University
T. Ahn
Korea Telecom
S. Hares
Huawei
D. Lopez
Telefonica I+D
March 11, 2019

Applicability of Interfaces to Network Security Functions to Network-
Based Security Services
draft-ietf-i2nsf-applicability-09

Abstract

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

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. I2NSF Framework	5
4. Time-dependent Web Access Control Service	6
5. I2NSF Framework with SFC	8
6. I2NSF Framework with SDN	10
6.1. Firewall: Centralized Firewall System	13
6.2. Deep Packet Inspection: Centralized VoIP/VoLTE Security System	14
6.3. Attack Mitigation: Centralized DDoS-attack Mitigation System	16
7. I2NSF Framework with NFV	19
8. Security Considerations	20
9. Acknowledgments	20
10. Contributors	21
11. References	21
11.1. Normative References	21
11.2. Informative References	22
Appendix A. Changes from draft-ietf-i2nsf-applicability-08 . . .	25
Authors' Addresses	25

1. Introduction

Interface to Network Security Functions (I2NSF) defines a framework and interfaces for interacting with Network Security Functions (NSFs). Note that Network Security Function (NSF) is defined as a functional block for a security service within an I2NSF framework that has well-defined I2NSF NSF-facing interface and other external interfaces and well-defined functional behavior [NFV-Terminology].

The I2NSF framework allows heterogeneous NSFs developed by different security solution vendors to be used in the Network Functions Virtualization (NFV) environment [ETSI-NFV] by utilizing the capabilities of such products and the virtualization of security functions in the NFV platform. In the I2NSF framework, each NSF initially registers the profile of its own capabilities into the system in order for themselves to be available in the system. In addition, the Security Controller is validated by the I2NSF User

(also called I2NSF Client) that a system administrator (as a user) is employing, so that the system administrator can request security services through the Security Controller.

This document illustrates the applicability of the I2NSF framework with four different scenarios:

1. The enforcement of time-dependent web access control.
2. The application of I2NSF to a Service Function Chaining (SFC) environment [RFC7665].
3. The integration of the I2NSF framework with Software-Defined Networking (SDN) [RFC7149] to provide different security functionality such as firewalls [opsawg-firewalls], Deep Packet Inspection (DPI), and Distributed Denial of Service (DDoS) attack mitigation.
4. The use of Network Functions Virtualization (NFV) [ETSI-NFV] as a supporting technology.

The implementation of I2NSF in these scenarios has allowed us to verify the applicability and effectiveness of the I2NSF framework for a variety of use cases.

2. Terminology

This document uses the terminology described in [RFC7665], [RFC7149], [ITU-T.Y.3300], [ONF-OpenFlow], [ONF-SDN-Architecture], [ITU-T.X.1252], [ITU-T.X.800], [NFV-Terminology], [RFC8329], [i2nsf-terminology], [consumer-facing-inf-dm], [i2nsf-nsf-cap-im], [nsf-facing-inf-dm], [registration-inf-dm], and [nsf-triggered-steering]. In addition, the following terms are defined below:

- o Software-Defined Networking (SDN): A set of techniques that enables to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].
- o Network Function: A functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behavior [NFV-Terminology].
- o Network Security Function (NSF): A functional block within a security service within a network infrastructure that has well-

defined external interfaces and well-defined functional behavior[NFV-Terminology].

- o Network Functions Virtualization (NFV): A principle of separating network functions (or network security functions) from the hardware they run on by using virtual hardware abstraction [NFV-Terminology].
- o Service Function Chaining (SFC): The execution of an ordered set of abstract service functions (i.e., network functions) according to ordering constraints that must be applied to packets, frames, and flows selected as a result of classification. The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied [RFC7665].
- o Firewall: A service function at the junction of two network segments that inspects some suspicious packets that attempt to cross the boundary. It also rejects any packet that does not satisfy certain criteria for, for example, disallowed port numbers or IP addresses.
- o Centralized Firewall System: A centralized firewall that can establish and distribute policy rules into network resources for efficient firewall management.
- o Centralized VoIP Security System: A centralized security system that handles the security functions required for VoIP and VoLTE services.
- o Centralized DDoS-attack Mitigation System: A centralized mitigator that can establish and distribute access control policy rules into network resources for efficient DDoS-attack mitigation.

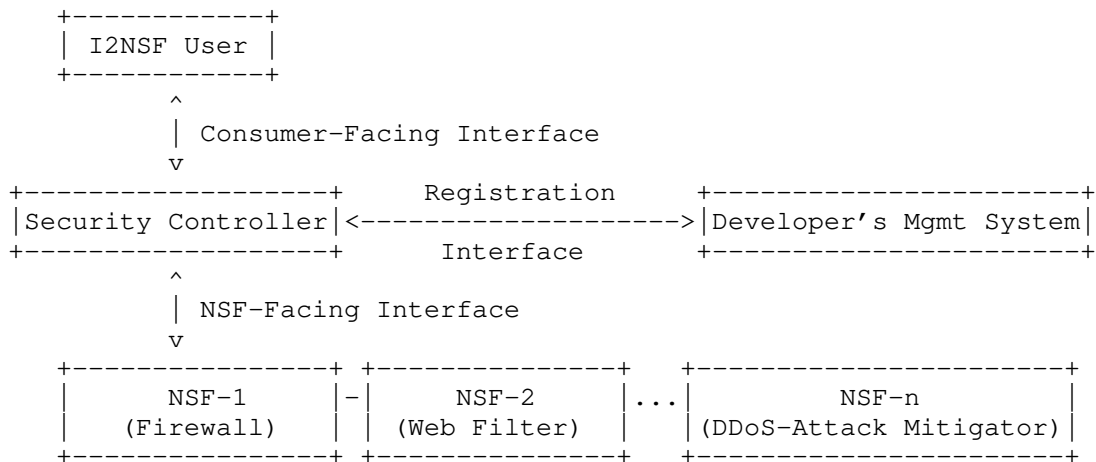


Figure 1: I2NSF Framework

3. I2NSF Framework

This section summarizes the I2NSF framework as defined in [RFC8329]. As shown in Figure 1, an I2NSF User can use security functions by delivering high-level security policies, which specify security requirements that the I2NSF user wants to enforce, to the Security Controller via the Consumer-Facing Interface [consumer-facing-inf-dm].

The Security Controller receives and analyzes the high-level security policies from an I2NSF User, and identifies what types of security capabilities are required to meet these high-level security policies. The Security Controller then identifies NSFs that have the required security capabilities, and generates low-level security policies for each of the NSFs so that the high-level security policies are eventually enforced by those NSFs [policy-translation]. Finally, the Security Controller sends the generated low-level security policies to the NSFs [i2nsf-nsf-cap-im][nsf-facing-inf-dm].

The Security Controller requests NSFs to perform low-level security services via the NSF-Facing Interface. As shown in Figure 1, with a Developer's Management System (DMS), developers (or vendors) inform the Security Controller of the capabilities of the NSFs through the I2NSF Registration Interface [registration-inf-dm] for registering (or deregistering) the corresponding NSFs. Note that an inside attacker at the DMS can seriously weaken the I2NSF system's security. To deal with this type of threat, the role of the DMS should be restricted to providing an I2NSF system with the software package/image for NSF execution, and the DMS should never be able to access

NSFs in online/activated status for the I2NSF system's security. On the other hand, an access to running (online) NSFs should be allowed only to the Security Controller, not the DMS. Also, the Security Controller can detect and prevent inside attacks by monitoring the activity of all the DMSs as well as the NSFs through the I2NSF NSF monitoring functionality [nsf-monitoring-dm].

The Consumer-Facing Interface between an I2NSF User and the Security Controller can be implemented using, for example, RESTCONF [RFC8040]. Data models specified by YANG [RFC6020] describe high-level security policies to be specified by an I2NSF User. The data model defined in [consumer-facing-inf-dm] can be used for the I2NSF Consumer-Facing Interface.

The NSF-Facing Interface between the Security Controller and NSFs can be implemented using NETCONF [RFC6241]. YANG data models describe low-level security policies for the sake of NSFs, which are translated from the high-level security policies by the Security Controller. The data model defined in [nsf-facing-inf-dm] can be used for the I2NSF NSF-Facing Interface.

The Registration Interface between the Security Controller and the Developer's Management System can be implemented by RESTCONF [RFC8040]. The data model defined in [registration-inf-dm] can be used for the I2NSF Registration Interface.

Also, the I2NSF framework can enforce multiple chained NSFs for the low-level security policies by means of SFC techniques for the I2NSF architecture described in [nsf-triggered-steering].

The following sections describe different security service scenarios illustrating the applicability of the I2NSF framework.

4. Time-dependent Web Access Control Service

This service scenario assumes that an enterprise network administrator wants to control the staff members' access to a particular Internet service (e.g., Example.com) during business hours. The following is an example high-level security policy rule for a web filter that the administrator requests: Block the staff members' access to Example.com from 9 AM to 6 PM. Figure 2 is an example XML code for this web filter:

```
<I2NSF>
  <name>block_website</name>
  <cond>
    <src>Staff_Member's_PC</src>
    <dest>Example.com</dest>
    <time-span-start>9:00AM</time-span-start>
    <time-span-end>-6:00PM</time-span-end>
  </cond>
  <action>block</action>
</I2NSF>
```

Figure 2: An XML Example for Time-based Web-filter

The security policy name is "block_website" with the tag "name". The filtering condition has the source group "Staff_Member's_PC" with the tag "src", the destination website "Example.com" with the tag "dest", the filtering start time is the time "9:00AM" with the tag "time-span-start", and the filtering end time is the time "6:00PM" with the tag "time-span-end". The action is to "block" the packets satisfying the above condition, that is, to drop those packets.

After receiving the high-level security policy, the Security Controller identifies required security capabilities, e.g., IP address and port number inspection capabilities and URL inspection capability. In this scenario, it is assumed that the IP address and port number inspection capabilities are required to check whether a received packet is an HTTP packet from a staff member. The URL inspection capability is required to check whether the target URL of a received packet is in the Example.com domain or not.

The Security Controller maintains the security capabilities of each NSF running in the I2NSF system, which have been reported by the Developer's Management System via the Registration interface. Based on this information, the Security Controller identifies NSFs that can perform the IP address and port number inspection and URL inspection [policy-translation]. In this scenario, it is assumed that an NSF of firewall has the IP address and port number inspection capabilities and an NSF of web filter has URL inspection capability.

The Security Controller generates low-level security rules for the NSFs to perform IP address and port number inspection, URL inspection, and time checking. Specifically, the Security Controller may interoperate with an access control server in the enterprise network in order to retrieve the information (e.g., IP address in use, company identifier (ID), and role) of each employee that is currently using the network. Based on the retrieved information, the Security Controller generates low-level security rules to check

whether the source IP address of a received packet matches any one being used by a staff member. In addition, the low-level security rules should be able to determine that a received packet is of HTTP protocol. The low-level security rules for web filter check that the target URL field of a received packet is equal to Example.com. Finally, the Security Controller sends the low-level security rules of the IP address and port number inspection to the NSF of firewall and the low-level rules for URL inspection to the NSF of web filter.

The following describes how the time-dependent web access control service is enforced by the NSFs of firewall and web filter.

1. A staff member tries to access Example.com during business hours, e.g., 10 AM.
 2. The packet is forwarded from the staff member's device to the firewall, and the firewall checks the source IP address and port number. Now the firewall identifies the received packet is an HTTP packet from the staff member.
 3. The firewall triggers the web filter to further inspect the packet, and the packet is forwarded from the firewall to the web filter. SFC technology can be utilized to support such packet forwarding in the I2NSF framework [nsf-triggered-steering].
 4. The web filter checks the target URL field of the received packet, and realizes the packet is toward Example.com. The web filter then checks that the current time is in business hours. If so, the web filter drops the packet, and consequently the staff member's access to Example.com during business hours is blocked.
5. I2NSF Framework with SFC

In the I2NSF architecture, an NSF can trigger an advanced security action (e.g., DPI or DDoS attack mitigation) on a packet based on the result of its own security inspection of the packet. For example, a firewall triggers further inspection of a suspicious packet with DPI. For this advanced security action to be fulfilled, the suspicious packet should be forwarded from the current NSF to the successor NSF. SFC [RFC7665] is a technology that enables this advanced security action by steering a packet with multiple service functions (e.g., NSFs), and this technology can be utilized by the I2NSF architecture to support the advanced security action.

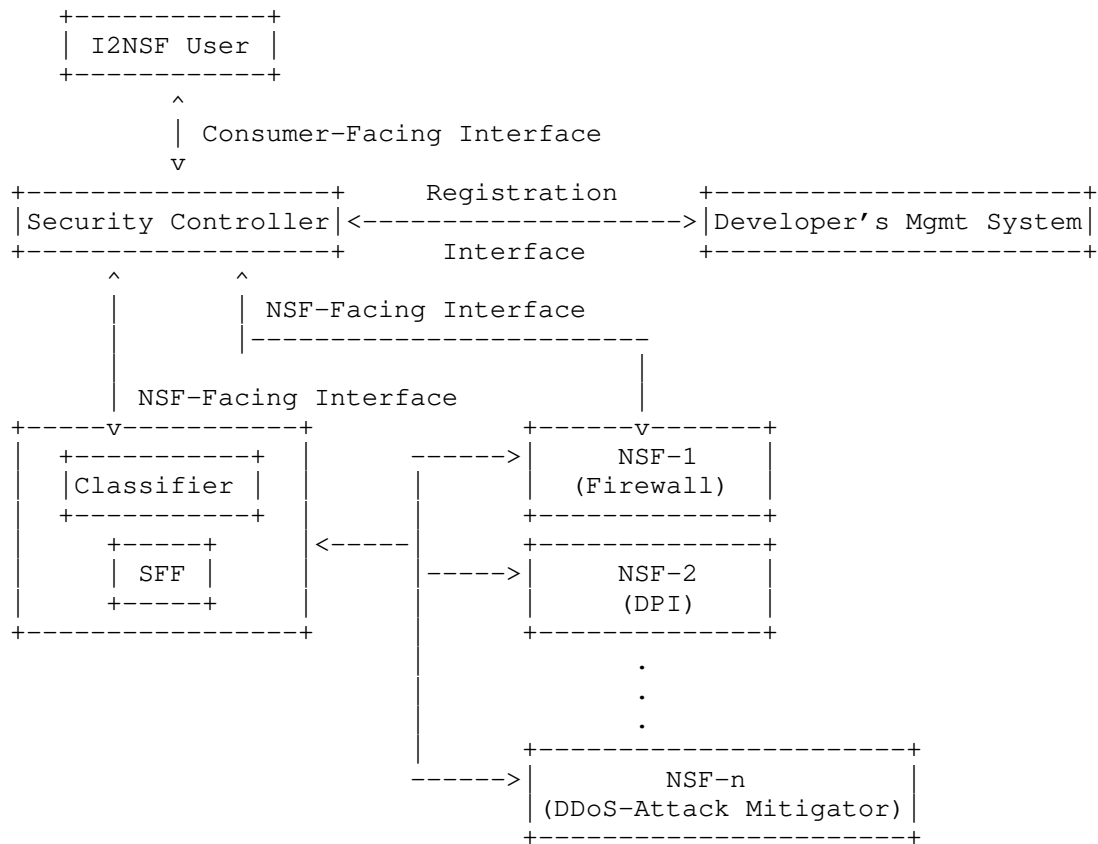


Figure 3: An I2NSF Framework with SFC

Figure 3 shows an I2NSF framework with the support of SFC. As shown in the figure, SFC generally requires classifiers and service function forwarders (SFFs); classifiers are responsible for determining which service function path (SFP) (i.e., an ordered sequence of service functions) a given packet should pass through, according to pre-configured classification rules, and SFFs perform forwarding the given packet to the next service function (e.g., NSF) on the SFP of the packet by referring to their forwarding tables. In the I2NSF architecture with SFC, the Security Controller can take responsibilities of generating classification rules for classifiers and forwarding tables for SFFs. By analyzing high-level security policies from I2NSF users, the Security Controller can construct SFPs that are required to meet the high-level security policies, generates classification rules of the SFPs, and then configures classifiers with the classification rules over NSF-Facing Interface so that relevant traffic packets can follow the SFPs. Also, based on the

global view of NSF instances available in the system, the Security Controller constructs forwarding tables, which are required for SFFs to forward a given packet to the next NSF over the SFP, and configures SFFs with those forwarding tables over NSF-Facing Interface.

To trigger an advanced security action in the I2NSF architecture, the current NSF appends a metadata describing the security capability required for the advanced action to the suspicious packet and sends the packet to the classifier. Based on the metadata information, the classifier searches an SFP which includes an NSF with the required security capability, changes the SFP-related information (e.g., service path identifier and service index [RFC8300]) of the packet with the new SFP that has been found, and then forwards the packet to the SFF. When receiving the packet, the SFF checks the SFP-related information such as the service path identifier and service index contained in the packet and forwards the packet to the next NSF on the SFP of the packet, according to its forwarding table.

6. I2NSF Framework with SDN

This section describes an I2NSF framework with SDN for I2NSF applicability and use cases, such as firewall, deep packet inspection, and DDoS-attack mitigation functions. SDN enables some packet filtering rules to be enforced in network forwarding elements (e.g., switch) by controlling their packet forwarding rules. By taking advantage of this capability of SDN, it is possible to optimize the process of security service enforcement in the I2NSF system.

Figure 4 shows an I2NSF framework [RFC8329] with SDN networks to support network-based security services. In this system, the enforcement of security policy rules is divided into the SDN forwarding elements (e.g., switch running as either a hardware middle box or a software virtual switch) and NSFs (e.g., firewall running in a form of a virtual network function [ETSI-NFV]). Especially, SDN forwarding elements enforce simple packet filtering rules that can be translated into their packet forwarding rules, whereas NSFs enforce NSF-related security rules requiring the security capabilities of the NSFs. For this purpose, the Security Controller instructs the SDN Controller via NSF-Facing Interface so that SDN forwarding elements can perform the required security services with flow tables under the supervision of the SDN Controller.

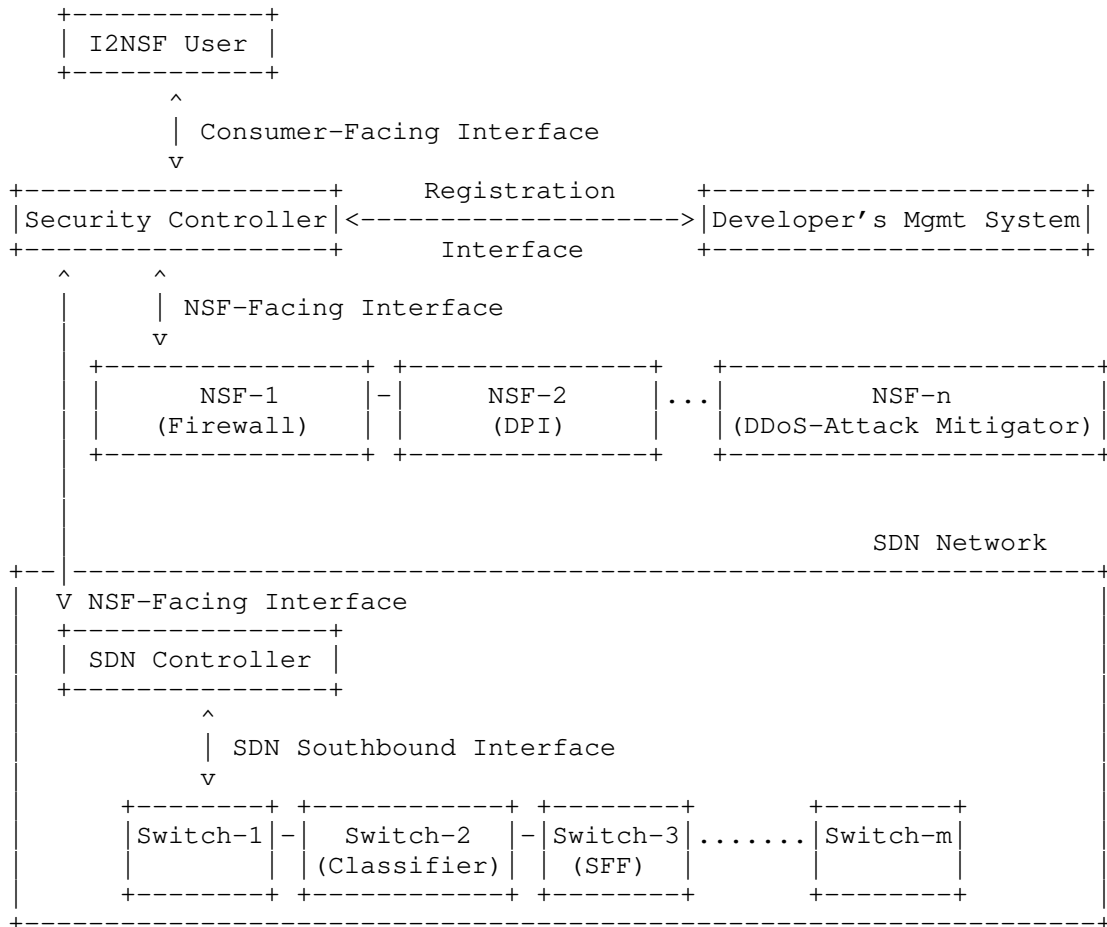


Figure 4: An I2NSF Framework with SDN Network

As an example, let us consider two different types of security rules: Rule A is a simple packet filtering rule that checks only the IP address and port number of a given packet, whereas rule B is a time-consuming packet inspection rule for analyzing whether an attached file being transmitted over a flow of packets contains malware. Rule A can be translated into packet forwarding rules of SDN forwarding elements and thus be enforced by these elements. In contrast, rule B cannot be enforced by forwarding elements, but it has to be enforced by NSFs with anti-malware capability. Specifically, a flow of packets is forwarded to and reassembled by an NSF to reconstruct the attached file stored in the flow of packets. The NSF then analyzes the file to check the existence of malware. If the file contains malware, the NSF drops the packets.

In an I2NSF framework with SDN, the Security Controller can analyze given security policy rules and automatically determine which of the given security policy rules should be enforced by SDN forwarding elements and which should be enforced by NSFs. If some of the given rules requires security capabilities that can be provided by SDN forwarding elements, then the Security Controller instructs the SDN Controller via NSF-Facing Interface so that SDN forwarding elements can enforce those security policy rules with flow tables under the supervision of the SDN Controller. Or if some rules require security capabilities that cannot be provided by SDN forwarding elements but by NSFs, then the Security Controller instructs relevant NSFs to enforce those rules.

The distinction between software-based SDN forwarding elements and NSFs, which can both run as virtual network functions, may be necessary for some management purposes in this system. For this, we can take advantage of the NFV MANO where there is a subsystem that maintains the descriptions of the capabilities each VNF can offer [ETSI-NFV-MANO]. This subsystem can determine whether a given software element (VNF instance) is an NSF or a virtualized SDN switch. For example, if a VNF instance has anti-malware capability according to the description of the VNF, it could be considered as an NSF. A VNF onboarding system [VNF-ONBOARDING] can be used as such a subsystem that maintains the descriptions of each VNF to tell whether a VNF instance is for an NSF or for a virtualized SDN switch.

For the support of SFC in the I2NSF framework with SDN, as shown in Figure 4, network forwarding elements (e.g., switch) can play the role of either SFC Classifier or SFF, which are explained in Section 5. Classifier and SFF have an NSF-Facing Interface with Security Controller. This interface is used to update security service function chaining information for traffic flows. For example, when it needs to update an SFP for a traffic flow in an SDN network, as shown in Figure 4, SFF (denoted as Switch-3) asks Security Controller to update the SFP for the traffic flow (needing another security service as an NSF) via NSF-Facing Interface. This update lets Security Controller ask Classifier (denoted as Switch-2) to update the mapping between the traffic flow and SFP in Classifier via NSF-Facing Interface.

The following subsections introduce three use cases for cloud-based security services: (i) firewall system, (ii) deep packet inspection system, and (iii) attack mitigation system. [RFC8192]

6.1. Firewall: Centralized Firewall System

A centralized network firewall can manage each network resource and apply common rules to individual network elements (e.g., switch). The centralized network firewall controls each forwarding element, and firewall rules can be added or deleted dynamically.

The procedure of firewall operations in this system is as follows:

1. A switch forwards an unknown flow's packet to one of the SDN Controllers.
2. The SDN Controller forwards the unknown flow's packet to an appropriate security service application, such as the Firewall.
3. The Firewall analyzes, typically, the headers and contents of the packet.
4. If the Firewall regards the packet as a malicious one with a suspicious pattern, it reports the malicious packet to the SDN Controller.
5. The SDN Controller installs new rules (e.g., drop packets with the suspicious pattern) into underlying switches.
6. The suspected packets are dropped by these switches.

Existing SDN protocols can be used through standard interfaces between the firewall application and switches
[RFC7149] [ITU-T.Y.3300] [ONF-OpenFlow] [ONF-SDN-Architecture].

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

- o Cost: The cost of adding firewalls to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add firewall on each network resource. To solve this, each network resource can be managed centrally such that a single firewall is manipulated by a centralized server.
- o Performance: The performance of firewalls is often slower than the link speed of network interfaces. Every network resource for firewall needs to check firewall rules according to network conditions. Firewalls can be adaptively deployed among network switches, depending on network conditions in the framework.

- o The management of access control: Since there may be hundreds of network resources in a network, the dynamic management of access control for security services like firewall is a challenge. In the framework, firewall rules can be dynamically added for new malware.
- o The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for firewall within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.
- o Packet-based access mechanism: Packet-based access mechanism is not enough for firewall in practice since the basic unit of access control is usually users or applications. Therefore, application level rules can be defined and added to the firewall system through the centralized server.

6.2. Deep Packet Inspection: Centralized VoIP/VoLTE Security System

A centralized VoIP/VoLTE security system can monitor each VoIP/VoLTE flow and manage VoIP/VoLTE security rules, according to the configuration of a VoIP/VoLTE security service called VoIP Intrusion Prevention System (IPS). This centralized VoIP/VoLTE security system controls each switch for the VoIP/VoLTE call flow management by manipulating the rules that can be added, deleted or modified dynamically.

The centralized VoIP/VoLTE security system can cooperate with a network firewall to realize VoIP/VoLTE security service. Specifically, a network firewall performs the basic security check of an unknown flow's packet observed by a switch. If the network firewall detects that the packet is an unknown VoIP call flow's packet that exhibits some suspicious patterns, then it triggers the VoIP/VoLTE security system for more specialized security analysis of the suspicious VoIP call packet.

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

1. A switch forwards an unknown flow's packet to the SDN Controller, and the SDN Controller further forwards the unknown flow's packet to the Firewall for basic security inspection.
2. The Firewall analyzes the header fields of the packet, and figures out that this is an unknown VoIP call flow's signal packet (e.g., SIP packet) of a suspicious pattern.

3. The Firewall triggers an appropriate security service function, such as VoIP IPS, for detailed security analysis of the suspicious signal packet. In order for this triggering of VoIP IPS to be served, the suspicious packet is sent to the Service Function Forwarder (SFF) that is usually a switch in an SDN network, as shown in Figure 4. The SFF forwards the suspicious signal packet to the VoIP IPS.
4. The VoIP IPS analyzes the headers and contents of the signal packet, such as calling number and session description headers [RFC4566].
5. If, for example, the VoIP IPS regards the packet as a spoofed packet by hackers or a scanning packet searching for VoIP/VoLTE devices, it drops the packet. In addition, the VoIP IPS requests the SDN Controller to block that packet and the subsequent packets that have the same call-id.
6. The SDN Controller installs new rules (e.g., drop packets) into underlying switches.
7. The malicious packets are dropped by these switches.

Existing SDN protocols can be used through standard interfaces between the VoIP IPS application and switches [RFC7149][ITU-T.Y.3300][ONF-OpenFlow][ONF-SDN-Architecture].

Legacy hardware based VoIP IPS has some challenges, such as provisioning time, the granularity of security, expensive cost, and the establishment of policy. The I2NSF framework can resolve the challenges through the above centralized VoIP/VoLTE security system based on SDN as follows:

- o Provisioning: The provisioning time of setting up a legacy VoIP IPS to network is substantial because it takes from some hours to some days. By managing the network resources centrally, VoIP IPS can provide more agility in provisioning both virtual and physical network resources from a central location.
- o The granularity of security: The security rules of a legacy VoIP IPS are compounded considering the granularity of security. The proposed framework can provide more granular security by centralizing security control into a switch controller. The VoIP IPS can effectively manage security rules throughout the network.
- o Cost: The cost of adding VoIP IPS to network resources, such as routers, gateways, and switches is substantial due to the reason that we need to add VoIP IPS on each network resource. To solve

this, each network resource can be managed centrally such that a single VoIP IPS is manipulated by a centralized server.

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

6.3. Attack Mitigation: Centralized DDoS-attack Mitigation System

A centralized DDoS-attack mitigation can manage each network resource and configure rules to each switch for DDoS-attack mitigation (called DDoS-attack Mitigator) on a common server. The centralized DDoS-attack mitigation system defends servers against DDoS attacks outside the private network, that is, from public networks.

Servers are categorized into stateless servers (e.g., DNS servers) and stateful servers (e.g., web servers). For DDoS-attack mitigation, the forwarding of traffic flows in switches can be dynamically configured such that malicious traffic flows are handled by the paths separated from normal traffic flows in order to minimize the impact of those malicious traffic on the servers. This flow path separation can be done by a flow forwarding path management scheme based on [AVANT-GUARD]. This management should consider the load balance among the switches for the defense against DDoS attacks.

The procedure of DDoS-attack mitigation in this system is as follows:

1. A Switch periodically reports an inter-arrival pattern of a flow's packets to one of the SDN Controllers.
2. The SDN Controller forwards the flow's inter-arrival pattern to an appropriate security service application, such as DDoS-attack Mitigator.
3. The DDoS-attack Mitigator analyzes the reported pattern for the flow.
4. If the DDoS-attack Mitigator regards the pattern as a DDoS attack, it computes a packet dropping probability corresponding to suspiciousness level and reports this DDoS-attack flow to the SDN Controller.
5. The SDN Controller installs new rules into switches (e.g., forward packets with the suspicious inter-arrival pattern with a dropping probability).

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

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

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

- o Cost: The cost of adding DDoS-attack mitigators to network resources such as routers, gateways, and switches is substantial due to the reason that we need to add DDoS-attack mitigator on each network resource. To solve this, each network resource can be managed centrally such that a single DDoS-attack mitigator is manipulated by a centralized server.
- o Performance: The performance of DDoS-attack mitigators is often slower than the link speed of network interfaces. The checking of DDoS attacks may reduce the performance of the network interfaces. DDoS-attack mitigators can be adaptively deployed among network switches, depending on network conditions in the framework.
- o The management of network resources: Since there may be hundreds of network resources in an administered network, the dynamic management of network resources for performance (e.g., load balancing) is a challenge for DDoS-attack mitigation. In the framework, for dynamic network resource management, a flow forwarding path management scheme can handle the load balancing of network switches [AVANT-GUARD]. With this management scheme, the current and near-future workload can be spread among the network switches for DDoS-attack mitigation. In addition, DDoS-attack mitigation rules can be dynamically added for new DDoS attacks.
- o The establishment of policy: Policy should be established for each network resource. However, it is difficult to describe what flows are permitted or denied for new DDoS-attacks (e.g., DNS reflection attack) within a specific organization network under management. Thus, a centralized view is helpful to determine security policies for such a network.

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

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

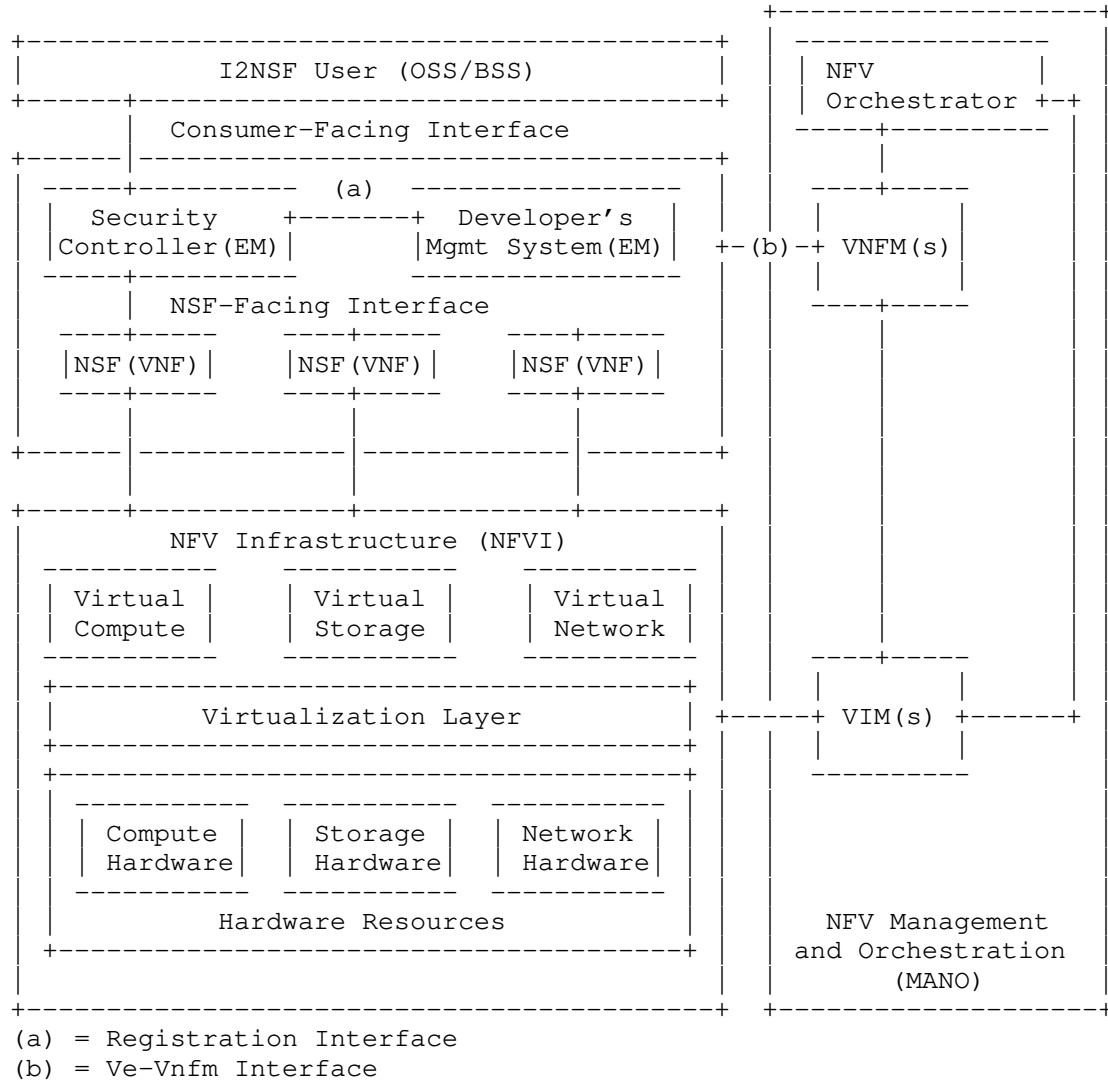


Figure 5: I2NSF Framework Implementation with respect to the NFV Reference Architectural Framework

7. I2NSF Framework with NFV

This section discusses the implementation of the I2NSF framework using Network Functions Virtualization (NFV).

NFV is a promising technology for improving the elasticity and efficiency of network resource utilization. In NFV environments, NSFs can be deployed in the forms of software-based virtual instances rather than physical appliances. Virtualizing NSFs makes it possible to rapidly and flexibly respond to the amount of service requests by dynamically increasing or decreasing the number of NSF instances. Moreover, NFV technology facilitates flexibly including or excluding NSFs from multiple security solution vendors according to the changes on security requirements. In order to take advantages of the NFV technology, the I2NSF framework can be implemented on top of an NFV infrastructure as show in Figure 5.

Figure 5 shows an I2NSF framework implementation based on the NFV reference architecture that the European Telecommunications Standards Institute (ETSI) defines [ETSI-NFV]. The NSFs are deployed as virtual network functions (VNFs) in Figure 5. The Developer's Management System (DMS) in the I2NSF framework is responsible for registering capability information of NSFs into the Security Controller. Those NSFs are created or removed by a virtual network functions manager (VNFM) in the NFV architecture that performs the life-cycle management of VNFs. The Security Controller controls and monitors the configurations (e.g., function parameters and security policy rules) of VNFs. Both the DMS and Security Controller can be implemented as the Element Managements (EMs) in the NFV architecture. Finally, the I2NSF User can be implemented as OSS/BSS (Operational Support Systems/Business Support Systems) in the NFV architecture that provides interfaces for users in the NFV system.

The operation procedure in the I2NSF framework based on the NFV architecture is as follows:

1. The VNFM has a set of virtual machine (VM) images of NSFs, and each VM image can be used to create an NSF instance that provides a set of security capabilities. The DMS initially registers a mapping table of the ID of each VM image and the set of capabilities that can be provided by an NSF instance created from the VM image into the Security Controller.
2. If the Security Controller does not have any instantiated NSF that has the set of capabilities required to meet the security requirements from users, it searches the mapping table (registered by the DMS) for the VM image ID corresponding to the required set of capabilities.

3. The Security Controller requests the DMS to instantiate an NSF with the VM image ID via VNFM.
4. When receiving the instantiation request, the VNFM first asks the NFV orchestrator for the permission required to create the NSF instance, requests the VIM to allocate resources for the NSF instance, and finally creates the NSF instance based on the allocated resources.
5. Once the NSF instance has been created by the VNFM, the DMS performs the initial configurations of the NSF instance and then notifies the Security Controller of the NSF instance.
6. After being notified of the created NSF instance, the Security Controller delivers low-level security policy rules to the NSF instance for policy enforcement.

We can conclude that the I2NSF framework can be implemented based on the NFV architecture framework. Note that the registration of the capabilities of NSFs is performed through the Registration Interface and the lifecycle management for NSFs (VNFs) is performed through the Ve-Vnfm interface between the DMS and VNFM, as shown in Figure 5. More details about the I2NSF framework based on the NFV reference architecture are described in [i2nsf-nfv-architecture].

8. Security Considerations

The same security considerations for the I2NSF framework [RFC8329] are applicable to this document.

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

9. Acknowledgments

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

This work has been partially supported by the European Commission under Horizon 2020 grant agreement no. 700199 "Securing against intruders and other threats through a NFV-enabled environment (SHIELD)". This support does not imply endorsement.

10. Contributors

I2NSF is a group effort. I2NSF has had a number of contributing authors. The following are considered co-authors:

- o Hyoungshick Kim (Sungkyunkwan University)
- o Jinyong Tim Kim (Sungkyunkwan University)
- o Hyunsik Yang (Soongsil University)
- o Younghan Kim (Soongsil University)
- o Jung-Soo Park (ETRI)
- o Se-Hui Lee (Korea Telecom)
- o Mohamed Boucadair (Orange)

11. References

11.1. Normative References

[ETSI-NFV]

"Network Functions Virtualisation (NFV); Architectural Framework", Available:
https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf, October 2013.

[ITU-T.Y.3300]

"Framework of Software-Defined Networking", Available: <https://www.itu.int/rec/T-REC-Y.3300-201406-I>, June 2014.

[NFV-Terminology]

"Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV", Available:
https://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf, December 2014.

[ONF-OpenFlow]

"OpenFlow Switch Specification (Version 1.4.0)", Available: <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf>, October 2013.

[ONF-SDN-Architecture]

"SDN Architecture (Issue 1.1)", Available:
https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf, June 2016.

- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.
- [RFC7665] Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.
- [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, July 2017.
- [RFC8300] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", RFC 8300, January 2018.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

11.2. Informative References

[AVANT-GUARD]

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

[consumer-facing-inf-dm]

Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", draft-ietf-i2nsf-consumer-facing-interface-dm-03 (work in progress), March 2019.

[ETSI-NFV-MANO]

"Network Functions Virtualisation (NFV); Management and Orchestration", Available:
https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf,
December 2014.

[i2nsf-nfv-architecture]

Yang, H., Kim, Y., Jeong, J., and J. Kim, "I2NSF on the NFV Reference Architecture", draft-yang-i2nsf-nfv-architecture-04 (work in progress), November 2018.

[i2nsf-nsf-cap-im]

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

[i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-07 (work in progress), January 2019.

[ITU-T.X.1252]

"Baseline Identity Management Terms and Definitions",
April 2010.

[ITU-T.X.800]

"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[nsf-facing-inf-dm]

Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-03 (work in progress), March 2019.

[nsf-monitoring-dm]

Jeong, J., Chung, C., Hares, S., Xia, L., and H. Birkholz, "A YANG Data Model for Monitoring I2NSF Network Security Functions", draft-ietf-i2nsf-nsf-monitoring-data-model-00 (work in progress), March 2019.

[nsf-triggered-steering]

Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-06 (work in progress), July 2018.

[opsawg-firewalls]

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

[policy-translation]

Yang, J., Jeong, J., and J. Kim, "Security Policy Translation in Interface to Network Security Functions", draft-yang-i2nsf-security-policy-translation-03 (work in progress), March 2019.

[registration-inf-dm]

Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-ietf-i2nsf-registration-interface-dm-02 (work in progress), March 2019.

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

[VNF-ONBOARDING]

"VNF Onboarding", Available:
<https://wiki.opnfv.org/display/mano/VNF+Onboarding>,
November 2016.

Appendix A. Changes from draft-ietf-i2nsf-applicability-08

The following changes have been made from draft-ietf-i2nsf-applicability-08:

- o This version has reflected the additional comments from Eric Rescorla who is a Security Area Director as follows.
- o In Section 3, for a Developer's Management System, the problem of an inside attacker is addressed, and a possible solution for the inside attacks is suggested through I2NSF NSF monitoring functionality. Also, some restrictions on the role of the DMS are required to deal with the inside attacks.
- o In Section 4, an XML code for the time-dependent web access control is explained as an example.
- o In Section 6, the definitions of an SDN forwarding element and an NSF are clarified such that an SDN forwarding element is a switch running as either a hardware middle box or a software virtual switch, and an NSF is a virtual network function for a security service. It also discusses about how to determine whether a given software element in virtualized environments is an NSF or a virtualized switch.

Authors' Addresses

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

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

Sangwon Hyun
Department of Computer Engineering
Chosun University
309 Pilmun-daero, Dong-Gu
Gwangju 61452
Republic of Korea

Phone: +82 62 230 7473
EMail: shyun@chosun.ac.kr

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

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

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

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

S. Hares
Huawei
J. Jeong
J. Kim
Sungkyunkwan University
R. Moskowitz
HTT Consulting
Q. Lin
Huawei
March 11, 2019

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

Abstract

This document defines a YANG data model for capabilities of various Network Security Functions (NSFs) in Interface to Network Security Functions (I2NSF) framework to centrally manage capabilities of various NSFs.

Status of This Memo

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

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

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
3.1. Tree Diagrams	4
4. Overview	4
5. YANG Tree Diagram	6
5.1. Capabilities of Network Security Function	6
6. YANG Data Modules	8
6.1. I2NSF Capability YANG Data Module	9
7. IANA Considerations	36
8. Security Considerations	37
9. References	37
9.1. Normative References	37
9.2. Informative References	39
Appendix A. Changes from draft-ietf-i2nsf-capability-data- model-02	40
Appendix B. Acknowledgments	40
Appendix C. Contributors	40
Authors' Addresses	41

1. Introduction

As the industry becomes more sophisticated and network devices (e.g., Internet of Things, Self-driving vehicles, and VoIP/VoLTE smartphones), service providers have a lot of problems mentioned in [RFC8192]. To resolve these problems, [i2nsf-nsf-cap-im] specifies the information model of the capabilities of Network Security Functions (NSFs).

This document provides a data model using YANG [RFC6020][RFC7950] that defines the capabilities of NSFs to centrally manage capabilities of those security devices. The security devices can register their own capabilities into Network Operator Management (Mgmt) System (i.e., Security Controller) with this YANG data model through the registration interface [RFC8329]. With the capabilities of those security devices registered centrally, those security devices can be easily managed [RFC8329]. This YANG data model is based on the information model for I2NSF NSF capabilities [i2nsf-nsf-cap-im].

This YANG data model uses an "Event-Condition-Action" (ECA) policy model that is used as the basis for the design of I2NSF Policy described in [RFC8329] and [i2nsf-nsf-cap-im]. Rules. The "ietf-i2nsf-capability" YANG module defined in this document provides the following features:

- o Definition for general capabilities of network security functions.
- o Definition for event capabilities of generic network security function.
- o Definition for condition capabilities of generic network security function.
- o Definition for condition capabilities of advanced network security function.
- o Definition for action capabilities of generic network security function.
- o Definition for resolution strategy capabilities of generic network security function.
- o Definition for default action capabilities of generic network security function.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119][RFC8174].

3. Terminology

This document uses the terminology described in [i2nsf-terminology][i2nsf-nsf-cap-im][RFC8431][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

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

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [RFC8340] is as follows:

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

4. Overview

This section explains overview how the YANG data model can be used in I2NSF framework described in [RFC8329]. Figure 1 shows capabilities of NSFs in I2NSF Framework. As shown in this figure, Developer's Mgmt System can register NSFs with capabilities that the network security device can support. To register NSFs in this way, the Developer's Mgmt System utilizes this standardized capabilities YANG data model through registration interface. With the capabilities of those network security devices registered centrally, those security devices can be easily managed, which can resolve the a lot of problems described in [RFC8192]. The following shows use cases.

Note [i2nsf-nsf-yang] is used to configure security policy rules of generic network security functions and [i2nsf-advanced-nsf-dm] is used to configure security policy rules of advanced network security functions according to the capabilities of network security devices registered in I2NSF Framework.

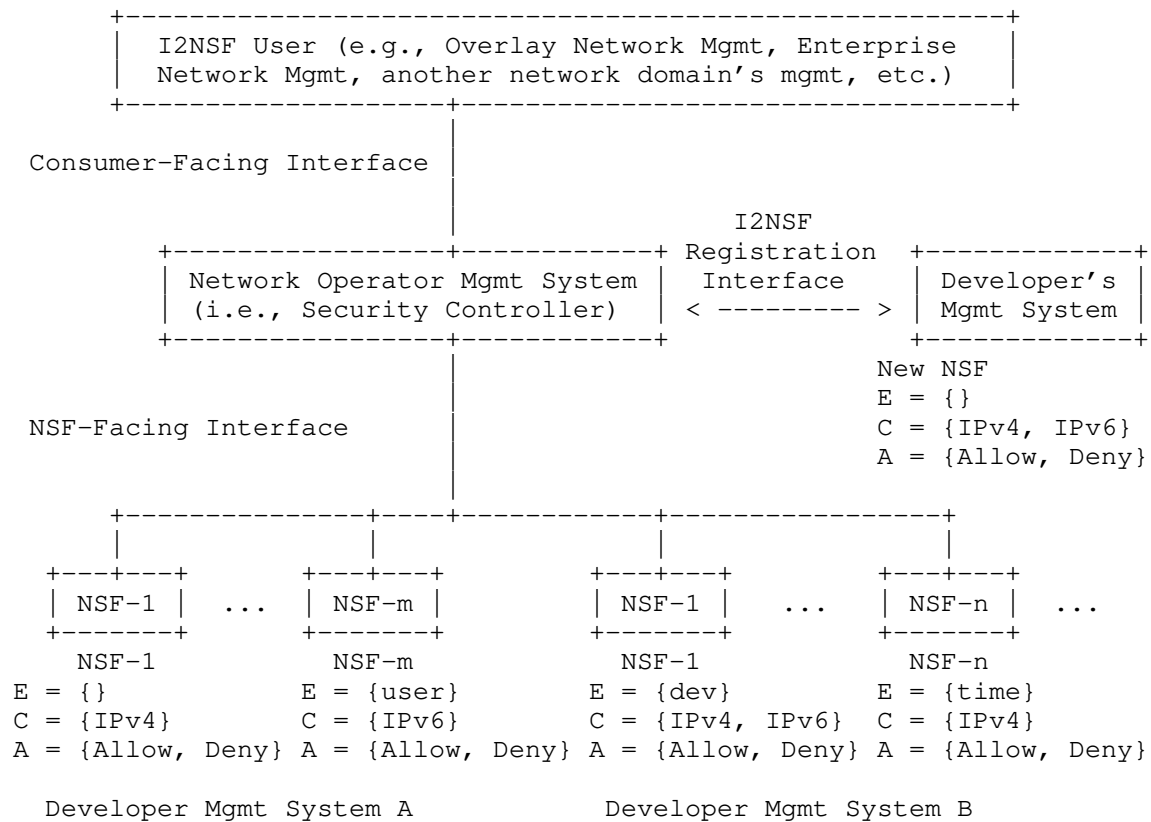


Figure 1: Capabilities of NSFs in I2NSF Framework

- o If network manager wants to apply security policy rules about blocking malicious users, it is a tremendous burden to apply all of these rules to NSFs one by one. This problem can be resolved by managing the capabilities of NSFs. If network manager wants to block malicious users with IPv6, network manager sends the security policy rules about blocking the users to Network Operator Mgmt System using I2NSF user (i.e., a web browser or a software). When the Network Operator Mgmt System receives the security policy rules, it automatically sends that security policy rules to appropriate NSFs (i.e., NSF-m in Developer Mgmt System A and NSF-1 in Developer Mgmt System B) which can support the capabilities (i.e., IPv6). Therefore, I2NSF User need not consider NSFs where to apply the rules.
- o If NSFs find the malicious packets, it is a tremendous burden for network manager to apply the rule about blocking the malicious packets to NSFs one by one. This problem can be resolved by

managing the capabilities of NSFs. If NSFs find the suspicious packets with IPv4, they can ask the Network Operator Mgmt System for information about the suspicious packets with IPv4. to alter specific rules and/or configurations. When the Network Operator Mgmt System receives information, it inspects the information about the suspicious packets with IPv4. If the suspicious packets are determined to be malicious packets, the Network Operator Mgmt System creates and sends the security policy rule against malicious packets to appropriate NSFs (i.e., NSF-1 in Developer Mgmt System A and NSF-1 and NSF-n in Developer Mgmt System B) which can support the capabilities (i.e., IPv4). Therefore, the new security policy rule against malicious packets can be applied to appropriate NSFs without intervention of humans.

5. YANG Tree Diagram

This section shows an YANG tree diagram of capabilities for network security functions, as defined in the [i2nsf-nsf-cap-im].

5.1. Capabilities of Network Security Function

This section shows YANG tree diagram for capabilities of network security functions.


```

module: ietf-i2nsf-capability
  +--rw nsf
    +--rw time-capabilities*          enumeration
    +--rw event-capabilities
      |
      | +--rw system-event-cap*    identityref
      | +--rw system-alarm-cap*   identityref
    +--rw condition-capabilities
      |
      | +--rw generic-nsf-capabilities
      |   |
      |   | +--rw ipv4-cap*    identityref
      |   | +--rw ipv6-cap*    identityref
      |   | +--rw tcp-cap*     identityref
      |   | +--rw udp-cap*     identityref
      |   | +--rw icmp-cap*    identityref
      |   +--rw advanced-nsf-capabilities
      |     |
      |     | +--rw antivirus-cap*    identityref
      |     | +--rw antiddos-cap*    identityref
      |     | +--rw ips-cap*         identityref
      |     | +--rw http-cap*        identityref
      |     | +--rw voip-volte-cap*  identityref
    +--rw action-capabilities
      |
      | +--rw ingress-action-cap*    identityref
      | +--rw egress-action-cap*     identityref
      | +--rw log-action-cap*        identityref
    +--rw resolution-strategy-capabilities* identityref
    +--rw default-action-capabilities*    identityref

```

Figure 2: YANG Tree Diagram for Capabilities of Network Security Functions

This YANG tree diagram shows capabilities of network security functions.

The NSF includes NSF capabilities. The NSF capabilities include time capabilities, event capabilities, condition capabilities, action capabilities, resolution strategy capabilities, and default action capabilities.

Time capabilities are used to specify capabilities when to execute the I2NSF policy rule. The time capabilities are defined as absolute time and periodic time.

Event capabilities are used to specify capabilities how to trigger the evaluation of the condition clause of the I2NSF Policy Rule. The event capabilities are defined as system event and system alarm. The event capability can be extended according to specific vendor condition features. The event capability is described in detail in [i2nsf-nsf-cap-im].

Condition capabilities are used to specify capabilities of a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to determine whether or not the set of actions in that (imperative) I2NSF policy rule can be executed or not. The condition capability is classified as condition capabilities of generic network security functions and advanced network security functions. The condition capabilities of generic network security functions are defined as IPv4 capability, IPv6 capability, tcp capability, udp capability, and icmp capability. The condition capabilities of advanced network security functions are defined as antivirus capability, antiddos capability, ips capability, http capability, and VoIP/VoLTE capability. The condition capability can be extended according to specific vendor condition features. The condition capability is described in detail in [i2nsf-nsf-cap-im].

Action capabilities is used to specify capabilities how to control and monitor aspects of flow-based NSFs when the event and condition clauses are satisfied. The action capabilities are defined as ingress action capability, egress action capability, and log action capability. The action capability can be extended according to specific vendor action features. The action capability is described in detail in [i2nsf-nsf-cap-im].

Resolution strategy capabilities are used to specify capabilities how to resolve conflicts that occur between the actions of the same or different policy rules that are matched and contained in this particular NSF. The resolution strategy capabilities are defined as First Matching Rule (FMR), Last Matching Rule (LMR), Prioritized Matching Rule (PMR) with Errors (PMRE), and Prioritized Matching Rule with No Errors (PMRN). The resolution strategy capability can be extended according to specific vendor action features. The resolution strategy capability is described in detail in [i2nsf-nsf-cap-im].

Default action capabilities are used to specify capabilities how to execute I2NSF policy rule when no rule matches a packet. The default action capabilities are defined as pass, drop, reject, alert, and mirror. The default action capability can be extended according to specific vendor action features. The default action capability is described in detail in [i2nsf-nsf-cap-im].

6. YANG Data Modules

6.1. I2NSF Capability YANG Data Module

This section introduces an YANG data module for capabilities of network security functions, as defined in the [i2nsf-nsf-cap-im].

<CODE BEGINS> file "ietf-i2nsf-capability@2019-03-11.yang"

```
module ietf-i2nsf-capability {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability";
  prefix
    iicapa;

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

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

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

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

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

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

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

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

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 8341; see
the RFC itself for full legal notices.";

```
revision "2019-03-11"{
  description "Initial revision.";
  reference
    "RFC XXXX: I2NSF Capability YANG Data Model";
}

/*
 * Identities
 */

identity event {
  description
    "Base identity for event of policy.";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
    - Event";
}

identity system-event-capability {
  base event;
  description
    "Identity for system event";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
    - System alarm";
}

identity system-alarm-capability {
  base event;
  description
    "Identity for system alarm";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
    - System alarm";
}

identity access-violation {
  base system-event-capability;
  description
    "Identity for access violation
    among system events";
```

```
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
        - System event";
  }

  identity configuration-change {
    base system-event-capability;
    description
      "Identity for configuration change
        among system events";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
        - System event";
  }

  identity memory-alarm {
    base system-alarm-capability;
    description
      "Identity for memory alarm
        among system alarms";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
        - System alarm";
  }

  identity cpu-alarm {
    base system-alarm-capability;
    description
      "Identity for cpu alarm
        among system alarms";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
        - System alarm";
  }

  identity disk-alarm {
    base system-alarm-capability;
    description
      "Identity for disk alarm
        among system alarms";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
        - System alarm";
  }

  identity hardware-alarm {
    base system-alarm-capability;
    description
```

```
    "Identity for hardware alarm
    among system alarms";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
    - System alarm";
}

identity interface-alarm {
  base system-alarm-capability;
  description
    "Identity for interface alarm
    among system alarms";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
    - System alarm";
}

identity condition {
  description
    "Base identity for conditions of policy";
}

identity ipv4-capability {
  base condition;
  description
    "Identity for capabilities of IPv4 condition";
  reference
    "RFC 791: Internet Protocol";
}

identity exact-ipv4-header-length {
  base ipv4-capability;
  description
    "Identity for exact header length capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Header Length";
}

identity range-ipv4-header-length {
  base ipv4-capability;
  description
    "Identity for range header length capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Header Length";
}
```

```
identity ipv4-tos {
  base ipv4-capability;
  description
    "Identity for type of service capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Type of Service";
}

identity exact-ipv4-total-length {
  base ipv4-capability;
  description
    "Identity for exact total length capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Total Length";
}

identity range-ipv4-total-length {
  base ipv4-capability;
  description
    "Identity for range total length capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Total Length";
}

identity ipv4-id {
  base ipv4-capability;
  description
    "Identity for identification capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Identification";
}

identity ipv4-fragment-flags {
  base ipv4-capability;
  description
    "Identity for fragment flags capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Fragmentation  Flags";
}

identity exact-ipv4-fragment-offset {
  base ipv4-capability;
  description
```

```
    "Identity for exact fragment offset capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Fragmentation  Offset";
}

identity range-ipv4-fragment-offset {
  base ipv4-capability;
  description
    "Identity for range fragment offset capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Fragmentation  Offset";
}

identity exact-ipv4-ttl {
  base ipv4-capability;
  description
    "Identity for exact time to live capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Time To Live (TTL)";
}

identity range-ipv4-ttl {
  base ipv4-capability;
  description
    "Identity for range time to live capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Time To Live (TTL)";
}

identity ipv4-protocol {
  base ipv4-capability;
  description
    "Identity for protocol capability
    of IPv4 condition";
  reference
    "RFC 790: Assigned numbers - Assigned Internet
    Protocol Number
    RFC 791: Internet Protocol - Protocol";
}

identity exact-ipv4-address {
  base ipv4-capability;
  description
    "Identity for exact address capability
```



```
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Address";
}

identity range-ipv4-address {
  base ipv4-capability;
  description
    "Identity for range-address capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Address";
}

identity ipv4-ipopts {
  base ipv4-capability;
  description
    "Identity for option capability
    of IPv4 condition";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity ipv4-sameip {
  base ipv4-capability;
  description
    "Identity for sameIP capability
    of IPv4 condition";
}

identity ipv4-geoip {
  base ipv4-capability;
  description
    "Identity for geography capability
    of IPv4 condition";
}

identity ipv6-capability {
  base condition;
  description
    "Identity for capabilities of IPv6 condition";
  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification";
}

identity ipv6-traffic-class {
  base ipv6-capability;
```

```
    description
      "Identity for traffic class capability
      of IPv6 condition";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Traffic Class";
  }

  identity exact-ipv6-flow-label {
    base ipv6-capability;
    description
      "Identity for exact flow label capability
      of IPv6 condition";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Flow Label";
  }

  identity range-ipv6-flow-label {
    base ipv6-capability;
    description
      "Identity for range flow label capability
      of IPv6 condition";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Flow Label";
  }

  identity exact-ipv6-payload-length {
    base ipv6-capability;
    description
      "Identity for exact payload length capability
      of IPv6 condition";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Payload Length";
  }

  identity range-ipv6-payload-length {
    base ipv6-capability;
    description
      "Identity for range payload length capability
      of IPv6 condition";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Payload Length";
  }
}
```

```
identity ipv6-next-header {
  base ipv6-capability;
  description
    "Identity for next header capability
    of IPv6 condition";
  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Next Header";
}

identity exact-ipv6-hop-limit {
  base ipv6-capability;
  description
    "Identity for exact hop limit capability
    of IPv6 condition";
  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Hop Limit";
}

identity range-ipv6-hop-limit {
  base ipv6-capability;
  description
    "Identity for range hop limit capability
    of IPv6 condition";
  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Hop Limit";
}

identity exact-ipv6-address {
  base ipv6-capability;
  description
    "Identity for exact address capability
    of IPv6 condition";
  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Address";
}

identity range-ipv6-address {
  base ipv6-capability;
  description
    "Identity for range address capability
    of IPv6 condition";
  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Address";
}
```

```
}

identity tcp-capa {
  base condition;
  description
    "Identity for capabilities of tcp condition";
  reference
    "RFC 793: Transmission Control Protocol";
}

identity exact-tcp-port-num {
  base tcp-capa;
  description
    "Identity for exact port number capability
    of tcp condition";
  reference
    "RFC 793: Transmission Control Protocol - Port Number";
}

identity range-tcp-port-num {
  base tcp-capa;
  description
    "Identity for range port number capability
    of tcp condition";
  reference
    "RFC 793: Transmission Control Protocol - Port Number";
}

identity exact-tcp-seq-num {
  base tcp-capa;
  description
    "Identity for exact sequence number capability
    of tcp condition";
  reference
    "RFC 793: Transmission Control Protocol - Sequence Number";
}

identity range-tcp-seq-num {
  base tcp-capa;
  description
    "Identity for range sequence number capability
    of tcp condition";
  reference
    "RFC 793: Transmission Control Protocol - Sequence Number";
}

identity exact-tcp-ack-num {
  base tcp-capa;
```

```
    description
      "Identity for exact acknowledgement number capability
      of tcp condition";
    reference
      "RFC 793: Transmission Control Protocol - Acknowledgement Number";
  }

  identity range-tcp-ack-num {
    base tcp-cap;
    description
      "Identity for range acknowledgement number capability
      of tcp condition";
    reference
      "RFC 793: Transmission Control Protocol - Acknowledgement Number";
  }

  identity exact-tcp-window-size {
    base tcp-cap;
    description
      "Identity for exact window size capability
      of tcp condition";
    reference
      "RFC 793: Transmission Control Protocol - Window Size";
  }

  identity range-tcp-window-size {
    base tcp-cap;
    description
      "Identity for range window size capability
      of tcp condition";
    reference
      "RFC 793: Transmission Control Protocol - Window Size";
  }

  identity tcp-flags {
    base tcp-cap;
    description
      "Identity for flags capability
      of tcp condition";
    reference
      "RFC 793: Transmission Control Protocol - Flags";
  }

  identity udp-cap {
    base condition;
    description
      "Identity for capabilities of udp condition";
    reference
```

```
    "RFC 768: User Datagram Protocol";
}

identity exact-udp-port-num {
  base udp-cap;
  description
    "Identity for exact port number capability
    of udp condition";
  reference
    "RFC 768: User Datagram Protocol - Port Number";
}

identity range-udp-port-num {
  base udp-cap;
  description
    "Identity for range port number capability
    of udp condition";
  reference
    "RFC 768: User Datagram Protocol - Port Number";
}

identity exact-udp-total-length {
  base udp-cap;
  description
    "Identity for exact total-length capability
    of udp condition";
  reference
    "RFC 768: User Datagram Protocol - Total Length";
}

identity range-udp-total-length {
  base udp-cap;
  description
    "Identity for range total-length capability
    of udp condition";
  reference
    "RFC 768: User Datagram Protocol - Total Length";
}

identity icmp-cap {
  base condition;
  description
    "Identity for capabilities of icmp condition";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity icmp-type {
```

```
    base icmp-capability;
    description
        "Identity for icmp type capability
        of icmp condition";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity http-capability {
    base condition;
    description
        "Identity for capabilities of http condition";
}

identity uri {
    base http-capability;
    description
        "Identity for uri capabilities of
        http condition";
}

identity url {
    base http-capability;
    description
        "Identity for url capabilities of
        http condition";
}

identity log-action-capability {
    description
        "Identity for capabilities of log action";
}

identity rule-log {
    base log-action-capability;
    description
        "Identity for rule log capability
        of log action";
}

identity session-log {
    base log-action-capability;
    description
        "Identity for session log capability
        of log action";
}

identity ingress-action-capability {
```

```
    description
      "Identity for capabilities of ingress action";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Action";
  }

  identity egress-action-capability {
    description
      "Base identity for egress action";
  }

  identity default-action-capability {
    description
      "Identity for capabilities of default action";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Default action";
  }

  identity pass {
    base ingress-action-capability;
    base egress-action-capability;
    base default-action-capability;
    description
      "Identity for pass";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Actions and
        default action";
  }

  identity drop {
    base ingress-action-capability;
    base egress-action-capability;
    base default-action-capability;
    description
      "Identity for drop";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Actions and
        default action";
  }

  identity reject {
    base ingress-action-capability;
    base egress-action-capability;
    base default-action-capability;
```



```
    description
      "Identity for reject";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Actions and
      default action";
  }

  identity alert {
    base ingress-action-capability;
    base egress-action-capability;
    base default-action-capability;
    description
      "Identity for alert";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Actions and
      default action";
  }

  identity mirror {
    base ingress-action-capability;
    base egress-action-capability;
    base default-action-capability;
    description
      "Identity for mirror";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Actions and
      default action";
  }

  identity invoke-signaling {
    base egress-action-capability;
    description
      "Identity for invoke signaling";
  }

  identity tunnel-encapsulation {
    base egress-action-capability;
    description
      "Identity for tunnel encapsulation";
  }

  identity forwarding {
    base egress-action-capability;
    description
      "Identity for forwarding";
```

```
}

identity redirection {
  base egress-action-capability;
  description
    "Identity for redirection";
}

identity resolution-strategy-capability {
  description
    "Base identity for resolution strategy";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSF's Capabilities - Resolution Strategy";
}

identity fmr {
  base resolution-strategy-capability;
  description
    "Identity for First Matching Rule (FMR)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSF's Capabilities - Resolution Strategy";
}

identity lmr {
  base resolution-strategy-capability;
  description
    "Identity for Last Matching Rule (LMR)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSF's Capabilities - Resolution Strategy";
}

identity pmr {
  base resolution-strategy-capability;
  description
    "Identity for Prioritized Matching Rule (PMR)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSF's Capabilities - Resolution Strategy";
}

identity pmre {
  base resolution-strategy-capability;
  description
    "Identity for Prioritized Matching Rule
    with Errors (PMRE)";
```

```
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Resolution Strategy";
  }

  identity pmrn {
    base resolution-strategy-capability;
    description
      "Identity for Prioritized Matching Rule
        with No Errors (PMRN)";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Resolution Strategy";
  }

  identity advanced-nsf-capability {
    description
      "Base identity for advanced
        network security function capabilities";
    reference
      "RFC 8329: Framework for Interface to Network Security
        Functions - Differences from ACL Data Models
        draft-dong-i2nsf-asf-config-01: Configuration of
        Advanced Security Functions with I2NSF Security
        Controller";
  }

  identity antivirus-capability {
    base advanced-nsf-capability;
    description
      "Identity for antivirus capabilities";
    reference
      "RFC 8329: Framework for Interface to Network Security
        Functions - Differences from ACL Data Models
        draft-dong-i2nsf-asf-config-01: Configuration of
        Advanced Security Functions with I2NSF Security
        Controller - Antivirus";
  }

  identity antiddos-capability {
    base advanced-nsf-capability;
    description
      "Identity for antiddos capabilities";
    reference
      "RFC 8329: Framework for Interface to Network Security
        Functions - Differences from ACL Data Models
        draft-dong-i2nsf-asf-config-01: Configuration of
        Advanced Security Functions with I2NSF Security
```

```
        Controller - Antiddos";
    }

    identity ips-capability {
        base advanced-nsf-capability;
        description
            "Identity for IPS capabilities";
        reference
            "RFC 8329: Framework for Interface to Network Security
            Functions - Differences from ACL Data Models
            draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller - Intrusion Prevention System";
    }

    identity voip-volte-capability {
        base advanced-nsf-capability;
        description
            "Identity for VoIP/VoLTE capabilities";
        reference
            "RFC 3261: SIP: Session Initiation Protocol
            RFC 8329: Framework for Interface to Network Security
            Functions - Differences from ACL Data Models
            draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller";
    }

    identity detect {
        base antivirus-capability;
        description
            "Identity for detect capabilities
            of antivirus";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller - Antivirus";
    }

    identity exception-application {
        base antivirus-capability;
        description
            "Identity for exception application capabilities
            of antivirus";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller - Antivirus";
    }
```

```
}

identity exception-signature {
  base antivirus-capa;
  description
    "Identity for exception signature capabilities
    of antivirus";
  reference
    "draft-dong-i2nsf-asf-config-01: Configuration of
    Advanced Security Functions with I2NSF Security
    Controller - Antivirus";
}

identity whitelists {
  base antivirus-capa;
  description
    "Identity for whitelists capabilities
    of antivirus";
  reference
    "draft-dong-i2nsf-asf-config-01: Configuration of
    Advanced Security Functions with I2NSF Security
    Controller - Antivirus";
}

identity syn-flood-action {
  base antiddos-capa;
  description
    "Identity for syn flood action capabilities
    of antiddos";
  reference
    "draft-dong-i2nsf-asf-config-01: Configuration of
    Advanced Security Functions with I2NSF Security
    Controller - Antiddos";
}

identity udp-flood-action {
  base antiddos-capa;
  description
    "Identity for udp flood action capabilities
    of antiddos";
  reference
    "draft-dong-i2nsf-asf-config-01: Configuration of
    Advanced Security Functions with I2NSF Security
    Controller - Antiddos";
}

identity http-flood-action {
  base antiddos-capa;
```

```
    description
      "Identity for http flood action capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity https-flood-action {
    base antiddos-cap;
    description
      "Identity for https flood action capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity dns-request-flood-action {
    base antiddos-cap;
    description
      "Identity for dns request flood action capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity dns-reply-flood-action {
    base antiddos-cap;
    description
      "Identity for dns reply flood action capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity icmp-flood-action {
    base antiddos-cap;
    description
      "Identity for icmp flood action capabilities
      of antiddos";
    reference
```

```
    "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity sip-flood-action {
    base antiddos-capability;
    description
      "Identity for sip flood action capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity detect-mode {
    base antiddos-capability;
    description
      "Identity for detect mode capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity baseline-learn {
    base antiddos-capability;
    description
      "Identity for baseline learn capabilities
      of antiddos";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Antiddos";
  }

  identity signature-set {
    base ips-capability;
    description
      "Identity for signature set capabilities
      of IPS";
    reference
      "draft-dong-i2nsf-asf-config-01: Configuration of
      Advanced Security Functions with I2NSF Security
      Controller - Intrusion Prevention System";
  }
```

```
identity ips-exception-signature {
  base ips-cap;
  description
    "Identity for ips exception signature capabilities
    of IPS";
  reference
    "draft-dong-i2nsf-asf-config-01: Configuration of
    Advanced Security Functions with I2NSF Security
    Controller - Intrusion Prevention System";
}

identity voice-id {
  base voip-volte-cap;
  description
    "Identity for voice-id capabilities
    of VoIP/VoLTE";
  reference
    "RFC 3261: SIP: Session Initiation Protocol";
}

identity user-agent {
  base voip-volte-cap;
  description
    "Identity for user agent capabilities
    of VoIP/VoLTE";
  reference
    "RFC 3261: SIP: Session Initiation Protocol";
}

/*
 * Grouping
 */

grouping nsf-capabilities {
  description
    "Capabilities of network security function";
  reference
    "RFC 8329: Framework for Interface to Network Security
    Functions - I2NSF Flow Security Policy Structure
    draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Capability Information Model Design";

  leaf-list time-capabilities {
    type enumeration {
      enum absolute-time {
        description
          "Capabilities of absolute time.
          If network security function has the absolute time
```



```
        capability, the network security function
        supports rule execution according to absolute time.";
    }
    enum periodic-time {
        description
            "Capabilities of periodic time.
            If network security function has the periodic time
            capability, the network security function
            supports rule execution according to periodic time.";
    }
}
description
    "This is capabilities for time";
}

container event-capabilities {
    description
        "Capabilities of events.
        If network security function has
        the event capabilities, the network security functions
        supports rule execution according to system event
        and system alarm.";

    reference
        "RFC 8329: Framework for Interface to Network Security
        Functions - I2NSF Flow Security Policy Structure
        draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities - Design Principles and ECA
        Policy Model Overview
        draft-hong-i2nsf-nsf-monitoring-data-model-06: A YANG
        Data Model for Monitoring I2NSF Network Security
        Functions - System Alarm and System Events";

    leaf-list system-event-capa {
        type identityref {
            base system-event-capa;
        }
        description
            "Capabilities for a system event";
    }

    leaf-list system-alarm-capa {
        type identityref {
            base system-alarm-capa;
        }
        description
            "Capabilities for a system alarm";
    }
}
```

```
}

container condition-capabilities {
  description
    "Capabilities of conditions.";

  container generic-nsf-capabilities {
    description
      "Capabilities of conditions.
      If a network security function has
      the condition capabilities, the network security function
      supports rule execution according to conditions of IPv4,
      IPv6, foruth layer, ICMP, and payload.";
    reference
      "RFC 791: Internet Protocol
      RFC 792: Internet Control Message Protocol
      RFC 793: Transmission Control Protocol
      RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Next Header
      RFC 8329: Framework for Interface to Network Security
      Functions - I2NSF Flow Security Policy Structure
      draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Design Principles and ECA Policy
      Model Overview";

    leaf-list ipv4-capa {
      type identityref {
        base ipv4-capa;
      }
      description
        "Capabilities for an IPv4 packet";
      reference
        "RFC 791: Internet Protocol";
    }

    leaf-list ipv6-capa {
      type identityref {
        base ipv6-capa;
      }
      description
        "Capabilities for an IPv6 packet";
      reference
        "RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next Header";
    }

    leaf-list tcp-capa {
      type identityref {
```

```
        base tcp-capa;
    }
    description
        "Capabilities for a tcp packet";
    reference
        "RFC 793: Transmission Control Protocol";
}

leaf-list udp-capa {
    type identityref {
        base udp-capa;
    }
    description
        "Capabilities for an udp packet";
    reference
        "RFC 768: User Datagram Protocol";
}

leaf-list icmp-capa {
    type identityref {
        base icmp-capa;
    }
    description
        "Capabilities for an ICMP packet";
    reference
        "RFC 2460: Internet Protocol, Version 6 (IPv6) ";
}
}

container advanced-nsf-capabilities {
    description
        "Capabilities of advanced network security functions,
        such as anti virus, anti DDoS, IPS, and VoIP/VoLTE.";
    reference
        "RFC 8329: Framework for Interface to Network Security
        Functions - Differences from ACL Data Models
        draft-dong-i2nsf-asf-config-01: Configuration of
        Advanced Security Functions with I2NSF Security
        Controller";

    leaf-list antivirus-capa {
        type identityref {
            base antivirus-capa;
        }
        description
            "Capabilities for an antivirus";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
```

```
        Advanced Security Functions with I2NSF Security
        Controller";
    }

    leaf-list antiddos-capability {
        type identityref {
            base antiddos-capability;
        }
        description
            "Capabilities for an antiddos";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller";
    }

    leaf-list ips-capability {
        type identityref {
            base ips-capability;
        }
        description
            "Capabilities for an ips";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller";
    }

    leaf-list http-capability {
        type identityref {
            base http-capability;
        }
        description
            "Capabilities for a http";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
            Advanced Security Functions with I2NSF Security
            Controller";
    }

    leaf-list voip-volte-capability {
        type identityref {
            base voip-volte-capability;
        }
        description
            "Capabilities for a voip and volte";
        reference
            "draft-dong-i2nsf-asf-config-01: Configuration of
```

```
        Advanced Security Functions with I2NSF Security
        Controller";
    }
}
container action-capabilities {
  description
    "Capabilities of actions.
    If network security function has
    the action capabilities, the network security function
    supports rule execution according to actions.";

  leaf-list ingress-action-capability {
    type identityref {
      base ingress-action-capability;
    }
    description
      "Capabilities for an action";
  }

  leaf-list egress-action-capability {
    type identityref {
      base egress-action-capability;
    }
    description
      "Capabilities for an egress action";
  }

  leaf-list log-action-capability {
    type identityref {
      base log-action-capability;
    }
    description
      "Capabilities for a log action";
  }
}

leaf-list resolution-strategy-capabilities {
  type identityref {
    base resolution-strategy-capability;
  }
  description
    "Capabilities for a resolution strategy.
    The resolution strategies can be used to
    specify how to resolve conflicts that occur between
    the actions of the same or different policy rules that
    are matched and contained in this particular NSF";
  reference
```

```
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Resolution strategy";
  }

  leaf-list default-action-capabilities {
    type identityref {
      base default-action-capability;
    }
    description
      "Capabilities for a default action.
       A default action is used to execute I2NSF policy rule
       when no rule matches a packet. The default action is
       defined as pass, drop, reject, alert, and mirror.";
    reference
      "draft-ietf-i2nsf-capability-04: Information Model
       of NSFs Capabilities - Default action";
  }
}

/*
 * Data nodes
 */

container nsf {
  description
    "The list of capabilities of
     network security function";
  uses nsf-capabilities;
}

<CODE ENDS>
```

Figure 3: YANG Data Module of I2NSF Capability

7. 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-capability

Registrant Contact: The IESG.

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-capability

namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability

prefix: iicapa

reference: RFC XXXX

8. Security Considerations

The YANG module specified in this document defines a data schema designed to be accessed through network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the required transport secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the required transport secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides a means of restricting access to specific NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [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>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [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., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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, July 2017.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.
- [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>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8431] Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", RFC RFC8431, September 2018.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

9.2. Informative References

[i2nsf-advanced-nsf-dm]

Pan, W. and L. Xia, "Configuration of Advanced Security Functions with I2NSF Security Controller", draft-dong-i2nsf-asf-config-01 (work in progress), October 2018.

[i2nsf-nsf-cap-im]

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

[i2nsf-nsf-yang]

Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-01 (work in progress), July 2018.

[i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-07 (work in progress), January 2019.

[supa-policy-info-model]

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

Appendix A. Changes from draft-ietf-i2nsf-capability-data-model-02

The following changes are made from draft-ietf-i2nsf-capability-data-model-03:

- o We revised this YANG data module according to guidelines for authors and reviewers of YANG data model documents [RFC6087].
- o We changed the structure of the overall YANG data module.
- o We changed enumeration type to identity type for scalable components.
- o We added a description for the YANG tree diagram of the YANG data module.
- o We revised overall sentences of this YANG data model document.
- o We added configuration examples to make it easier for reviewers to understand.

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. The following are considered co-authors:

- o Hyounghshick Kim (Sungkyunkwan University)
- o Daeyoung Hyun (Sungkyunkwan University)
- o Dongjin Hong (Sungkyunkwan University)
- o Liang Xia (Huawei)
- o Jung-Soo Park (ETRI)
- o Tae-Jin Ahn (Korea Telecom)
- o Se-Hui Lee (Korea Telecom)

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

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

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

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

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

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

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

EMail: linqiushi@huawei.com

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	5
3. Terminology	5
4. Information Model for Policy	5
4.1. Event Sub-Model	6
4.2. Condition Sub-Model	7
4.3. Action Sub-Model	9
5. Information Model for Multi-Tenancy	9
5.1. Policy-Domain	10
5.2. Policy-Tenant	11
5.3. Policy-Role	12
5.4. Policy-User	12
5.5. Policy Management Authentication Method	13
6. Information Model for Policy Endpoint Groups	14
6.1. User Group	15
6.2. Device-Group	16
6.3. Location-Group	16
7. Information Model for Threat Prevention	17
7.1. Threat-Feed	18
7.2. Payload-content	19
8. Role-Based Access Control (RBAC)	19
9. YANG Data Model for Security Policies for Consumer-Facing Interface	20
10. Example XML Output for Various Scenarios	37
10.1. DB registration: Information of positions and devices (Endpoint group)	38
10.2. Scenario 1: Block SNS access during business hours	38
10.3. Scenario 2: Block malicious VoIP/VoLTE packets coming to the company	40
10.4. Scenario 3: Mitigate HTTP and HTTPS flood attacks on a company web Server	41

11. Security Considerations	43
12. IANA Considerations	43
13. References	43
13.1. Normative References	43
13.2. Informative References	43
Appendix A. Changes from draft-ietf-i2nsf-consumer-facing- interface-dm-02	46
Appendix B. Acknowledgments	46
Appendix C. Contributors	47
Authors' Addresses	48

1. Introduction

In I2NSF framework, each vendor can register their NSF's 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 security 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 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 the later sections of this document (see section 10).

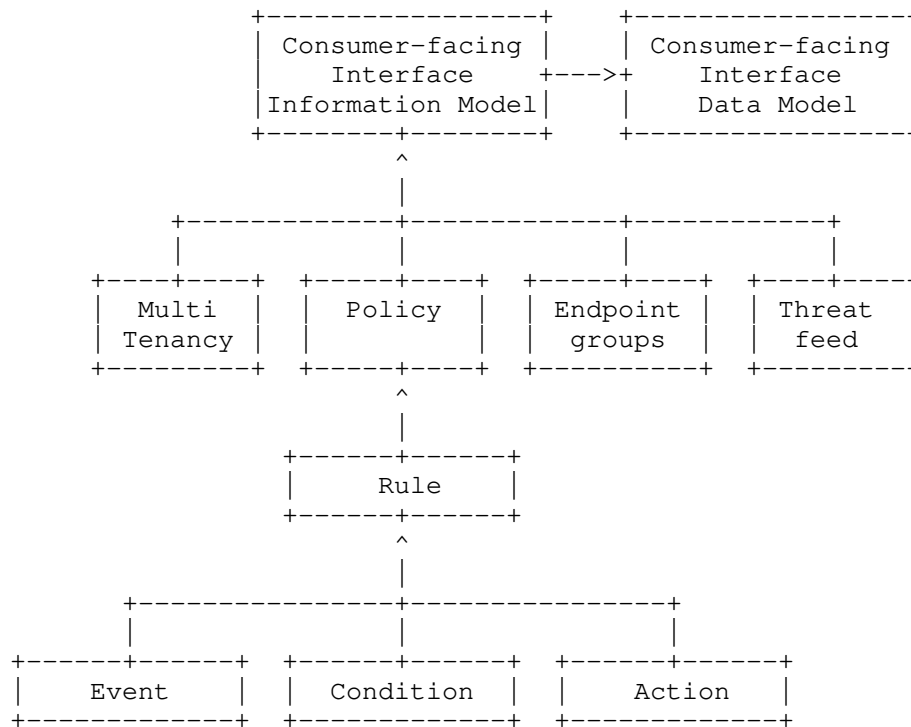


Figure 1: Diagram for High-level Abstraction of Consumer-Facing Interface

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 owner 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.

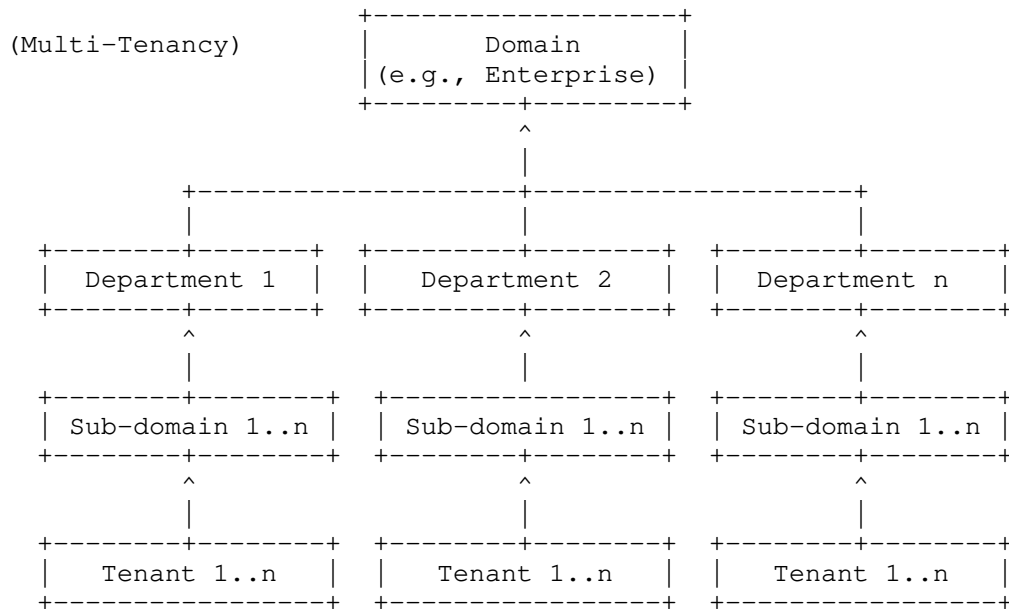


Figure 7: Multi-tenancy Diagram

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-mgmt-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 object. 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-mgmt-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.

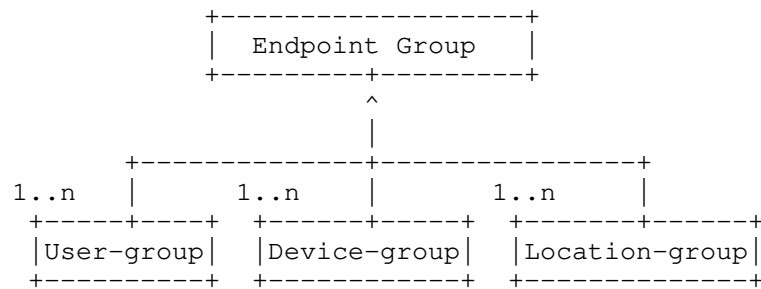


Figure 13: Endpoint Group Diagram

```

+--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.

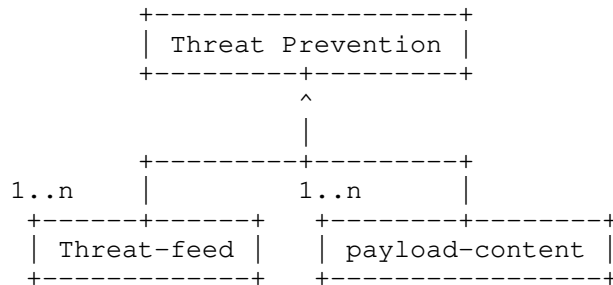


Figure 18: Threat Prevention Diagram

```

+--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 Access 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.

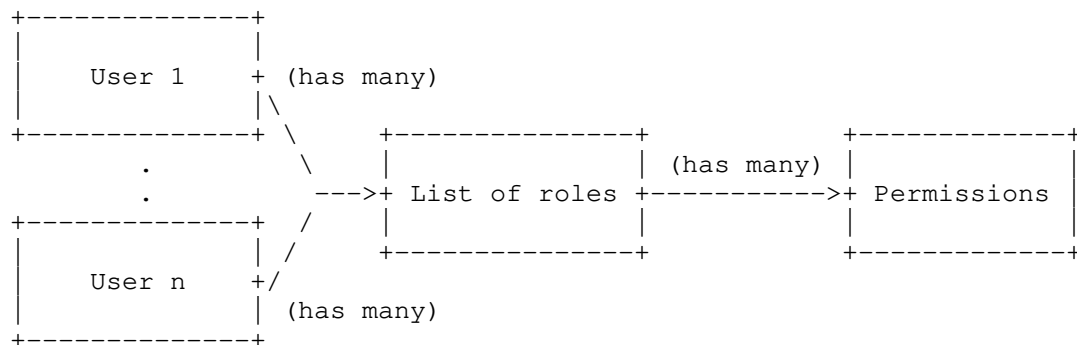


Figure 22: RBAC Diagram

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 hierarchy. 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.dunbar@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 ransomware {
        base malware-file-type;
        description
            "Identity for ransomware 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-mgmt-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-mgmt-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 management 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 contains 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 coming 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 NSF's are needed to meet the security requirements. Figure 26 represents the XML document generated from YANG discussed in previous sections. Once a high-level security 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 NSF's 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., Palislaamovic, 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., Palislaamovic, 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:

- o 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.
- o 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 separately. The security policies can now be set recursively (daily, weekly, and monthly).
- o "policy-role" now has the access-profile container, and privilege can be set separately per profile.
- o "policy-user" information, such as email and password is newly defined by regular expressions.
- o "authentication-method" in "policy-mgmt-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.
- o "Certificate-type" has been re-defined to include common certificate extensions, such as ".CRT", "CER", and "KEY".
- o Used groupings to represent the groups in the Endpoint groups.
- o 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

I2NSF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 25, 2019

J. Kim
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
Q. Lin
Huawei
March 24, 2019

I2NSF Network Security Function-Facing Interface YANG Data Model
draft-ietf-i2nsf-nsf-facing-interface-dm-04

Abstract

This document defines a YANG data model for configuring security policy rules on network security functions. The YANG data model in this document is corresponding to the information model for Network Security Functions (NSF)-Facing Interface in Interface to Network Security Functions (I2NSF).

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 25, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology	3
3.1. Tree Diagrams	3
4. YANG Tree Diagram	4
4.1. General I2NSF Security Policy Rule	4
4.2. Event Clause	6
4.3. Condition Clause	7
4.4. Action Clause	13
5. YANG Data Module	14
5.1. I2NSF NSF-Facing Interface YANG Data Module	14
6. IANA Considerations	88
7. Security Considerations	88
8. References	88
8.1. Normative References	89
8.2. Informative References	90
Appendix A. Configuration Examples	91
A.1. Security Requirement 1: Block SNS Access during Business Hours	91
A.2. Security Requirement 2: Block Malicious VoIP/VoLTE Packets Coming to the Company	94
A.3. Security Requirement 3: Mitigate HTTP and HTTPS Flood Attacks on a Company Web Server	97
Appendix B. Changes from draft-ietf-i2nsf-nsf-facing-interface-dm-03	100
Appendix C. Acknowledgments	100
Appendix D. Contributors	100
Authors' Addresses	101

1. Introduction

This document defines a YANG [RFC6020][RFC7950] data model for security policy rule configuration of network security devices. The YANG data model is corresponding to the information model [i2nsf-nsf-cap-im] for Network Security Functions (NSF) facing interface in Interface to Network Security Functions (I2NSF). The YANG data model in this document focuses on security policy configuration for generic network security functions. Note that security policy configuration for advanced network security functions are written in [i2nsf-advanced-nsf-dm].

This YANG data model uses an "Event-Condition-Action" (ECA) policy model that is used as the basis for the design of I2NSF Policy described in [RFC8329] and [i2nsf-nsf-cap-im]. Rules.

The "ietf-i2nsf-policy-rule-for-nsf" YANG module defined in this document provides the following features.

- o Configuration for general security policy rule of generic network security function.
- o Configuration for an event clause of generic network security function.
- o Configuration for a condition clause of generic network security function.
- o Configuration for an action clause of generic network security function.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119][RFC8174].

3. Terminology

This document uses the terminology described in [i2nsf-nsf-cap-im][RFC8431][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

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

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams [RFC8340] is as follows:

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

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

4. YANG Tree Diagram

This section shows an YANG tree diagram of generic network security functions. Note that a detailed data model for the configuration of the advanced network security functions is described in [i2nsf-advanced-nsf-dm]. The section describes the following subjects:

- o General I2NSF security policy rule of generic network security function.
- o An event clause of generic network security function.
- o A condition clause of generic network security function.
- o An action clause of generic network security function.

4.1. General I2NSF Security Policy Rule

This section shows YANG tree diagram for general I2NSF security policy rule.

```

module: ietf-i2nsf-policy-rule-for-nsf
  +--rw i2nsf-security-policy
    +--rw system-policy* [system-policy-name]
      +--rw system-policy-name      string
      +--rw priority-usage?         identityref
      +--rw resolution-strategy?    identityref
      +--rw default-action?         identityref
      +--rw rules* [rule-name]
        +--rw rule-name              string
        +--rw rule-description?      string
        +--rw rule-priority?         uint8
        +--rw rule-enable?           boolean
        +--rw rule-session-aging-time? uint16
        +--rw rule-long-connection
          +--rw enable?      boolean
          +--rw during?     uint16
        +--rw time-zone
          +--rw absolute-time-zone
            +--rw start-time?  start-time-type
            +--rw end-time?    end-time-type
          +--rw periodic-time-zone
            +--rw day
              +--rw every-day?    boolean
              +--rw specific-day* day-type
            +--rw month
              +--rw every-month?  boolean
              +--rw specific-month* month-type
        +--rw event-clause-container
          ...
        +--rw condition-clause-container
          ...
        +--rw action-clause-container
          ...
      +--rw rule-group
        +--rw groups* [group-name]
          +--rw group-name      string
          +--rw rule-range
            +--rw start-rule?  string
            +--rw end-rule?    string
          +--rw enable?        boolean

```

Figure 1: YANG Tree Diagram for Network Security Policy

This YANG tree diagram shows general I2NSF security policy rule for generic network security functions.

The system policy represents there could be multiple system policies in one NSF, and each system policy is used by one virtual instance of the NSF/device. The system policy includes system policy name, priority usage, resolution strategy, default action, and rules.

A resolution strategy is used to decide how to resolve conflicts that occur between the actions of the same or different policy rules that are matched and contained in this particular NSF. The resolution strategy is defined as First Matching Rule (FMR), Last Matching Rule (LMR), Prioritized Matching Rule (PMR) with Errors (PMRE), and Prioritized Matching Rule with No Errors (PMRN). The resolution strategy can be extended according to specific vendor action features. The resolution strategy is described in detail in [i2nsf-nsf-cap-im].

A default action is used to execute I2NSF policy rule when no rule matches a packet. The default action is defined as pass, drop, reject, alert, and mirror. The default action can be extended according to specific vendor action features. The default action is described in detail in [i2nsf-nsf-cap-im].

The rules include rule name, rule description, rule priority, rule enable, time zone, event clause container, condition clause container, and action clause container.

4.2. Event Clause

This section shows YANG tree diagram for an event clause of I2NSF security policy rule.

```

module: ietf-i2nsf-policy-rule-for-nsf
  +--rw i2nsf-security-policy
    +--rw system-policy* [system-policy-name]
      ...
    +--rw rules* [rule-name]
      ...
      +--rw event-clause-container
        +--rw event-clause-description?  string
        +--rw event-clauses
          +--rw system-event*  identityref
          +--rw system-alarm*  identityref
        +--rw condition-clause-container
          ...
        +--rw action-clause-container
          ...
      +--rw rule-group
        ...

```

Figure 2: YANG Tree Diagram for Network Security Policy

This YANG tree diagram shows an event clause of I2NSF security policy rule for generic network security functions. An event clause is any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. An event clause is used to trigger the evaluation of the condition clause of the I2NSF Policy Rule. The event clause is defined as system event and system alarm. The event clause can be extended according to specific vendor event features. The event clause is described in detail in [i2nsf-nsf-cap-im].

4.3. Condition Clause

This section shows YANG tree diagram for a condition clause of I2NSF security policy rule.

```

module: ietf-i2nsf-policy-rule-for-nsf
  +--rw i2nsf-security-policy
    ...
    +--rw rules* [rule-name]
      ...
      +--rw event-clause-container
        | ...
      +--rw condition-clause-container
        +--rw condition-clause-description?  string
        +--rw packet-security-ipv4-condition
          +--rw pkt-sec-ipv4-header-length
            +--rw (match-type)?

```

```

+---: (exact-match)
|   +---rw ipv4-header-length*          uint8
+---: (range-match)
|   +---rw range-ipv4-header-length*
[start-ipv4-header-length end-ipv4-header-length]
|       +---rw start-ipv4-header-length    uint8
|       +---rw end-ipv4-header-length      uint8
+---rw pkt-sec-ipv4-tos*                  identityref
+---rw pkt-sec-ipv4-total-length
|   +---rw (match-type)?
|   +---: (exact-match)
|   |   +---rw ipv4-total-length*          uint16
|   +---: (range-match)
|   |   +---rw range-ipv4-total-length*
[start-ipv4-total-length end-ipv4-total-length]
|       +---rw start-ipv4-total-length      uint16
|       +---rw end-ipv4-total-length        uint16
+---rw pkt-sec-ipv4-id*                   uint16
+---rw pkt-sec-ipv4-fragment-flags*       identityref
+---rw pkt-sec-ipv4-fragment-offset
|   +---rw (match-type)?
|   +---: (exact-match)
|   |   +---rw ipv4-fragment-offset*        uint16
|   +---: (range-match)
|   |   +---rw range-ipv4-fragment-offset*
[start-ipv4-fragment-offset end-ipv4-fragment-offset]
|       +---rw start-ipv4-fragment-offset    uint16
|       +---rw end-ipv4-fragment-offset      uint16
+---rw pkt-sec-ipv4-ttl
|   +---rw (match-type)?
|   +---: (exact-match)
|   |   +---rw ipv4-ttl*                    uint8
|   +---: (range-match)
|   |   +---rw range-ipv4-ttl*
[start-ipv4-ttl end-ipv4-ttl]
|       +---rw start-ipv4-ttl                uint8
|       +---rw end-ipv4-ttl                  uint8
+---rw pkt-sec-ipv4-protocol*              identityref
+---rw pkt-sec-ipv4-src
|   +---rw (match-type)?
|   +---: (exact-match)
|   |   +---rw ipv4-address* [ipv4]
|   |   +---rw ipv4                      inet:ipv4-address
|   |   +---rw (subnet)?
|   |   +---: (prefix-length)
|   |   |   +---rw prefix-length?          uint8
|   |   +---: (netmask)
|   |   |   +---rw netmask? yang:dotted-quad

```

```

    +---:(range-match)
      +---rw range-ipv4-address*
    [start-ipv4-address end-ipv4-address]
      +---rw start-ipv4-address    inet:ipv4-address
      +---rw end-ipv4-address      inet:ipv4-address
    +---rw pkt-sec-ipv4-dest
      +---rw (match-type)?
      +---:(exact-match)
        +---rw ipv4
          +---rw ipv4-address* [ipv4]
          +---rw ipv4          inet:ipv4-address
          +---rw (subnet)?
            +---:(prefix-length)
              | +---rw prefix-length?    uint8
            +---:(netmask)
              +---rw netmask?    yang:dotted-quad
      +---:(range-match)
        +---rw range-ipv4-address*
    [start-ipv4-address end-ipv4-address]
      +---rw start-ipv4-address    inet:ipv4-address
      +---rw end-ipv4-address      inet:ipv4-address
    +---rw pkt-sec-ipv4-iptests*    identityref
    +---rw pkt-sec-ipv4-sameip?      boolean
    +---rw pkt-sec-ipv4-geoip*       string
  +---rw packet-security-ipv6-condition
    +---rw pkt-sec-ipv6-traffic-class* identityref
    +---rw pkt-sec-ipv6-flow-label
      +---rw (match-type)?
      +---:(exact-match)
        | +---rw ipv6-flow-label*    uint32
      +---:(range-match)
        +---rw range-ipv6-flow-label*
    [start-ipv6-flow-label end-ipv6-flow-label]
      +---rw start-ipv6-flow-label    uint32
      +---rw end-ipv6-flow-label      uint32
    +---rw pkt-sec-ipv6-payload-length
      +---rw (match-type)?
      +---:(exact-match)
        | +---rw ipv6-payload-length*    uint16
      +---:(range-match)
        +---rw range-ipv6-payload-length*
    [start-ipv6-payload-length end-ipv6-payload-length]
      +---rw start-ipv6-payload-length    uint16
      +---rw end-ipv6-payload-length      uint16
    +---rw pkt-sec-ipv6-next-header*    identityref
    +---rw pkt-sec-ipv6-hop-limit
      +---rw (match-type)?
      +---:(exact-match)

```

```

| | | | | +---rw ipv6-hop-limit*          uint8
| | | | | +---:(range-match)
| | | | | +---rw range-ipv6-hop-limit*
| | | | | [start-ipv6-hop-limit end-ipv6-hop-limit]
| | | | | +---rw start-ipv6-hop-limit      uint8
| | | | | +---rw end-ipv6-hop-limit        uint8
| | | | | +---rw pkt-sec-ipv6-src
| | | | | +---rw (match-type)?
| | | | | +---:(exact-match)
| | | | | | +---rw ipv6
| | | | | | +---rw ipv6-address* [ipv6]
| | | | | | +---rw ipv6                inet:ipv6-address
| | | | | | +---rw prefix-length?      uint8
| | | | | +---:(range-match)
| | | | | +---rw range-ipv6-address*
| | | | | [start-ipv6-address end-ipv6-address]
| | | | | +---rw start-ipv6-address      inet:ipv6-address
| | | | | +---rw end-ipv6-address        inet:ipv6-address
| | | | | +---rw pkt-sec-ipv6-dest
| | | | | +---rw (match-type)?
| | | | | +---:(exact-match)
| | | | | | +---rw ipv6-address* [ipv6]
| | | | | | +---rw ipv6                inet:ipv6-address
| | | | | | +---rw prefix-length?      uint8
| | | | | +---:(range-match)
| | | | | +---rw range-ipv6-address*
| | | | | [start-ipv6-address end-ipv6-address]
| | | | | +---rw start-ipv6-address      inet:ipv6-address
| | | | | +---rw end-ipv6-address        inet:ipv6-address
| | | | | +---rw packet-security-tcp-condition
| | | | | +---rw pkt-sec-tcp-src-port-num
| | | | | +---rw (match-type)?
| | | | | +---:(exact-match)
| | | | | | +---rw port-num*            inet:port-number
| | | | | +---:(range-match)
| | | | | +---rw range-port-num*
| | | | | [start-port-num end-port-num]
| | | | | +---rw start-port-num          inet:port-number
| | | | | +---rw end-port-num            inet:port-number
| | | | | +---rw pkt-sec-tcp-dest-port-num
| | | | | +---rw (match-type)?
| | | | | +---:(exact-match)
| | | | | | +---rw port-num*            inet:port-number
| | | | | +---:(range-match)
| | | | | +---rw range-port-num*
| | | | | [start-port-num end-port-num]
| | | | | +---rw start-port-num          inet:port-number
| | | | | +---rw end-port-num            inet:port-number

```

```

    +--rw pkt-sec-tcp-seq-num
      +--rw (match-type)?
        +--:(exact-match)
          +--rw tcp-seq-num*          uint32
        +--:(range-match)
          +--rw range-tcp-seq-num*
      [start-tcp-seq-num end-tcp-seq-num]
        +--rw start-tcp-seq-num      uint32
        +--rw end-tcp-seq-num        uint32
    +--rw pkt-sec-tcp-ack-num
      +--rw (match-type)?
        +--:(exact-match)
          +--rw tcp-ack-num*          uint32
        +--:(range-match)
          +--rw range-tcp-ack-num*
      [start-tcp-ack-num end-tcp-ack-num]
        +--rw start-tcp-ack-num      uint32
        +--rw end-tcp-ack-num        uint32
    +--rw pkt-sec-tcp-window-size
      +--rw (match-type)?
        +--:(exact-match)
          +--rw tcp-window-size*      uint16
        +--:(range-match)
          +--rw range-tcp-window-size*
      [start-tcp-window-size end-tcp-window-size]
        +--rw start-tcp-window-size  uint16
        +--rw end-tcp-window-size    uint16
    +--rw pkt-sec-tcp-flags*          identityref
    +--rw packet-security-udp-condition
    +--rw pkt-sec-udp-src-port-num
      +--rw (match-type)?
        +--:(exact-match)
          +--rw port-num*             inet:port-number
        +--:(range-match)
          +--rw range-port-num*
      [start-port-num end-port-num]
        +--rw start-port-num          inet:port-number
        +--rw end-port-num            inet:port-number
    +--rw pkt-sec-udp-dest-port-num
      +--rw (match-type)?
        +--:(exact-match)
          +--rw port-num*             inet:port-number
        +--:(range-match)
          +--rw range-port-num*
      [start-port-num end-port-num]
        +--rw start-port-num          inet:port-number
        +--rw end-port-num            inet:port-number
    +--rw pkt-sec-udp-total-length

```



```

    +--rw (match-type)?
      +--:(exact-match)
        | +--rw udp-total-length*          uint32
      +--:(range-match)
        +--rw range-udp-total-length*
[start-udp-total-length end-udp-total-length]
        +--rw start-udp-total-length      uint32
        +--rw end-udp-total-length        uint32
+--rw packet-security-icmp-condition
| +--rw pkt-sec-icmp-type*  identityref
+--rw packet-security-http-condition
| +--rw pkt-sec-uri-content*  string
| +--rw pkt-sec-url-content*  string
+--rw packet-security-voice-condition
| +--rw pkt-sec-src-voice-id*  string
| +--rw pkt-sec-dest-voice-id*  string
| +--rw pkt-sec-user-agent*  string
+--rw packet-security-ddos-condition
+--rw pkt-sec-alert-rate?  uint32
+--rw packet-payload-condition
| +--rw packet-payload-description?  string
| +--rw pkt-payload-content*  string
+--rw acl-number*  uint32
+--rw application-condition
| +--rw application-description?  string
| +--rw application-object*  string
| +--rw application-group*  string
| +--rw application-label*  string
| +--rw category
| +--rw application-category*
[name application-subcategory]
| +--rw name  string
| +--rw application-subcategory  string
+--rw target-condition
| +--rw target-description?  string
| +--rw device-sec-context-cond
| +--rw target-device*  identityref
+--rw users-condition
+--rw users-description?  string
+--rw user
| +--rw (user-name)?
| +--:(tenant)
| | +--rw tenant  uint8
| +--:(vn-id)
| | +--rw vn-id  uint8
+--rw group
| +--rw (group-name)?
| +--:(tenant)

```

Figure 3: YANG Tree Diagram for Network Security Policy

This YANG tree diagram shows an condition clause of I2NSF security policy rule for generic network security functions. A condition clause is defined as a set of attributes, features, and/or values that are to be compared with a set of known attributes, features, and/or values in order to determine whether or not the set of actions in that (imperative) I2NSF policy rule can be executed or not. The condition clause is classified as conditions of generic network security functions and advanced network security functions. The condition clause of generic network security functions is defined as packet security IPv4 condition, packet security IPv6 condition, packet security tcp condition, and packet security icmp condition. The condition clause of advanced network security functions is defined as packet security http condition, packet security voice condition, and packet security ddos condition. Note that this document deals only with simple conditions of advanced network security functions. The condition clauses of advanced network security functions are described in detail in [i2nsf-advanced-nsf-dm]. The condition clause can be extended according to specific vendor condition features. The condition clause is described in detail in [i2nsf-nsf-cap-im].

This section shows YANG tree diagram for an action clause of I2NSF security policy rule.

```

module: ietf-i2nsf-policy-rule-for-nsf
  +--rw i2nsf-security-policy
    ...
    +--rw rules* [rule-name]
      ...
      +--rw event-clause-container
      |   ...
      +--rw condition-clause-container
      |   ...
      +--rw action-clause-container
        +--rw action-clause-description?   string
        +--rw packet-action
          +--rw ingress-action?             identityref
          +--rw egress-action?              identityref
          +--rw log-action?                  identityref
        +--rw advanced-action
          +--rw content-security-control*    identityref
          +--rw attack-mitigation-control*  identityref

```

Figure 4: YANG Tree Diagram for Network Security Policy

This YANG tree diagram shows an action clause of I2NSF security policy rule for generic network security functions. An action is used to control and monitor aspects of flow-based NSFs when the event and condition clauses are satisfied. NSFs provide security services by executing various actions. The action clause is defined as ingress action, egress action, log action, and advanced action for additional inspection. The advanced action is described in detail in [RFC8329] and [i2nsf-nsf-cap-im]. The action clause can be extended according to specific vendor action features. The action clause is described in detail in [i2nsf-nsf-cap-im].

5. YANG Data Module

5.1. I2NSF NSF-Facing Interface YANG Data Module

This section introduces an YANG data module for configuration of security policy rules on network security functions.

<CODE BEGINS> file "ietf-i2nsf-policy-rule-for-nsf@2019-03-24.yang"

```

module ietf-i2nsf-policy-rule-for-nsf {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf";
  prefix
    iiprfn;

```

```
import ietf-inet-types{
  prefix inet;
  reference "RFC 6991";
}
import ietf-yang-types{
  prefix yang;
  reference "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.dunbar@huawei.com>

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

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

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

description
  "This module defines a YANG data module for network security
   functions.

   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 8341; see
   the RFC itself for full legal notices.";
```

```
revision "2019-03-24"{
  description "Initial revision.";
  reference
    "RFC XXXX: I2NSF Network Security Function-Facing Interface
     YANG Data Model";
}

/*
 * Identities
 */

identity priority-usage-type {
  description
    "Base identity for priority usage type.";
}

identity priority-by-order {
  base priority-usage-type;
  description
    "Identity for priority by order";
}

identity priority-by-number {
  base priority-usage-type;
  description
    "Identity for priority by number";
}

identity event {
  description
    "Base identity for event of policy.";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
     - Event";
}

identity system-event {
  base event;
  description
    "Identity for system event";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
     - System event";
}

identity system-alarm {
  base event;
  description
```

```
    "Identity for system alarm";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System alarm";
}

identity access-violation {
  base system-event;
  description
    "Identity for access violation
      among system events";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System event";
}

identity configuration-change {
  base system-event;
  description
    "Identity for configuration change
      among system events";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System event";
}

identity memory-alarm {
  base system-alarm;
  description
    "Identity for memory alarm
      among system alarms";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System alarm";
}

identity cpu-alarm {
  base system-alarm;
  description
    "Identity for cpu alarm
      among system alarms";
  reference
    "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System alarm";
}

identity disk-alarm {
  base system-alarm;
```

```
    description
      "Identity for disk alarm
      among system alarms";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System alarm";
  }

  identity hardware-alarm {
    base system-alarm;
    description
      "Identity for hardware alarm
      among system alarms";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System alarm";
  }

  identity interface-alarm {
    base system-alarm;
    description
      "Identity for interface alarm
      among system alarms";
    reference
      "draft-hong-i2nsf-nsf-monitoring-data-model-06
      - System alarm";
  }

  identity type-of-service {
    description
      "Base identity for type of service of IPv4";
    reference
      "RFC 791: Internet Protocol - Type of Service";
  }

  identity traffic-class {
    description
      "Base identity for traffic-class of IPv6";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Traffic Class";
  }

  identity normal {
    base type-of-service;
    base traffic-class;
    description
      "Identity for normal";
```

```
reference
  "RFC 791: Internet Protocol - Type of Service
   RFC 2460: Internet Protocol, Version 6 (IPv6)
   Specification - Traffic Class";
}

identity minimize-cost {
  base type-of-service;
  base traffic-class;
  description
    "Identity for minimize cost";
  reference
    "RFC 791: Internet Protocol - Type of Service
     RFC 2460: Internet Protocol, Version 6 (IPv6)
     Specification - Traffic Class";
}

identity maximize-reliability {
  base type-of-service;
  base traffic-class;
  description
    "Identity for maximize reliability";
  reference
    "RFC 791: Internet Protocol - Type of Service
     RFC 2460: Internet Protocol, Version 6 (IPv6)
     Specification - Traffic Class";
}

identity maximize-throughput {
  base type-of-service;
  base traffic-class;
  description
    "Identity for maximize throughput";
  reference
    "RFC 791: Internet Protocol - Type of Service
     RFC 2460: Internet Protocol, Version 6 (IPv6)
     Specification - Traffic Class";
}

identity minimize-delay {
  base type-of-service;
  base traffic-class;
  description
    "Identity for minimize delay";
  reference
    "RFC 791: Internet Protocol - Type of Service
     RFC 2460: Internet Protocol, Version 6 (IPv6)
     Specification - Traffic Class";
```



```
}

identity maximize-security {
  base type-of-service;
  base traffic-class;
  description
    "Identity for maximize security";
  reference
    "RFC 791: Internet Protocol - Type of Service
     RFC 2460: Internet Protocol, Version 6 (IPv6)
     Specification - Traffic Class";
}

identity fragmentation-flags-type {
  description
    "Base identity for fragmentation flags type";
  reference
    "RFC 791: Internet Protocol - Fragmentation Flags";
}

identity fragment {
  base fragmentation-flags-type;
  description
    "Identity for fragment";
  reference
    "RFC 791: Internet Protocol - Fragmentation Flags";
}

identity no-fragment {
  base fragmentation-flags-type;
  description
    "Identity for no fragment";
  reference
    "RFC 791: Internet Protocol - Fragmentation Flags";
}

identity reserved {
  base fragmentation-flags-type;
  description
    "Identity for reserved";
  reference
    "RFC 791: Internet Protocol - Fragmentation Flags";
}

identity protocol {
  description
    "Base identity for protocol of IPv4";
  reference
```

```
    "RFC 790: Assigned numbers - Assigned Internet
      Protocol Number
      RFC 791: Internet Protocol - Protocol";
  }

  identity next-header {
    description
      "Base identity for next header of IPv6";
    reference
      "RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next Header";
  }

  identity icmp {
    base protocol;
    base next-header;
    description
      "Identity for icmp";
    reference
      "RFC 790: - Assigned numbers - Assigned Internet
        Protocol Number
        RFC 791: Internet Protocol - Type of Service
        RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next Header";
  }

  identity igmp {
    base protocol;
    base next-header;
    description
      "Identity for igmp";
    reference
      "RFC 790: - Assigned numbers - Assigned Internet
        Protocol Number
        RFC 791: Internet Protocol - Type of Service
        RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next Header";
  }

  identity tcp {
    base protocol;
    base next-header;
    description
      "Identity for tcp";
    reference
      "RFC 790: - Assigned numbers - Assigned Internet
        Protocol Number
        RFC 791: Internet Protocol - Type of Service
```

```
        RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next Header";
    }

    identity igmp {
        base protocol;
        base next-header;
        description
            "Identity for igmp";
        reference
            "RFC 790: - Assigned numbers - Assigned Internet
            Protocol Number
            RFC 791: Internet Protocol - Type of Service
            RFC 2460: Internet Protocol, Version 6 (IPv6)
            Specification - Next Header";
    }

    identity udp {
        base protocol;
        base next-header;
        description
            "Identity for udp";
        reference
            "RFC 790: - Assigned numbers - Assigned Internet
            Protocol Number
            RFC 791: Internet Protocol - Type of Service
            RFC 2460: Internet Protocol, Version 6 (IPv6)
            Specification - Next Header";
    }

    identity gre {
        base protocol;
        base next-header;
        description
            "Identity for gre";
        reference
            "RFC 790: - Assigned numbers - Assigned Internet
            Protocol Number
            RFC 791: Internet Protocol - Type of Service
            RFC 2460: Internet Protocol, Version 6 (IPv6)
            Specification - Next Header";
    }

    identity esp {
        base protocol;
        base next-header;
        description
            "Identity for esp";
```

```
reference
  "RFC 790: - Assigned numbers - Assigned Internet
    Protocol Number
  RFC 791: Internet Protocol - Type of Service
  RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Next Header";
}

identity ah {
  base protocol;
  base next-header;
  description
    "Identity for ah";
  reference
    "RFC 790: - Assigned numbers - Assigned Internet
      Protocol Number
    RFC 791: Internet Protocol - Type of Service
    RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Next Header";
}

identity mobile {
  base protocol;
  base next-header;
  description
    "Identity for mobile";
  reference
    "RFC 790: - Assigned numbers - Assigned Internet
      Protocol Number
    RFC 791: Internet Protocol - Type of Service
    RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Next Header";
}

identity tlsp {
  base protocol;
  base next-header;
  description
    "Identity for tlsp";
  reference
    "RFC 790: - Assigned numbers - Assigned Internet
      Protocol Number
    RFC 791: Internet Protocol - Type of Service
    RFC 2460: Internet Protocol, Version 6 (IPv6)
      Specification - Next Header";
}

identity skip {
```

```
base protocol;
base next-header;
description
  "Identity for skip";
reference
  "RFC 790: - Assigned numbers - Assigned Internet
  Protocol Number
  RFC 791: Internet Protocol - Type of Service
  RFC 2460: Internet Protocol, Version 6 (IPv6)
  Specification - Next Header";
}

identity ipv6-icmp {
  base protocol;
  base next-header;
  description
    "Identity for IPv6 icmp ";
  reference
    "RFC 790: - Assigned numbers - Assigned Internet
    Protocol Number
    RFC 791: Internet Protocol - Type of Service
    RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Next Header";
}

identity eigrp {
  base protocol;
  base next-header;
  description
    "Identity for eigrp";
  reference
    "RFC 790: - Assigned numbers - Assigned Internet
    Protocol Number
    RFC 791: Internet Protocol - Type of Service
    RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Next Header";
}

identity ospf {
  base protocol;
  base next-header;
  description
    "Identity for ospf";
  reference
    "RFC 790: - Assigned numbers - Assigned Internet
    Protocol Number
    RFC 791: Internet Protocol - Type of Service
```

```
        RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next Header";
    }

    identity l2tp {
        base protocol;
        base next-header;
        description
            "Identity for l2tp";
        reference
            "RFC 790: - Assigned numbers - Assigned Internet
            Protocol Number
            RFC 791: Internet Protocol - Type of Service
            RFC 2460: Internet Protocol, Version 6 (IPv6)
            Specification - Next Header";
    }

    identity ipopts {
        description
            "Base identity for IP options";
        reference
            "RFC 791: Internet Protocol - Options";
    }

    identity rr {
        base ipopts;
        description
            "Identity for record route";
        reference
            "RFC 791: Internet Protocol - Options";
    }

    identity eol {
        base ipopts;
        description
            "Identity for end of list";
        reference
            "RFC 791: Internet Protocol - Options";
    }

    identity nop {
        base ipopts;
        description
            "Identity for no operation";
        reference
            "RFC 791: Internet Protocol - Options";
    }
}
```

```
identity ts {
  base ipopts;
  description
    "Identity for time stamp";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity sec {
  base ipopts;
  description
    "Identity for IP security";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity esec {
  base ipopts;
  description
    "Identity for IP extended security";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity lsrr {
  base ipopts;
  description
    "Identity for loose source routing";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity ssrr {
  base ipopts;
  description
    "Identity for strict source routing";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity satid {
  base ipopts;
  description
    "Identity for stream identifier";
  reference
    "RFC 791: Internet Protocol - Options";
}
```

```
identity any {
  base ipopts;
  description
    "Identity for which any IP options are set";
  reference
    "RFC 791: Internet Protocol - Options";
}

identity tcp-flags {
  description
    "Base identity for tcp flags";
  reference
    "RFC 793: Transmission Control Protocol - Flags";
}

identity cwr {
  base tcp-flags;
  description
    "Identity for congestion window reduced";
  reference
    "RFC 793: Transmission Control Protocol - Flags";
}

identity ecn {
  base tcp-flags;
  description
    "Identity for explicit congestion notification";
  reference
    "RFC 793: Transmission Control Protocol - Flags";
}

identity urg {
  base tcp-flags;
  description
    "Identity for urgent";
  reference
    "RFC 793: Transmission Control Protocol - Flags";
}

identity ack {
  base tcp-flags;
  description
    "Identity for acknowledgement";
  reference
    "RFC 793: Transmission Control Protocol - Flags";
}

identity psh {
```



```
    base tcp-flags;
    description
        "Identity for push";
    reference
        "RFC 793: Transmission Control Protocol - Flags";
}

identity rst {
    base tcp-flags;
    description
        "Identity for reset";
    reference
        "RFC 793: Transmission Control Protocol - Flags";
}

identity syn {
    base tcp-flags;
    description
        "Identity for synchronize";
    reference
        "RFC 793: Transmission Control Protocol - Flags";
}

identity fin {
    base tcp-flags;
    description
        "Identity for finish";
    reference
        "RFC 793: Transmission Control Protocol - Flags";
}

identity icmp-type {
    description
        "Base identity for icmp types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity echo-reply {
    base icmp-type;
    description
        "Identity for echo reply";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity destination-unreachable {
    base icmp-type;
```

```
    description
      "Identity for destination unreachable";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity source-quench {
    base icmp-type;
    description
      "Identity for source quench";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity redirect {
    base icmp-type;
    description
      "Identity for redirect";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity alternate-host-address {
    base icmp-type;
    description
      "Identity for alternate host address";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity echo {
    base icmp-type;
    description
      "Identity for echo";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity router-advertisement {
    base icmp-type;
    description
      "Identity for router advertisement";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity router-solicitation {
    base icmp-type;
```

```
    description
      "Identity for router solicitation";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity time-exceeded {
    base icmp-type;
    description
      "Identity for time exceeded";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity parameter-problem {
    base icmp-type;
    description
      "Identity for parameter problem";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity timestamp {
    base icmp-type;
    description
      "Identity for timestamp";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity timestamp-reply {
    base icmp-type;
    description
      "Identity for timestamp reply";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity information-request {
    base icmp-type;
    description
      "Identity for information request";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity information-reply {
    base icmp-type;
```

```
    description
      "Identity for information reply";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity address-mask-request {
    base icmp-type;
    description
      "Identity for address mask request";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity address-mask-reply {
    base icmp-type;
    description
      "Identity for address mask reply";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity traceroute {
    base icmp-type;
    description
      "Identity for traceroute";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity datagram-conversion-error {
    base icmp-type;
    description
      "Identity for datagram conversion error";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity mobile-host-redirect {
    base icmp-type;
    description
      "Identity for mobile host redirect";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity ipv6-where-are-you {
    base icmp-type;
```

```
    description
      "Identity for IPv6 where are you";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity ipv6-i-am-here {
    base icmp-type ;
    description
      "Identity for IPv6 i am here";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity mobile-registration-request {
    base icmp-type;
    description
      "Identity for mobile registration request";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity mobile-registration-reply {
    base icmp-type;
    description
      "Identity for mobile registration reply";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity domain-name-request {
    base icmp-type;
    description
      "Identity for domain name request";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity domain-name-reply {
    base icmp-type;
    description
      "Identity for domain name reply";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity iskip {
    base icmp-type;
```

```
    description
      "Identity for icmp skip";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity photuris {
    base icmp-type;
    description
      "Identity for photuris";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity experimental-mobility-protocols {
    base icmp-type;
    description
      "Identity for experimental mobility protocols";
    reference
      "RFC 792: Internet Control Message Protocol";
  }

  identity extended-echo-request {
    base icmp-type;
    description
      "Identity for extended echo request";
    reference
      "RFC 792: Internet Control Message Protocol
      RFC 8335: PROBE: A Utility for Probing Interfaces";
  }

  identity extended-echo-reply {
    base icmp-type;
    description
      "Identity for extended echo reply";
    reference
      "RFC 792: Internet Control Message Protocol
      RFC 8335: PROBE: A Utility for Probing Interfaces";
  }

  identity net-unreachable {
    base icmp-type;
    description
      "Identity for net unreachable
      in destination unreachable types";
    reference
      "RFC 792: Internet Control Message Protocol";
  }
```

```
identity host-unreachable {
  base icmp-type;
  description
    "Identity for host unreachable
     in destination unreachable types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity protocol-unreachable {
  base icmp-type;
  description
    "Identity for protocol unreachable
     in destination unreachable types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity port-unreachable {
  base icmp-type;
  description
    "Identity for port unreachable
     in destination unreachable types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity fragment-set {
  base icmp-type;
  description
    "Identity for fragmentation set
     in destination unreachable types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity source-route-failed {
  base icmp-type;
  description
    "Identity for source route failed
     in destination unreachable types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity destination-network-unknown {
  base icmp-type;
  description
```

```
        "Identity for destination network unknown
        in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity destination-host-unknown {
    base icmp-type;
    description
        "Identity for destination host unknown
        in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity source-host-isolated {
    base icmp-type;
    description
        "Identity for source host isolated
        in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity communication-prohibited-with-destination-network {
    base icmp-type;
    description
        "Identity for which communication with destination network
        is administratively prohibited in destination unreachable
        types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity communication-prohibited-with-destination-host {
    base icmp-type;
    description
        "Identity for which communication with destination host
        is administratively prohibited in destination unreachable
        types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity destination-network-unreachable-for-tos {
    base icmp-type;
    description
        "Identity for destination network unreachable
```



```
        for type of service in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity destination-host-unreachable-for-tos {
    base icmp-type;
    description
        "Identity for destination host unreachable
        for type of service in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity communication-prohibited {
    base icmp-type;
    description
        "Identity for communication administratively prohibited
        in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity host-precedence-violation {
    base icmp-type;
    description
        "Identity for host precedence violation
        in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity precedence-cutoff-in-effect {
    base icmp-type;
    description
        "Identity for precedence cutoff in effect
        in destination unreachable types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity redirect-datagram-for-the-network {
    base icmp-type;
    description
        "Identity for redirect datagram for the network
        (or subnet) in redirect types";
    reference
        "RFC 792: Internet Control Message Protocol";
}
```

```
}

identity redirect-datagram-for-the-host {
  base icmp-type;
  description
    "Identity for redirect datagram for the host
    in redirect types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity redirect-datagram-for-the-tos-and-network {
  base icmp-type;
  description
    "Identity for redirect datagram for the type of
    service and network in redirect types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity redirect-datagram-for-the-tos-and-host {
  base icmp-type;
  description
    "Identity for redirect datagram for the type of
    service and host in redirect types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity normal-router-advertisement {
  base icmp-type;
  description
    "Identity for normal router advertisement
    in router advertisement types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity does-not-route-common-traffic {
  base icmp-type;
  description
    "Identity for does not route common traffic
    in router advertisement types";
  reference
    "RFC 792: Internet Control Message Protocol";
}

identity time-to-live-exceeded-in-transit {
```

```
    base icmp-type;
    description
        "Identity for time to live exceeded in transit
        in time exceeded types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity fragment-reassembly-time-exceeded {
    base icmp-type;
    description
        "Identity for fragment reassembly time exceeded
        in time exceeded types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity pointer-indicates-the-error {
    base icmp-type;
    description
        "Identity for pointer indicates the error
        in parameter problem types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity missing-a-required-option {
    base icmp-type;
    description
        "Identity for missing a required option
        in parameter problem types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity bad-length {
    base icmp-type;
    description
        "Identity for bad length
        in parameter problem types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity bad-spi {
    base icmp-type;
    description
        "Identity for bad spi
```

```
        in photuris types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity authentication-failed {
    base icmp-type;
    description
        "Identity for authentication failed
        in photuris types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity decompression-failed {
    base icmp-type;
    description
        "Identity for decompression failed
        in photuris types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity decryption-failed {
    base icmp-type;
    description
        "Identity for decryption failed
        in photuris types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity need-authentication {
    base icmp-type;
    description
        "Identity for need authentication
        in photuris types";
    reference
        "RFC 792: Internet Control Message Protocol";
}

identity need-authorization {
    base icmp-type;
    description
        "Identity for need authorization
        in photuris types";
    reference
        "RFC 792: Internet Control Message Protocol";
}
```

```
}

identity req-no-error {
  base icmp-type;
  description
    "Identity for request with no error
     in extended echo request types";
  reference
    "RFC 792: Internet Control Message Protocol
     RFC 8335: PROBE: A Utility for Probing Interfaces";
}

identity rep-no-error {
  base icmp-type;
  description
    "Identity for reply with no error
     in extended echo reply types";
  reference
    "RFC 792: Internet Control Message Protocol
     RFC 8335: PROBE: A Utility for Probing Interfaces";
}

identity malformed-query {
  base icmp-type;
  description
    "Identity for malformed query
     in extended echo reply types";
  reference
    "RFC 792: Internet Control Message Protocol
     RFC 8335: PROBE: A Utility for Probing Interfaces";
}

identity no-such-interface {
  base icmp-type;
  description
    "Identity for no such interface
     in extended echo reply types";
  reference
    "RFC 792: Internet Control Message Protocol
     RFC 8335: PROBE: A Utility for Probing Interfaces";
}

identity no-such-table-entry {
  base icmp-type;
  description
    "Identity for no such table entry
     in extended echo reply types";
  reference
```

```
    "RFC 792: Internet Control Message Protocol
    RFC 8335: PROBE: A Utility for Probing Interfaces";
}

identity multiple-interfaces-satisfy-query {
    base icmp-type;
    description
        "Identity for multiple interfaces satisfy query
        in extended echo reply types";
    reference
        "RFC 792: Internet Control Message Protocol
        RFC 8335: PROBE: A Utility for Probing Interfaces";
}

identity target-device {
    description
        "Base identity for target devices";
    reference
        "draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities";
}

identity pc {
    base target-device;
    description
        "Identity for pc";
}

identity mobile-phone {
    base target-device;
    description
        "Identity for mobile-phone";
}

identity voip-volte-phone {
    base target-device;
    description
        "Identity for voip-volte-phone";
}

identity tablet {
    base target-device;
    description
        "Identity for tablet";
}

identity iot {
    base target-device;
```

```
    description
      "Identity for IoT";
  }

  identity vehicle {
    base target-device;
    description
      "Identity for vehicle";
  }

  identity content-security-control {
    description
      "Base identity for content security control";
    reference
      "RFC 8329: Framework for Interface to
       Network Security Functions - Differences
       from ACL Data Models
       draft-ietf-i2nsf-capability-04: Information Model
       of NSFs Capabilities";
  }

  identity antivirus {
    base content-security-control;
    description
      "Identity for antivirus";
  }

  identity ips {
    base content-security-control;
    description
      "Identity for ips";
  }

  identity ids {
    base content-security-control;
    description
      "Identity for ids";
  }

  identity url-filtering {
    base content-security-control;
    description
      "Identity for url filtering";
  }

  identity mail-filtering {
    base content-security-control;
    description
```

```
    "Identity for mail filtering";
}

identity file-blocking {
    base content-security-control;
    description
        "Identity for file blocking";
}

identity file-isolate {
    base content-security-control;
    description
        "Identity for file isolate";
}

identity pkt-capture {
    base content-security-control;
    description
        "Identity for packet capture";
}

identity application-control {
    base content-security-control;
    description
        "Identity for application control";
}

identity voip-volte {
    base content-security-control;
    description
        "Identity for voip and volte";
}

identity attack-mitigation-control {
    description
        "Base identity for attack mitigation control";
    reference
        "RFC 8329: Framework for Interface to
        Network Security Functions - Differences
        from ACL Data Models
        draft-ietf-i2nsf-capability-04: Information Model
        of NSFs Capabilities";
}

identity syn-flood {
    base attack-mitigation-control;
    description
        "Identity for syn flood";
}
```



```
}

identity udp-flood {
  base attack-mitigation-control;
  description
    "Identity for udp flood";
}

identity icmp-flood {
  base attack-mitigation-control;
  description
    "Identity for icmp flood";
}

identity ip-frag-flood {
  base attack-mitigation-control;
  description
    "Identity for ip frag flood";
}

identity ipv6-related {
  base attack-mitigation-control;
  description
    "Identity for ipv6 related";
}

identity http-and-https-flood {
  base attack-mitigation-control;
  description
    "Identity for http and https flood";
}

identity dns-flood {
  base attack-mitigation-control;
  description
    "Identity for dns flood";
}

identity dns-amp-flood {
  base attack-mitigation-control;
  description
    "Identity for dns amp flood";
}

identity ssl-ddos {
  base attack-mitigation-control;
  description
    "Identity for ssl ddos";
}
```

```
}

identity ip-sweep {
  base attack-mitigation-control;
  description
    "Identity for ip sweep";
}

identity port-scanning {
  base attack-mitigation-control;
  description
    "Identity for port scanning";
}

identity ping-of-death {
  base attack-mitigation-control;
  description
    "Identity for ping of death";
}

identity teardrop {
  base attack-mitigation-control;
  description
    "Identity for teardrop";
}

identity oversized-icmp {
  base attack-mitigation-control;
  description
    "Identity for oversized icmp";
}

identity tracert {
  base attack-mitigation-control;
  description
    "Identity for tracert";
}

identity ingress-action {
  description
    "Base identity for action";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Ingress Action";
}

identity egress-action {
  description
```

```
    "Base identity for egress action";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Egress action";
}

identity default-action {
  description
    "Base identity for default action";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Default action";
}

identity pass {
  base ingress-action;
  base egress-action;
  base default-action;
  description
    "Identity for pass";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Actions and
      default action";
}

identity drop {
  base ingress-action;
  base egress-action;
  base default-action;
  description
    "Identity for drop";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Actions and
      default action";
}

identity reject {
  base ingress-action;
  base egress-action;
  base default-action;
  description
    "Identity for reject";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Actions and
      default action";
}
```

```
}

identity alert {
  base ingress-action;
  base egress-action;
  base default-action;
  description
    "Identity for alert";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Actions and
    default action";
}

identity mirror {
  base ingress-action;
  base egress-action;
  base default-action;
  description
    "Identity for mirror";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Actions and
    default action";
}

identity log-action {
  description
    "Base identity for log action";
}

identity rule-log {
  base log-action;
  description
    "Identity for rule log";
}

identity session-log {
  base log-action;
  description
    "Identity for session log";
}

identity invoke-signaling {
  base egress-action;
  description
    "Identity for invoke signaling";
}
```

```
identity tunnel-encapsulation {
  base egress-action;
  description
    "Identity for tunnel encapsulation";
}

identity forwarding {
  base egress-action;
  description
    "Identity for forwarding";
}

identity redirection {
  base egress-action;
  description
    "Identity for redirection";
}

identity resolution-strategy {
  description
    "Base identity for resolution strategy";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Resolution Strategy";
}

identity fmr {
  base resolution-strategy;
  description
    "Identity for First Matching Rule (FMR)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Resolution Strategy";
}

identity lmr {
  base resolution-strategy;
  description
    "Identity for Last Matching Rule (LMR)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Resolution Strategy";
}

identity pmr {
  base resolution-strategy;
  description
```

```
    "Identity for Prioritized Matching Rule (PMR)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Resolution Strategy";
}

identity pmre {
  base resolution-strategy;
  description
    "Identity for Prioritized Matching Rule
      with Errors (PMRE)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Resolution Strategy";
}

identity pmrn {
  base resolution-strategy;
  description
    "Identity for Prioritized Matching Rule
      with No Errors (PMRN)";
  reference
    "draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Resolution Strategy";
}

/*
 * Typedefs
 */

typedef start-time-type {
  type union {
    type string {
      pattern '\d{2}:\d{2}:\d{2}(\.\d+)?'
        + '(Z|[\+|-]\d{2}:\d{2})';
    }

    type enumeration {
      enum right-away {
        description
          "Immediate rule execution
            in the system.";
      }
    }
  }

  description
    "Start time when the rules are applied.";
}
```

```
}

typedef end-time-type {
  type union {
    type string {
      pattern '\d{2}:\d{2}:\d{2}(\.\d+)?'
        + '(Z|[\+\-]\d{2}:\d{2})';
    }

    type enumeration {
      enum infinitely {
        description
          "Infinite rule execution
           in the system.";
      }
    }
  }
  description
    "End time when the rules are applied.";
}

typedef day-type {
  type enumeration {
    enum sunday {
      description
        "Sunday for periodic day";
    }
    enum monday {
      description
        "Monday for periodic day";
    }
    enum tuesday {
      description
        "Tuesday for periodic day";
    }
    enum wednesday {
      description
        "Wednesday for periodic day";
    }
    enum thursday {
      description
        "Thursday for periodic day";
    }
    enum friday {
      description
        "Friday for periodic day";
    }
    enum saturday {
```

```
        description
            "Saturday for periodic day";
    }
}
description
    "This can be used for the rules to be applied
    according to periodic day";
}

typedef month-type {
    type enumeration {
        enum january {
            description
                "January for periodic month";
        }
        enum february {
            description
                "February for periodic month";
        }
        enum march {
            description
                "March for periodic month";
        }
        enum april {
            description
                "April for periodic month";
        }
        enum may {
            description
                "May for periodic month";
        }
        enum june {
            description
                "June for periodic month";
        }
        enum july {
            description
                "July for periodic month";
        }
        enum august {
            description
                "August for periodic month";
        }
        enum september {
            description
                "September for periodic month";
        }
        enum october {
```



```
        description
            "October for periodic month";
    }
    enum november {
        description
            "November for periodic month";
    }
    enum december {
        description
            "December for periodic month";
    }
}
description
    "This can be used for the rules to be applied
    according to periodic month";
}

/*
 * Groupings
 */

grouping ipv4 {
    list ipv4-address {
        key "ipv4";
        description
            "The list of IPv4 address.";

        leaf ipv4 {
            type inet:ipv4-address;
            description
                "The value of IPv4 address.";
        }
        choice subnet {
            description
                "The subnet can be specified as a prefix length or
                netmask.";
            leaf prefix-length {
                type uint8 {
                    range "0..32";
                }
                description
                    "The length of the subnet prefix.";
            }
            leaf netmask {
                type yang:dotted-quad;
                description
                    "The subnet specified as a netmask.";
            }
        }
    }
}
```

```
    }
  }
  description
    "Grouping for an IPv4 address";

  reference
    "RFC 791: Internet Protocol - IPv4 address
     RFC 8344: A YANG Data Model for IP Management";
}

grouping ipv6 {
  list ipv6-address {
    key "ipv6";
    description
      "The list of IPv6 address.";

    leaf ipv6 {
      type inet:ipv6-address;
      description
        "The value of IPv6 address.";
    }

    leaf prefix-length {
      type uint8 {
        range "0..128";
      }
      description
        "The length of the subnet prefix.";
    }
  }
  description
    "Grouping for an IPv6 address";

  reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
     Specification - IPv6 address
     RFC 8344: A YANG Data Model for IP Management";
}

grouping pkt-sec-ipv4 {
  choice match-type {
    description
      "There are two types to configure a security policy
       for IPv4 address, such as exact match and range match.";
    case exact-match {
      uses ipv4;
      description
        "Exact match for an IPv4 address.";
    }
  }
}
```

```
    }
    case range-match {
      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 IPv4 address.";
      }
    }
  }
  description
    "Grouping for an IPv4 address.";

  reference
    "RFC 791: Internet Protocol - IPv4 address";
}

grouping pkt-sec-ipv6 {
  choice match-type {
    description
      "There are two types to configure a security policy
      for IPv6 address, such as exact match and range match.";
    case exact-match {
      uses ipv6;
      description
        "Exact match for an IPv6 address.";
    }
    case range-match {
      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 IPv6 address.";
    }
}
}
description
    "Grouping for IPv6 address.";

reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - IPv6 address";
}

grouping pkt-sec-port-number {
    choice match-type {
        description
            "There are two types to configure a security policy
            for a port number, such as exact match and range match.";
        case exact-match {
            leaf-list port-num {
                type inet:port-number;
                description
                    "Exact match for a port number.";
            }
        }
        case range-match {
            list range-port-num {
                key "start-port-num end-port-num";
                leaf start-port-num {
                    type inet:port-number;
                    description
                        "Start port number for a range match.";
                }
                leaf end-port-num {
                    type inet:port-number;
                    description
                        "Start port number for a range match.";
                }
            }
            description
                "Range match for a port number.";
        }
    }
}
description
    "Grouping for port number.";
```

```
    reference
      "RFC 793: Transmission Control Protocol - Port number
       RFC 768: User Datagram Protocol - Port Number";
  }

/*
 * Data nodes
 */

container i2nsf-security-policy {
  description
    "Container for security policy
     including a set of security rules according to certain logic,
     i.e., their similarity or mutual relations, etc. The network
     security policy is able to apply over both the unidirectional
     and bidirectional traffic across the NSF.
     The I2NSF security policies use the Event-Condition-Action
     (ECA) policy model ";

  reference
    "RFC 8329: Framework for Interface to Network Security
     Functions - I2NSF Flow Security Policy Structure
     draft-ietf-i2nsf-capability-04: Information Model
     of NSFs Capabilities - Design Principles and ECA Policy Model
     Overview";

  list system-policy {
    key "system-policy-name";
    description
      "The system-policy represents there could be multiple system
       policies in one NSF, and each system policy is used by
       one virtual instance of the NSF/device.";

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

    leaf priority-usage {
      type identityref {
        base priority-usage-type;
      }
      default priority-by-order;
    }
  }
}
```

```
    description
      "Priority usage type for security policy rule:
       priority by order and priority by number";
  }

  leaf resolution-strategy {
    type identityref {
      base resolution-strategy;
    }
    default fmr;
    description
      "The resolution strategies can be used to
       specify how to resolve conflicts that occur between
       the actions of the same or different policy rules that
       are matched and contained in this particular NSF";

    reference
      "draft-ietf-i2nsf-capability-04: Information Model
       of NSFs Capabilities - Resolution strategy";
  }

  leaf default-action {
    type identityref {
      base default-action;
    }
    default alert;
    description
      "This default action can be used to specify a predefined
       action when no other alternative action was matched
       by the currently executing I2NSF Policy Rule. An analogy
       is the use of a default statement in a C switch statement.";

    reference
      "draft-ietf-i2nsf-capability-04: Information Model
       of NSFs Capabilities - Default action";
  }

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

    leaf rule-name {
      type string;
      mandatory true;
      description
        "The name of the rule.
```

```
        This must be unique.";
    }

    leaf rule-description {
        type string;
        description
            "This description gives more information about
            rules.";
    }

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

    leaf rule-enable {
        type boolean;
        description
            "True is enable.
            False is not enable.";
    }

    leaf session-aging-time {
        type uint16;
        description
            "This is session aging time.";
    }

    container long-connection {
        description
            "This is long-connection";

        leaf enable {
            type boolean;
            description
                "True is enable.
                False is not enable.";
        }

        leaf during {
            type uint16;
            description
                "This is during time.";
        }
    }
}
```

```
}

container time-zone {
  description
    "Time zone when the rules are applied";
  container absolute-time-zone {
    description
      "Rule execution according to absolute time";

    leaf start-time {
      type start-time-type;
      default right-away;
      description
        "Start time when the rules are applied";
    }
    leaf end-time {
      type end-time-type;
      default infinitely;
      description
        "End time when the rules are applied";
    }
  }
}

container periodic-time-zone {
  description
    "Rule execution according to periodic time";

  container day {
    description
      "Rule execution according to day.";
    leaf every-day {
      type boolean;
      default true;
      description
        "Rule execution every day";
    }

    leaf-list specific-day {
      when "../every-day = 'false'";
      type day-type;
      description
        "Rule execution according
        to specific day";
    }
  }
}

container month {
```



```
        description
            "Rule execution according to month.";
        leaf every-month {
            type boolean;
            default true;
            description
                "Rule execution every day";
        }

        leaf-list specific-month {
            when "../every-month = 'false'";
            type month-type;
            description
                "Rule execution according
                 to month day";
        }
    }
}

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

    reference
        "RFC 8329: Framework for Interface to Network Security
         Functions - I2NSF Flow Security Policy Structure
         draft-ietf-i2nsf-capability-04: Information Model
         of NSFs Capabilities - Design Principles and ECA
         Policy Model Overview
         draft-hong-i2nsf-nsf-monitoring-data-model-06: A YANG
         Data Model for Monitoring I2NSF Network Security
         Functions - System Alarm and System Events";

    leaf event-clause-description {
        type string;
        description
            "Description for an event clause";
    }
}
```

```
container event-clauses {
  description
    "It has two event types such as
    system event and system alarm.";
  reference
    "RFC 8329: Framework for Interface to Network Security
    Functions - I2NSF Flow Security Policy Structure
    draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Design Principles and ECA Policy
    Model Overview
    draft-hong-i2nsf-nsf-monitoring-data-model-06: A YANG
    Data Model for Monitoring I2NSF Network Security
    Functions - System Alarm and System Events";

  leaf-list system-event {
    type identityref {
      base system-event;
    }
    description
      "The security policy rule according to
      system events.";
  }

  leaf-list system-alarm {
    type identityref {
      base system-alarm;
    }
    description
      "The security policy rule according to
      system alarms.";
  }
}

container condition-clause-container {
  description
    "A condition is defined as a set
    of attributes, features, and/or values that are to be
    compared with a set of known attributes, features,
    and/or values in order to determine whether or not the
    set of Actions in that (imperative) I2NSF Policy Rule
    can be executed or not. Examples of I2NSF Conditions
    include matching attributes of a packet or flow, and
    comparing the internal state of an NSF to a desired
    state.";
  reference
    "RFC 8329: Framework for Interface to Network Security
    Functions - I2NSF Flow Security Policy Structure
```

```
draft-ietf-i2nsf-capability-04: Information Model
of NSFs Capabilities - Design Principles and ECA Policy
Model Overview";

leaf condition-clause-description {
  type string;
  description
    "Description for a condition clause.";
}

container packet-security-ipv4-condition {
  description
    "The purpose of this container is to represent IPv4
    packet header information to determine if the set
    of policy actions in this ECA policy rule should be
    executed or not.";
  reference
    "RFC 791: Internet Protocol";

  leaf ipv4-description {
    type string;
    description
      "This is description for ipv4 condition.";
  }

  container pkt-sec-ipv4-header-length {
    choice match-type {
      description
        "There are two types to configure a security
        policy for IPv4 header length, such as exact match
        and range match.";
      case exact-match {
        leaf-list ipv4-header-length {
          type uint8 {
            range "5..15";
          }
          description
            "Exact match for an IPv4 header length.";
        }
      }
      case range-match {
        list range-ipv4-header-length {
          key "start-ipv4-header-length
            end-ipv4-header-length";
          leaf start-ipv4-header-length {
            type uint8 {
              range "5..15";
            }
          }
        }
      }
    }
  }
}
```

```
        description
            "Start IPv4 header length for a range match.";
    }

    leaf end-ipv4-header-length {
        type uint8 {
            range "5..15";
        }
        description
            "End IPv4 header length for a range match.";
    }
    description
        "Range match for an IPv4 header length.";
}

}
}
description
    "The security policy rule according to
    IPv4 header length.";
reference
    "RFC 791: Internet Protocol - Header length";
}

leaf-list pkt-sec-ipv4-tos {
    type identityref {
        base type-of-service;
    }
    description
        "The security policy rule according to
        IPv4 type of service.";
    reference
        "RFC 791: Internet Protocol - Type of service";
}

container pkt-sec-ipv4-total-length {
    choice match-type {
        description
            "There are two types to configure a security
            policy for IPv4 total length, such as exact match
            and range match.";
        case exact-match {
            leaf-list ipv4-total-length {
                type uint16;
                description
                    "Exact match for an IPv4 total length.";
            }
        }
        case range-match {
```

```
    list range-ipv4-total-length {
      key "start-ipv4-total-length end-ipv4-total-length";
      leaf start-ipv4-total-length {
        type uint16;
        description
          "Start IPv4 total length for a range match.";
      }
      leaf end-ipv4-total-length {
        type uint16;
        description
          "End IPv4 total length for a range match.";
      }
      description
        "Range match for an IPv4 total length.";
    }
  }
  description
    "The security policy rule according to
    IPv4 total length.";
  reference
    "RFC 791: Internet Protocol - Total length";
}

leaf-list pkt-sec-ipv4-id {
  type uint16;
  description
    "The security policy rule according to
    IPv4 identification.";
  reference
    "RFC 791: Internet Protocol - Identification";
}

leaf-list pkt-sec-ipv4-fragment-flags {
  type identityref {
    base fragmentation-flags-type;
  }
  description
    "The security policy rule according to
    IPv4 fragment flags.";
  reference
    "RFC 791: Internet Protocol - Fragment flags";
}

container pkt-sec-ipv4-fragment-offset {
  choice match-type {
    description
      "There are two types to configure a security
```

```
    policy for IPv4 fragment offset, such as exact match
    and range match.";
  case exact-match {
    leaf-list ipv4-fragment-offset {
      type uint16 {
        range "0..16383";
      }
      description
        "Exact match for an IPv4 fragment offset.";
    }
  }
  case range-match {
    list range-ipv4-fragment-offset {
      key "start-ipv4-fragment-offset
        end-ipv4-fragment-offset";
      leaf start-ipv4-fragment-offset {
        type uint16 {
          range "0..16383";
        }
        description
          "Start IPv4 fragment offset for a range match.";
      }
      leaf end-ipv4-fragment-offset {
        type uint16 {
          range "0..16383";
        }
        description
          "End IPv4 fragment offset for a range match.";
      }
      description
        "Range match for an IPv4 fragment offset.";
    }
  }
}
description
  "The security policy rule according to
  IPv4 fragment offset.";
reference
  "RFC 791: Internet Protocol - Fragment offset";
}

container pkt-sec-ipv4-ttl {
  choice match-type {
    description
      "There are two types to configure a security
      policy for IPv4 TTL, such as exact match
      and range match.";
    case exact-match {
```

```
    leaf-list ipv4-ttl {
      type uint8;
      description
        "Exact match for an IPv4 TTL.";
    }
  }
  case range-match {
    list range-ipv4-ttl {
      key "start-ipv4-ttl end-ipv4-ttl";
      leaf start-ipv4-ttl {
        type uint8;
        description
          "Start IPv4 TTL for a range match.";
      }
      leaf end-ipv4-ttl {
        type uint8;
        description
          "End IPv4 TTL for a range match.";
      }
    }
    description
      "Range match for an IPv4 TTL.";
  }
}
description
  "The security policy rule according to
  IPv4 time-to-live (TTL).";
reference
  "RFC 791: Internet Protocol - Time to live";
}

leaf-list pkt-sec-ipv4-protocol {
  type identityref {
    base protocol;
  }
  description
    "The security policy rule according to
    IPv4 protocol.";
  reference
    "RFC 791: Internet Protocol - Protocol";
}

container pkt-sec-ipv4-src {
  uses pkt-sec-ipv4;
  description
    "The security policy rule according to
    IPv4 source address.";
```

```
    reference
      "RFC 791: Internet Protocol - IPv4 Address";
  }

  container pkt-sec-ipv4-dest {
    uses pkt-sec-ipv4;
    description
      "The security policy rule according to
       IPv4 destination address.";
    reference
      "RFC 791: Internet Protocol - IPv4 Address";
  }

  leaf-list pkt-sec-ipv4-ipopts {
    type identityref {
      base ipopts;
    }
    description
      "The security policy rule according to
       IPv4 options.";
    reference
      "RFC 791: Internet Protocol - Options";
  }

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

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

container packet-security-ipv6-condition {
  description
    "The purpose of this container is to represent
     IPv6 packet header information to determine
```



```
        if the set of policy actions in this ECA policy
        rule should be executed or not.";
reference
  "RFC 2460: Internet Protocol, Version 6 (IPv6)
  Specification";

leaf ipv6-description {
  type string;
  description
    "This is description for ipv6 condition.";
}

leaf-list pkt-sec-ipv6-traffic-class {
  type identityref {
    base traffic-class;
  }
  description
    "The security policy rule according to
    IPv6 traffic class.";
reference
  "RFC 2460: Internet Protocol, Version 6 (IPv6)
  Specification - Traffic class";
}

container pkt-sec-ipv6-flow-label {
  choice match-type {
    description
      "There are two types to configure a security
      policy for IPv6 flow label, such as exact match
      and range match.";
    case exact-match {
      leaf-list ipv6-flow-label {
        type uint32 {
          range "0..1048575";
        }
        description
          "Exact match for an IPv6 flow label.";
      }
    }
    case range-match {
      list range-ipv6-flow-label {
        key "start-ipv6-flow-label end-ipv6-flow-label";
        leaf start-ipv6-flow-label {
          type uint32 {
            range "0..1048575";
          }
          description

```

```
        "Start IPv6 flow label for a range match.";
    }
    leaf end-ipv6-flow-label {
        type uint32 {
            range "0..1048575";
        }
        description
            "End IPv6 flow label for a range match.";
    }
    description
        "Range match for an IPv6 flow label.";
}
}
description
    "The security policy rule according to
    IPv6 flow label.";
reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Flow label";
}

container pkt-sec-ipv6-payload-length {
    choice match-type {
        description
            "There are two types to configure a security
            policy for IPv6 payload length, such as
            exact match and range match.";
        case exact-match {
            leaf-list ipv6-payload-length {
                type uint16;
                description
                    "Exact match for an IPv6 payload length.";
            }
        }
        case range-match {
            list range-ipv6-payload-length {
                key "start-ipv6-payload-length
                    end-ipv6-payload-length";
                leaf start-ipv6-payload-length {
                    type uint16;
                    description
                        "Start IPv6 payload length for a range match.";
                }
            }
            leaf end-ipv6-payload-length {
                type uint16;
                description
                    "End IPv6 payload length for a range match.";
            }
        }
    }
}
```

```
        }
        description
            "Range match for an IPv6 payload length.";
    }
}
description
    "The security policy rule according to
    IPv6 payload length.";
reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Payload length";
}

leaf-list pkt-sec-ipv6-next-header {
    type identityref {
        base next-header;
    }
    description
        "The security policy rule according to
        IPv6 next header.";
    reference
        "RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - Next header";
}

container pkt-sec-ipv6-hop-limit {
    choice match-type {
        description
            "There are two types to configure a security
            policy for IPv6 hop limit, such as exact match
            and range match.";
        case exact-match {
            leaf-list ipv6-hop-limit {
                type uint8;
                description
                    "Exact match for an IPv6 hop limit.";
            }
        }
        case range-match {
            list range-ipv6-hop-limit {
                key "start-ipv6-hop-limit end-ipv6-hop-limit";
                leaf start-ipv6-hop-limit {
                    type uint8;
                    description
                        "Start IPv6 hop limit for a range match.";
                }
                leaf end-ipv6-hop-limit {
```

```
        type uint8;
        description
            "End IPv6 hop limit for a range match.";
    }
    description
        "Range match for an IPv6 hop limit.";
}
}
}
description
    "The security policy rule according to
    IPv6 hop limit.";
reference
    "RFC 2460: Internet Protocol, Version 6 (IPv6)
    Specification - Hop limit";
}

container pkt-sec-ipv6-src {
    uses pkt-sec-ipv6;
    description
        "The security policy rule according to
        IPv6 source address.";
    reference
        "RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - IPv6 address";
}

container pkt-sec-ipv6-dest {
    uses pkt-sec-ipv6;
    description
        "The security policy rule according to
        IPv6 destination address.";
    reference
        "RFC 2460: Internet Protocol, Version 6 (IPv6)
        Specification - IPv6 address";
}

}

container packet-security-tcp-condition {
    description
        "The purpose of this container is to represent
        TCP packet header information to determine
        if the set of policy actions in this ECA policy
        rule should be executed or not.";
    reference
        "RFC 793: Transmission Control Protocol";
}
```

```
leaf tcp-description {
    type string;
    description
        "This is description for tcp condition.";
}

container pkt-sec-tcp-src-port-num {
    uses pkt-sec-port-number;
    description
        "The security policy rule according to
        tcp source port number.";
    reference
        "RFC 793: Transmission Control Protocol
        - Port number";
}

container pkt-sec-tcp-dest-port-num {
    uses pkt-sec-port-number;
    description
        "The security policy rule according to
        tcp destination port number.";
    reference
        "RFC 793: Transmission Control Protocol
        - Port number";
}

container pkt-sec-tcp-seq-num {
    choice match-type {
        description
            "There are two types to configure a security
            policy for tcp sequence number,
            such as exact match and range match.";
        case exact-match {
            leaf-list tcp-seq-num {
                type uint32;
                description
                    "Exact match for an tcp sequence number.";
            }
        }
        case range-match {
            list range-tcp-seq-num {
                key "start-tcp-seq-num end-tcp-seq-num";
                leaf start-tcp-seq-num {
                    type uint32;
                    description
                        "Start tcp sequence number for a range match.";
                }
            }
        }
    }
}
```

```
        leaf end-tcp-seq-num {
            type uint32;
            description
                "End tcp sequence number for a range match.";
        }
        description
            "Range match for a tcp sequence number.";
    }
}
description
    "The security policy rule according to
    tcp sequence number.";
reference
    "RFC 793: Transmission Control Protocol
    - Sequence number";
}

container pkt-sec-tcp-ack-num {
    choice match-type {
        description
            "There are two types to configure a security
            policy for tcp acknowledgement number,
            such as exact match and range match.";
        case exact-match {
            leaf-list tcp-ack-num {
                type uint32;
                description
                    "Exact match for an tcp acknowledgement number.";
            }
        }
        case range-match {
            list range-tcp-ack-num {
                key "start-tcp-ack-num end-tcp-ack-num";
                leaf start-tcp-ack-num {
                    type uint32;
                    description
                        "Start tcp acknowledgement number
                        for a range match.";
                }
                leaf end-tcp-ack-num {
                    type uint32;
                    description
                        "End tcp acknowledgement number
                        for a range match.";
                }
            }
            description
                "Range match for a tcp acknowledgement number.";
        }
    }
}
```

```
    }
  }
}
description
  "The security policy rule according to
  tcp acknowledgement number.";
reference
  "RFC 793: Transmission Control Protocol
  - Acknowledgement number";
}

container pkt-sec-tcp-window-size {
  choice match-type {
    description
      "There are two types to configure a security
      policy for tcp window size,
      such as exact match and range match.";
    case exact-match {
      leaf-list tcp-window-size {
        type uint16;
        description
          "Exact match for an tcp window size.";
      }
    }
    case range-match {
      list range-tcp-window-size {
        key "start-tcp-window-size end-tcp-window-size";
        leaf start-tcp-window-size {
          type uint16;
          description
            "Start tcp window size for a range match.";
        }
        leaf end-tcp-window-size {
          type uint16;
          description
            "End tcp window size for a range match.";
        }
      }
      description
        "Range match for a tcp window size.";
    }
  }
}
description
  "The security policy rule according to
  tcp window size.";
reference
  "RFC 793: Transmission Control Protocol
  - Window size";
```

```
    }

    leaf-list pkt-sec-tcp-flags {
      type identityref {
        base tcp-flags;
      }
      description
        "The security policy rule according to
        tcp flags.";
      reference
        "RFC 793: Transmission Control Protocol
        - Flags";
    }
  }

  container packet-security-udp-condition {
    description
      "The purpose of this container is to represent
      UDP packet header information to determine
      if the set of policy actions in this ECA policy
      rule should be executed or not.";
    reference
      "RFC 793: Transmission Control Protocol";

    leaf udp-description {
      type string;
      description
        "This is description for udp condition.";
    }

    container pkt-sec-udp-src-port-num {
      uses pkt-sec-port-number;
      description
        "The security policy rule according to
        udp source port number.";
      reference
        "RFC 793: Transmission Control Protocol
        - Port number";
    }

    container pkt-sec-udp-dest-port-num {
      uses pkt-sec-port-number;
      description
        "The security policy rule according to
        udp destination port number.";
      reference
```



```
        "RFC 768: User Datagram Protocol
        - Total Length";
    }

    container pkt-sec-udp-total-length {
        choice match-type {
            description
                "There are two types to configure a security
                policy for udp sequence number,
                such as exact match and range match.";
            case exact-match {
                leaf-list udp-total-length {
                    type uint32;
                    description
                        "Exact match for an udp-total-length.";
                }
            }
            case range-match {
                list range-udp-total-length {
                    key "start-udp-total-length end-udp-total-length";
                    leaf start-udp-total-length {
                        type uint32;
                        description
                            "Start udp total length for a range match.";
                    }
                    leaf end-udp-total-length {
                        type uint32;
                        description
                            "End udp total length for a range match.";
                    }
                }
                description
                    "Range match for a udp total length.";
            }
        }
    }
    description
        "The security policy rule according to
        udp total length.";
    reference
        "RFC 768: User Datagram Protocol
        - Total Length";
}

container packet-security-icmp-condition {
    description
        "The purpose of this container is to represent
```

```
        ICMP packet header information to determine
        if the set of policy actions in this ECA policy
        rule should be executed or not.";
reference
  "RFC 792: Internet Control Message Protocol
  RFC 8335: PROBE: A Utility for Probing Interfaces";

leaf icmp-description {
  type string;
  description
    "This is description for icmp condition.";
}

leaf-list pkt-sec-icmp-type-and-code {
  type identityref {
    base icmp-type;
  }
  description
    "The security policy rule according to
    ICMP parameters.";
reference
  "RFC 792: Internet Control Message Protocol
  RFC 8335: PROBE: A Utility for Probing Interfaces";
}

container packet-security-http-condition {
  description
    "Condition for http.";

  leaf http-description {
    type string;
    description
      "This is description for http condition.";
  }

  leaf-list pkt-sec-uri-content {
    type string;
    description
      "The security policy rule according to
      uri content.";
  }

  leaf-list pkt-sec-url-content {
    type string;
    description
      "The security policy rule according to
      url content.";
```

```
    }
  }

  container packet-security-voice-condition {
    description
      "For the VoIP/VoLTE security system, a VoIP/
      VoLTE security system can monitor each
      VoIP/VoLTE flow and manage VoIP/VoLTE
      security rules controlled by a centralized
      server for VoIP/VoLTE security service
      (called VoIP IPS). The VoIP/VoLTE security
      system controls each switch for the
      VoIP/VoLTE call flow management by
      manipulating the rules that can be added,
      deleted, or modified dynamically.";
    reference
      "RFC 3261: SIP: Session Initiation Protocol";

    leaf voice-description {
      type string;
      description
        "This is description for voice condition.";
    }

    leaf-list pkt-sec-src-voice-id {
      type string;
      description
        "The security policy rule according to
        a source voice ID for VoIP and VoLTE.";
    }

    leaf-list pkt-sec-dest-voice-id {
      type string;
      description
        "The security policy rule according to
        a destination voice ID for VoIP and VoLTE.";
    }

    leaf-list pkt-sec-user-agent {
      type string;
      description
        "The security policy rule according to
        an user agent for VoIP and VoLTE.";
    }
  }

  container packet-security-ddos-condition {
    description
```

```
        "Condition for DDoS attack.";

    leaf ddos-description {
        type string;
        description
            "This is description for ddos condition.";
    }

    leaf pkt-sec-alert-rate {
        type uint32;
        description
            "The alert rate of flood detect for
             same packets.";
    }
}

container packet-payload-condition {
    description
        "Condition for packet payload";
    leaf packet-payload-description {
        type string;
        description
            "This is description for payload condition.
             Vendors can write instructions for payload condition
             that vendor made";
    }
    leaf-list pkt-payload-content {
        type string;
        description
            "The content keyword is very important in
             signatures. Between the quotation marks you
             can write on what you would like the
             signature to match.";
    }
}

leaf-list acl-number {
    type uint32;
    description
        "This is acl-number.";
}

container application-condition {
    description
        "Condition for application";
    leaf application-description {
        type string;
        description

```

```
        "This is description for application condition.";
    }
    leaf-list application-object {
        type string;
        description
            "This is application object.";
    }
    leaf-list application-group {
        type string;
        description
            "This is application group.";
    }
    leaf-list application-label {
        type string;
        description
            "This is application label.";
    }
    container category {
        description
            "This is application category";
        list application-category {
            key "name application-subcategory";
            description
                "This is application category list";
            leaf name {
                type string;
                description
                    "This is name for application category.";
            }
            leaf application-subcategory {
                type string;
                description
                    "This is application subcategory.";
            }
        }
    }
}

container target-condition {
    description
        "Condition for target";
    leaf target-description {
        type string;
        description
            "This is description for target condition.
            Vendors can write instructions for target condition
            that vendor made";
    }
}
```

```
container device-sec-context-cond {
  description
    "The device attribute that can identify a device,
    including the device type (i.e., router, switch,
    pc, ios, or android) and the device's owner as
    well.";

  leaf-list target-device {
    type identityref {
      base target-device;
    }
    description
      "Leaf list for target devices";
  }
}

container users-condition {
  description
    "Condition for users";
  leaf users-description {
    type string;
    description
      "This is description for user condition.
      Vendors can write instructions for user condition
      that vendor made";
  }
}

container user{
  description
    "The user (or user group) information with which
    network flow is associated: The user has many
    attributes such as name, id, password, type,
    authentication mode and so on. Name/id is often
    used in the security policy to identify the user.
    Besides, NSF is aware of the IP address of the
    user provided by a unified user management system
    via network. Based on name-address association,
    NSF is able to enforce the security functions
    over the given user (or user group)";

  choice user-name {
    description
      "The name of the user.
      This must be unique.";

    case tenant {
      description
        "Tenant information.";
    }
  }
}
```

```
        leaf tenant {
            type uint8;
            mandatory true;
            description
                "User's tenant information.";
        }
    }

    case vn-id {
        description
            "VN-ID information.";

        leaf vn-id {
            type uint8;
            mandatory true;
            description
                "User's VN-ID information.";
        }
    }
}

container group {
    description
        "The user (or user group) information with which
        network flow is associated: The user has many
        attributes such as name, id, password, type,
        authentication mode and so on. Name/id is often
        used in the security policy to identify the user.
        Besides, NSF is aware of the IP address of the
        user provided by a unified user management system
        via network. Based on name-address association,
        NSF is able to enforce the security functions
        over the given user (or user group)";

    choice group-name {
        description
            "The name of the user.
            This must be unique.";

        case tenant {
            description
                "Tenant information.";

            leaf tenant {
                type uint8;
                mandatory true;
                description
                    "User's tenant information.";
            }
        }
    }
}
```

```
    }
  }

  case vn-id {
    description
      "VN-ID information.";

    leaf vn-id {
      type uint8;
      mandatory true;
      description
        "User's VN-ID information.";
    }
  }
}

leaf security-grup {
  type string;
  mandatory true;
  description
    "security-grup.";
}

container url-category-condition {
  description
    "Condition for url category";
  leaf url-category-description {
    type string;
    description
      "This is description for url category condition.
      Vendors can write instructions for context condition
      that vendor made";
  }

  leaf-list pre-defined-category {
    type string;
    description
      "This is pre-defined-category.";
  }
  leaf-list user-defined-category {
    type string;
    description
      "This user-defined-category.";
  }
}

container context-condition {
```



```
    description
      "Condition for context";
    leaf context-description {
      type string;
      description
        "This is description for context condition.
        Vendors can write instructions for context condition
        that vendor made";
    }
  }

  container gen-context-condition {
    description
      "Condition for generic context";
    leaf gen-context-description {
      type string;
      description
        "This is description for generic context condition.
        Vendors can write instructions for generic context
        condition that vendor made";
    }
  }

  container geographic-location {
    description
      "The location where network traffic is associated
      with. The region can be the geographic location
      such as country, province, and city,
      as well as the logical network location such as
      IP address, network section, and network domain.";

    leaf-list src-geographic-location {
      type uint32;
      description
        "This is mapped to ip address. We can acquire
        source region through ip address stored in the
        database.";
    }
    leaf-list dest-geographic-location {
      type uint32;
      description
        "This is mapped to ip address. We can acquire
        destination region through ip address stored
        in the database.";
    }
  }
}
```

```
container action-clause-container {
  description
    "An action is used to control and monitor aspects of
    flow-based NSFs when the event and condition clauses
    are satisfied. NSFs provide security functions by
    executing various Actions. Examples of I2NSF Actions
    include providing intrusion detection and/or protection,
    web and flow filtering, and deep packet inspection
    for packets and flows.";
  reference
    "RFC 8329: Framework for Interface to Network Security
    Functions - I2NSF Flow Security Policy Structure
    draft-ietf-i2nsf-capability-04: Information Model
    of NSFs Capabilities - Design Principles and ECA Policy
    Model Overview";

  leaf action-clause-description {
    type string;
    description
      "Description for an action clause.";
  }

  container packet-action {
    description
      "Action for packets";
    reference
      "RFC 8329: Framework for Interface to Network Security
      Functions - I2NSF Flow Security Policy Structure
      draft-ietf-i2nsf-capability-04: Information Model
      of NSFs Capabilities - Design Principles and ECA
      Policy Model Overview";

    leaf ingress-action {
      type identityref {
        base ingress-action;
      }
      description
        "Action: pass, drop, reject, alert, and mirror.";
    }

    leaf egress-action {
      type identityref {
        base egress-action;
      }
      description
        "Egress action: pass, drop, reject, alert, mirror,
        invoke-signaling, tunnel-encapsulation,
        forwarding, and redirection.";
    }
  }
}
```

```
    }

    leaf log-action {
      type identityref {
        base log-action;
      }
      description
        "Log action: rule log and session log";
    }
  }

  container advanced-action {
    description
      "If the packet need be additionally inspected,
       the packet are passed to advanced network
       security functions according to the profile.";
    reference
      "RFC 8329: Framework for Interface to Network Security
       Functions - Differences from ACLData Models";

    leaf-list content-security-control {
      type identityref {
        base content-security-control;
      }
      description
        "The Profile is divided into content security
         control and attack-mitigation-control.
         Content security control: antivirus, ips, ids,
         url filtering, mail filtering, file blocking,
         file isolate, packet capture, application control,
         voip and volte.";
    }

    leaf-list attack-mitigation-control {
      type identityref {
        base attack-mitigation-control;
      }
      description
        "The Profile is divided into content security
         control and attack-mitigation-control.
         Attack mitigation control: syn flood, udp flood,
         icmp flood, ip frag flood, ipv6 related, http flood,
         https flood, dns flood, dns amp flood, ssl ddos,
         ip sweep, port scanning, ping of death, teardrop,
         oversized icmp, tracert.";
    }
  }
}
```

```
    }
  }
  container rule-group {
    description
      "This is rule group";

    list groups {
      key "group-name";
      description
        "This is a group for rules";

      leaf group-name {
        type string;
        description
          "This is a group for rules";
      }

      container rule-range {
        description
          "This is a rule range.";

        leaf start-rule {
          type string;
          description
            "This is a start rule";
        }
        leaf end-rule {
          type string;
          description
            "This is a end rule";
        }
      }
    }
    leaf enable {
      type boolean;
      description
        "This is enable
        False is not enable.";
    }
    leaf description {
      type string;
      description
        "This is a desription for rule-group";
    }
  }
}
}
```

<CODE ENDS>

Figure 5: YANG Data Module of I2NSF NSF-Facing-Interface

6. 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-policy-rule-for-nsf

Registrant Contact: The IESG.

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-policy-rule-for-nsf

namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf

prefix: iiprfn

reference: RFC XXXX

7. Security Considerations

The YANG module specified in this document defines a data schema designed to be accessed through network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the required transport secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the required transport secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides a means of restricting access to specific NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [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>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [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>>.

- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8431] Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for the Routing Information Base (RIB)", RFC 8431, DOI 10.17487/RFC8431, September 2018, <<https://www.rfc-editor.org/info/rfc8431>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

8.2. Informative References

- [i2nsf-advanced-nsf-dm]
Pan, W. and L. Xia, "Configuration of Advanced Security Functions with I2NSF Security Controller", draft-dong-i2nsf-asf-config-01 (work in progress), October 2018.
- [i2nsf-nsf-cap-dm]
Hares, S., Jeong, J., Kim, J., Moskowitz, R., and Q. Lin, "I2NSF Capability YANG Data Model", draft-ietf-i2nsf-capability-data-model-03 (work in progress), March 2019.
- [i2nsf-nsf-cap-im]
Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSF's Capabilities", draft-ietf-i2nsf-capability-04 (work in progress), October 2018.
- [supa-policy-info-model]
Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-03 (work in progress), May 2017.

Appendix A. Configuration Examples

This section shows configuration examples of "ietf-i2nsf-policy-rule-for-nsf" module for security policy rules of network security devices. For security requirements, we assume that the NSFs (i.e., General firewall, Time based firewall, URL filter, VoIP/VoLTE filter, and http and https flood mitigation) described in Appendix A. Configuration Examples of [i2nsf-nsf-cap-dm] are registered in I2NSF framework. With the registered NSFs, we show configuration examples for security policy rules of network security functions according to the following three security requirements: (i) Block SNS access during business hours, (ii) Block malicious VoIP/VoLTE packets coming to the company, and (iii) Mitigate http and https flood attacks on company web server.

A.1. Security Requirement 1: Block SNS Access during Business Hours

This section shows a configuration example for blocking SNS access during business hours.


```
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
<system-policy>
  <system-policy-name>sns_access</system-policy-name>
  <rules>
    <rule-name>block_sns_access_during_operation_time</rule-name>
    <time-zone>
      <absolute-time-zone>
        <start-time>09:00:00Z</start-time>
        <end-time>18:00:00Z</end-time>
      </absolute-time-zone>
    </time-zone>
    <condition-clause-container>
      <packet-security-ipv4-condition>
        <pkt-sec-ipv4-src>
          <range-ipv4-address>
            <start-ipv4-address>221.159.112.1</start-ipv4-address>
            <end-ipv4-address>221.159.112.90</end-ipv4-address>
          </range-ipv4-address>
        </pkt-sec-ipv4-src>
      </packet-security-ipv4-condition>
    </condition-clause-container>
    <action-clause-container>
      <advanced-action>
        <content-security-control>url-filtering</content-security-control>
      </advanced-action>
    </action-clause-container>
  </rules>
</system-policy>
</i2nsf-security-policy>
```

Figure 6: Configuration XML for Time based Firewall to Block SNS
Access during Business Hours

```
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
<system-policy>
  <system-policy-name>sns_access</system-policy-name>
  <rules>
    <rule-name>block_sns_access_during_operation_time</rule-name>
    <condition-clause-container>
      <packet-security-http-condition>
        <pkt-sec-url-content>facebook</pkt-sec-url-content>
        <pkt-sec-url-content>instagram</pkt-sec-url-content>
      </packet-security-http-condition>
    </condition-clause-container>
    <action-clause-container>
      <packet-action>
        <egress-action>drop</egress-action>
      </packet-action>
    </action-clause-container>
  </rules>
</system-policy>
</i2nsf-security-policy>
```

Figure 7: Configuration XML for Web Filter to Block SNS Access during Business Hours

Figure 6 and Figure 7 show the configuration XML documents for time based firewall and web filter to block SNS access during business hours. For the security requirement, two NSFs (i.e., a time based firewall and a web filter) were used because one NSF can not meet the security requirement. The instances of XML documents for the time based firewall and the web filter are as follows: Note that a detailed data model for the configuration of the advanced network security function (i.e., web filter) is described in [i2nsf-advanced-nsf-dm].

Time based Firewall

1. The name of the system policy is sns_access.
2. The name of the rule is block_sns_access_during_operation_time.
3. The rule is operated during the business hours (i.e., from 9 a.m. to 6 p.m.).
4. The rule inspects a source IPv4 address (i.e., from 221.159.112.1 to 221.159.112.90) to inspect the outgoing packets of employees.

5. If the outgoing packets match the rules above, the time based firewall sends the packets to url filtering for additional inspection because the time based firewall can not inspect contents of the packets for the SNS URL.

Web Filter

1. The name of the system policy is sns_access.
 2. The name of the rule is block_facebook_and_instagram.
 3. The rule inspects URL address to block the access packets to the facebook or the instagram.
 4. If the outgoing packets match the rules above, the packets are blocked.
- A.2. Security Requirement 2: Block Malicious VoIP/VoLTE Packets Coming to the Company

This section shows a configuration example for blocking malicious VoIP/VoLTE packets coming to the company.

```
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
<system-policy>
  <system-policy-name>voip_volte_inspection</system-policy-name>
  <rules>
    <rule-name>block_malicious_voice_id</rule-name>
    <condition-clause-container>
      <packet-security-ipv4-condition>
        <pkt-sec-ipv4-dest>
          <range-ipv4-address>
            <start-ipv4-address>221.159.112.1</start-ipv4-address>
            <end-ipv4-address>221.159.112.90</end-ipv4-address>
          </range-ipv4-address>
        </pkt-sec-ipv4-dest>
      </packet-security-ipv4-condition>
      <packet-security-tcp-condition>
        <pkt-sec-tcp-dest-port-num>
          <port-num>5060</port-num>
          <port-num>5061</port-num>
        </pkt-sec-tcp-dest-port-num>
      </packet-security-tcp-condition>
    </condition-clause-container>
    <action-clause-container>
      <advanced-action>
        <content-security-control>voip-volte</content-security-control>
      </advanced-action>
    </action-clause-container>
  </rules>
</system-policy>
</i2nsf-security-policy>
```

Figure 8: Configuration XML for General Firewall to Block Malicious VoIP/VoLTE Packets Coming to the Company

```
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
<system-policy>
  <system-policy-name>voip_volte_inspection</system-policy-name>
  <rules>
    <rule-name>block_malicious_voice_id</rule-name>
    <condition-clause-container>
      <packet-security-voice-condition>
        <pkt-sec-src-voice-id>11111@voip.black.com</pkt-sec-src-voice-id>
        <pkt-sec-src-voice-id>22222@voip.black.com</pkt-sec-src-voice-id>
      </packet-security-voice-condition>
    </condition-clause-container>
    <action-clause-container>
      <packet-action>
        <ingress-action>drop</ingress-action>
      </packet-action>
    </action-clause-container>
  </rules>
</system-policy>
</i2nsf-security-policy>
```

Figure 9: Configuration XML for VoIP/VoLTE Filter to Block Malicious VoIP/VoLTE Packets Coming to the Company

Figure 8 and Figure 9 show the configuration XML documents for general firewall and VoIP/VoLTE filter to block malicious VoIP/VoLTE packets coming to the company. For the security requirement, two NSFs (i.e., a general firewall and a VoIP/VoLTE filter) were used because one NSF can not meet the security requirement. The instances of XML documents for the general firewall and the VoIP/VoLTE filter are as follows: Note that a detailed data model for the configuration of the advanced network security function (i.e., VoIP/VoLTE filter) is described in [i2nsf-advanced-nsf-dm].

General Firewall

1. The name of the system policy is voip_volte_inspection.
2. The name of the rule is block_malicious_voip_volte_packets.
3. The rule inspects a destination IPv4 address (i.e., from 221.159.112.1 to 221.159.112.90) to inspect the packets coming into the company.
4. The rule inspects a port number (i.e., 5060 and 5061) to inspect VoIP/VoLTE packet.

5. If the incoming packets match the rules above, the general firewall sends the packets to VoIP/VoLTE filter for additional inspection because the general firewall can not inspect contents of the VoIP/VoLTE packets.

VoIP/VoLTE Filter

1. The name of the system policy is `malicious_voice_id`.
 2. The name of the rule is `block_malicious_voice_id`.
 3. The rule inspects the voice id of the VoIP/VoLTE packets to block the malicious VoIP/VoLTE packets (i.e., `11111@voip.black.com` and `22222@voip.black.com`).
 4. If the incoming packets match the rules above, the packets are blocked.
- A.3. Security Requirement 3: Mitigate HTTP and HTTPS Flood Attacks on a Company Web Server

This section shows a configuration example for mitigating http and https flood attacks on a company web server.

```
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
<system-policy>
  <system-policy-name>flood_attack_mitigation</system-policy-name>
  <rules>
    <rule-name>mitigate_http_and_https_flood_attack</rule-name>
    <condition-clause-container>
      <packet-security-ipv4-condition>
        <pkt-sec-ipv4-dest>
          <ipv4-address>
            <ipv4>221.159.112.95</ipv4>
          </ipv4-address>
        </pkt-sec-ipv4-dest>
      </packet-security-ipv4-condition>
      <packet-security-tcp-condition>
        <pkt-sec-tcp-dest-port-num>
          <port-num>80</port-num>
          <port-num>443</port-num>
        </pkt-sec-tcp-dest-port-num>
      </packet-security-tcp-condition>
    </condition-clause-container>
    <action-clause-container>
      <advanced-action>
        <attack-mitigation-control>http-and-https-flood
        </attack-mitigation-control>
      </advanced-action>
    </action-clause-container>
  </rules>
</system-policy>
</i2nsf-security-policy>
```

Figure 10: Configuration XML for General Firewall to Mitigate HTTP and HTTPS Flood Attacks on a Company Web Server

```
<i2nsf-security-policy
xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-policy-rule-for-nsf">
<system-policy>
  <system-policy-name>flood_attack_mitigation</system-policy-name>
  <rules>
    <rule-name>mitigate_http_and_https_flood_attack</rule-name>
    <condition-clause-container>
      <packet-security-ddos-condition>
        <pkt-sec-alert-rate>100</pkt-sec-alert-rate>
      </packet-security-ddos-condition>
    </condition-clause-container>
    <action-clause-container>
      <packet-action>
        <ingress-action>drop</ingress-action>
      </packet-action>
    </action-clause-container>
  </rules>
</system-policy>
</i2nsf-security-policy>
```

Figure 11: Configuration XML for HTTP and HTTPS Flood Attack Mitigation to Mitigate HTTP and HTTPS Flood Attacks on a Company Web Server

Figure 10 and Figure 11 show the configuration XML documents for general firewall and http and https flood attack mitigation to mitigate http and https flood attacks on a company web server. For the security requirement, two NSFs (i.e., a general firewall and a http and https flood attack mitigation) were used because one NSF can not meet the security requirement. The instances of XML documents for the general firewall and http and https flood attack mitigation are as follows: Note that a detailed data model for the configuration of the advanced network security function (i.e., http and https flood attack mitigation) is described in [i2nsf-advanced-nsf-dm].

General Firewall

1. The name of the system policy is flood_attack_mitigation.
2. The name of the rule is mitigate_http_and_https_flood_attack.
3. The rule inspects a destination IPv4 address (i.e., 221.159.112.95) to inspect the access packets coming into the company web server.
4. The rule inspects a port number (i.e., 80 and 443) to inspect http and https packet.

5. If the packets match the rules above, the general firewall sends the packets to http and https flood attack mitigation for additional inspection because the general firewall can not control the amount of packets for http and https packets.

HTTP and HTTPS Flood Attack Mitigation

1. The name of the system policy is `http_and_https_flood_attack_mitigation`.
2. The name of the rule is `100_per_second`.
3. The rule controls the http and https packets according to the amount of incoming packets.
4. If the incoming packets match the rules above, the packets are blocked.

Appendix B. Changes from draft-ietf-i2nsf-nsf-facing-interface-dm-03

The following changes are made from draft-ietf-i2nsf-nsf-facing-interface-dm-04:

- o We added fields for a rule (e.g., rule session aging time, rule long connection, and rule group).
- o We added fields for a condition (e.g., payload, acl number, application, target, users, url category, context, and generic context)

Appendix C. 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 D. Contributors

This document is made by the group effort of I2NSF working group. Many people actively contributed to this document. The following are considered co-authors:

- o Hyounghshick Kim (Sungkyunkwan University)
- o Daeyoung Hyun (Sungkyunkwan University)

- o Dongjin Hong (Sungkyunkwan University)
- o Liang Xia (Huawei)
- o Tae-Jin Ahn (Korea Telecom)
- o Se-Hui Lee (Korea Telecom)

Authors' Addresses

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

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

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

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

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 34129
Republic of Korea

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

EMail: linqiushi@huawei.com

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

J. Jeong
C. Chung
Sungkyunkwan University
S. Hares
L. Xia
Huawei
H. Birkholz
Fraunhofer SIT
March 11, 2019

I2NSF NSF Monitoring YANG Data Model
draft-ietf-i2nsf-nsf-monitoring-data-model-00

Abstract

This document proposes an information model and the corresponding YANG data model for monitoring Network Security Functions (NSFs) in the Interface to Network Security Functions (I2NSF) framework. If the monitoring of NSFs is performed in a comprehensive way, it is possible to detect the indication of malicious activity, anomalous behavior or the potential sign of denial of service attacks in a timely manner. This monitoring functionality is based on the monitoring information that is generated by NSFs. Thus, this document describes not only an information model for monitoring NSFs along with a YANG data diagram, but also the corresponding YANG data model for monitoring NSFs.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

"This version of this YANG module is part of RFC 6087;"

"RFC XXXX: I2NSF NSF Monitoring YANG Data Model"

"reference: RFC 6087"

Please update the "revision" date of the YANG module.

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	4
2.1. Requirements Notation	4
2.2. Definitions	4
2.3. YANG	4
3. Use Cases for NSF Monitoring Data	4
4. Classification of NSF Monitoring Data	5
4.1. Retention and Emission	6
4.2. Notifications and Events	7
4.3. Unsolicited Poll and Solicited Push	8
4.4. I2NSF Monitoring Terminology for Retained Information . .	8
5. Conveyance of NSF Monitoring Information	9
5.1. Information Types and Acquisition Methods	10
6. Basic Information Model for All Monitoring Data	11
7. Extended Information Model for Monitoring Data	11
7.1. System Alarm	11
7.1.1. Memory Alarm	12
7.1.2. CPU Alarm	12
7.1.3. Disk Alarm	12
7.1.4. Hardware Alarm	13
7.1.5. Interface Alarm	13

7.2.	System Events	13
7.2.1.	Access Violation	13
7.2.2.	Configuration Change	14
7.3.	System Log	14
7.3.1.	Access Logs	14
7.3.2.	Resource Utilization Logs	15
7.3.3.	User Activity Logs	15
7.4.	System Counters	16
7.4.1.	Interface counters	16
7.5.	NSF Events	17
7.5.1.	DDoS Event	17
7.5.2.	Session Table Event	18
7.5.3.	Virus Event	18
7.5.4.	Intrusion Event	19
7.5.5.	Botnet Event	20
7.5.6.	Web Attack Event	21
7.6.	NSF Logs	22
7.6.1.	DDoS Logs	22
7.6.2.	Virus Logs	22
7.6.3.	Intrusion Logs	23
7.6.4.	Botnet Logs	23
7.6.5.	DPI Logs	23
7.6.6.	Vulnerability Scanning Logs	24
7.6.7.	Web Attack Logs	25
7.7.	NSF Counters	25
7.7.1.	Firewall counters	25
7.7.2.	Policy Hit Counters	27
8.	NSF Monitoring Management in I2NSF	27
9.	Tree Structure	28
10.	YANG Data Model	36
11.	IANA Considerations	71
12.	Security Considerations	72
13.	References	72
13.1.	Normative References	72
13.2.	Informative References	74
Appendix A.	Changes from draft-hong-i2nsf-nsf-monitoring-data-model-06	76
Appendix B.	Acknowledgments	76
Appendix C.	Contributors	77
Authors' Addresses	77

1. Introduction

According to [I-D.ietf-i2nsf-terminology], the interface provided by a Network Security Function (NSF) (e.g., Firewall, IPS, Anti-DDoS, or Anti-Virus function) to administrative entities (e.g., Security Controller) to enable remote management (i.e., configuring and monitoring) is referred to as an I2NSF NSF-Facing Interface

[I-D.ietf-i2nsf-nsf-facing-interface-dm]. Monitoring procedures intent to acquire vital types of data with respect to NSFs, (e.g., alarms, records, and counters) via data in motion (e.g., queries, notifications, and events). The monitoring of NSF plays an important role in an overall security framework, if it is done in a timely and comprehensive way. The monitoring information generated by an NSF can be a good, early indication of anomalous behavior or malicious activity, such as denial of service attacks (DoS).

This document defines a comprehensive NSF monitoring information model that provides visibility for an NSF for Security Controller. It specifies the information and illustrates the methods that enable an NSF to provide the information required in order to be monitored in a scalable and efficient way via the NSF-Facing Interface. The information model for monitoring presented in this document is a complementary information model to the information model for the security policy provisioning functionality of the NSF-Facing Interface specified in [I-D.ietf-i2nsf-capability].

This document also defines a YANG [RFC7950] data model for monitoring NSFs, which is derived from the information model for NSF monitoring.

2. Terminology

2.1. Requirements Notation

This document does not propose a protocol standard, and the use of words such as "should" follow their ordinary English meaning and not that for normative languages defined in [RFC2119] [RFC8174].

2.2. Definitions

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

2.3. YANG

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].

3. Use Cases for NSF Monitoring Data

As mentioned earlier, monitoring plays a critical role in an overall security framework. The monitoring of the NSF provides very valuable information to the security controller in maintaining the provisioned

security posture. Besides this, there are various other reasons to monitor the NSF as listed below:

- o The security administrator with I2NSF User can configure a policy that is triggered on a specific event occurring in the NSF or the network [RFC8329] [I-D.ietf-i2nsf-consumer-facing-interface-dm]. If a security controller detects the specified event, it configures additional security functions as defined by policies.
- o The events triggered by an NSF as a result of security policy violation can be used by Security Information and Event Management (SIEM) to detect any suspicious activity in a larger correlation context.
- o The events and activity logs from an NSF can be used to build advanced analytics, such as behavior and predictive models to improve security posture in large deployments.
- o The security controller can use events from the NSF for achieving high availability. It can take corrective actions such as restarting a failed NSF and horizontally scaling up the NSF.
- o The events and activity logs from the NSF can aid in the root cause analysis of an operational issue, so it can improve debugging.
- o The activity logs from the NSF can be used to build historical data for operational and business reasons.

4. Classification of NSF Monitoring Data

In order to maintain a strong security posture, it is not only necessary not only to configure an NSF's security policies but also to continuously monitor the NSF by consuming acquirable and observable information. This enables security administrators to assess the state of the network topology in a timely fashion. It is not possible to block all the internal and external threats based on static security posture. A more practical approach is supported by enabling dynamic security measures, for which continuous visibility is required. This document defines a set of information elements (and their scope) that can be acquired from an NSF and can be used as NSF monitoring information. In essence, these types of monitoring information can be leveraged to support constant visibility on multiple levels of granularity and can be consumed by the corresponding functions.

Three basic domains about the monitoring information originating from a system entity [RFC4949] or an NSF are highlighted in this document.

- o Retention and Emission
- o Notifications and Events
- o Unsolicited Poll and Solicited Push

The Alarm Management Framework in [RFC3877] defines an Event as something that happens which may be of interest. It defines a fault as a change in status, crossing a threshold, or an external input to the system. In the I2NSF domain, I2NSF events [I-D.ietf-i2nsf-terminology] are created and the scope of the Alarm Management Framework's Events is still applicable due to its broad definition. The model presented in this document elaborates on the workflow of creating I2NSF events in the context of NSF monitoring and on the way initial I2NSF events are created.

As with I2NSF components, every generic system entity can include a set of capabilities [I-D.ietf-i2nsf-terminology] that creates information about the context, composition, configuration, state or behavior of that system entity. This information is intended to be provided to other consumers of information and in the scope of this document, which deals with NSF information monitoring in an automated fashion.

4.1. Retention and Emission

Typically, a system entity populates standardized interface, such as SNMP, NETCONF, RESTCONF or CoMI to provide and emit created information directly via NSF-Facing Interface [I-D.ietf-i2nsf-terminology]. Alternatively, the created information is retained inside the system entity (or a hierarchy of system entities in a composite device) via records or counters that are not exposed directly via NSF-Facing Interfaces.

Information emitted via standardized interfaces can be consumed by an I2NSF User [I-D.ietf-i2nsf-terminology] that includes the capability to consume information not only via an I2NSF Interface (e.g., [I-D.ietf-i2nsf-consumer-facing-interface-dm]) but also via interfaces complementary to the standardized interfaces a generic system entity provides.

Information retained on a system entity requires a corresponding I2NSF User to access aggregated records of information, typically in the form of log-files or databases. There are ways to aggregate records originating from different system entities over a network, for examples via Syslog Protocol [RFC5424] or Syslog over TCP [RFC6587]. But even if records are conveyed, the result is the same

kind of retention in form of a bigger aggregate of records on another system entity.

An I2NSF User is required to process fresh [RFC4949] records created by I2NSF Functions in order to provide them to other I2NSF Components via the corresponding I2NSF Interfaces in a timely manner. This process is effectively based on homogenizing functions, which can access and convert specific kinds of records into information that can be provided and emitted via I2NSF interfaces.

When retained or emitted, the information required to support monitoring processes has to be processed by an I2NSF User at some point in the workflow. Typical locations of these I2NSF Users are:

- o a system entity that creates the information
- o a system entity that retains an aggregation of records
- o an I2NSF Component that includes the capabilities of using standardized interfaces provided by other system entities that are not I2NSF Components
- o an I2NSF Component that creates the information

4.2. Notifications and Events

A specific task of I2NSF User is to process I2NSF Policy Rules [I-D.ietf-i2nsf-terminology]. The rules of a policy are composed of three clauses: Events, Conditions, and Actions. In consequence, an I2NSF Event is specified to trigger an I2NSF Policy Rule. Such an I2NSF Event is defined as any important occurrence over time in the system being managed, and/or in the environment of the system being managed in [I-D.ietf-i2nsf-terminology], which aligns well with the generic definition of Event from [RFC3877].

The model illustrated in this document introduces a complementary type of information that can be a conveyed notification.

Notification: An occurrence of a change of context, composition, configuration, state or behavior of a system entity that can be directly or indirectly observed by an I2NSF User and can be used as input for an event-clause in I2NSF Policy Rules.

A notification is similar to an I2NSF Event with the exception that it is created by a system entity that is not an I2NSF Component and that its importance is yet to be assessed. Semantically, a notification is not an I2NSF Event in the context of I2NSF, although they can potentially use the exact same

information or data model. In respect to [RFC3877], a Notification is a specific subset of events, because they convey information about something that happens which may be of interest. In consequence, Notifications may contain information with very low expressiveness or relevance. Hence, additional post-processing functions, such as aggregation, correlation or simple anomaly detection, might have to be employed to satisfy a level of expressiveness that is required for an event-clause of an I2NSF Policy Rule.

It is important to note that the consumer of a notification (the observer) assesses the importance of a notification and not the producer. The producer can include metadata in a notification that supports the observer in assessing the importance (even metadata about severity), but the deciding entity is an I2NSF User.

4.3. Unsolicited Poll and Solicited Push

The freshness of the monitored information depends on the acquisition method. Ideally, an I2NSF User is accessing every relevant information about the I2NSF Component and is emitting I2NSF Events to a monitor entity (e.g., Security Controller and I2NSF User) NSF timely. Publication of events via a pubsub/broker model, peer-2-peer meshes, or static defined channels are only a few examples on how a solicited push of I2NSF Events can be facilitated. The actual mechanic implemented by an I2NSF Component is out of the scope of this document.

Often, the corresponding management interfaces have to be queried in intervals or on-demand if required by an I2NSF Policy rule. In some cases, a collection of information has to be conducted via login mechanics provided by a system entity. Accessing records of information via this kind of unsolicited polls can introduce a significant latency in regard to the freshness of the monitored information. The actual definition of intervals implemented by an I2NSF Component is also out of scope of this document.

4.4. I2NSF Monitoring Terminology for Retained Information

Records: Unlike information emitted via notifications and events, records do not require immediate attention from an analyst but may be useful for visibility and retroactive cyber forensic. Depending on the record format, there are different qualities in regard to structure and detail. Records are typically stored in log-files or databases on a system entity or NSF. Records in the form of log-files usually include less structures but potentially more detailed information in regard to the changes of a system entity's characteristics. In contrast, databases often use more

strict schemas or data models, therefore enforcing a better structure. However, they inhibit storing information that do not match those models ("closed world assumption"). Records can be continuously processed by I2NSF Agents that act as I2NSF Producer and emit events via functions specifically tailored to a certain type of record. Typically, records are information generated either by an NSF or a system entity about operational and informational data, or various changes in system characteristics, such as user activities, network/traffic status, and network activity. They are important for debugging, auditing and security forensic.

Counters: A specific representation of continuous value changes of information elements that potentially occur in high frequency. Prominent example are network interface counters, e.g., PDU amount or byte amount, drop counters, and error counters. Counters are useful in debugging and visibility into operational behavior of an NSF. An I2NSF Agent that observes the progression of counters can act as an I2NSF Producer and emit events in respect to I2NSF Policy Rules.

5. Conveyance of NSF Monitoring Information

As per the use cases of NSF monitoring data, information needs to be conveyed to various I2NSF Consumers based on requirements imposed by I2NSF Capabilities and workflows. There are multiple aspects to be considered in regard to the emission of monitoring information to requesting parties as listed below:

- o **Pull-Push Model:** A set of data can be pushed by an NSF to a requesting party or pulled by a requesting party from an NSF. Specific types of information might need both the models at the same time if there are multiple I2NSF Consumers with varying requirements. In general, any I2NSF Event including a high severity assessment is considered to be of great importance and should be processed as soon as possible (push-model). Records, in contrast, are typically not as critical (pull-model). The I2NSF Architecture does not mandate a specific scheme for each type of information and is therefore out of scope of this document.
- o **Pub-Sub Model:** In order for an I2NSF Provider to push monitoring information to multiple appropriate I2NSF Consumers, a subscription can be maintained by both I2NSF Components. Discovery of available monitoring information can be supported by an I2NSF Controller that takes the role of a broker and therefore includes I2NSF Capabilities that support registration.

- o **Export Frequency:** Monitoring information can be emitted immediately upon generation by an NSF to requesting I2NSF Consumers or can be pushed periodically. The frequency of exporting the data depends upon its size and timely usefulness. It is out of the scope of I2NSF and left to each NSF implementation.
- o **Authentication:** There may be a need for authentication between an I2NSF Producer of monitoring information and its corresponding I2NSF Consumer to ensure that critical information remains confidential. Authentication in the scope of I2NSF can also require its corresponding content authorization. This may be necessary, for example, if an NSF emits monitoring information to an I2NSF Consumer outside its administrative domain. The I2NSF Architecture does not mandate when and how specific authentication has to be implemented.
- o **Data-Transfer Model:** Monitoring information can be pushed by an NSF using a connection-less model that does require a persistent connection or streamed over a persistent connection. An appropriate model depends on the I2NSF Consumer requirements and the semantics of the information to be conveyed.
- o **Data Model and Interaction Model for Data in Motion:** There are a lot of transport mechanisms such as IP, UDP, and TCP. There are also open source implementations for specific set of data such as systems counter, e.g. IPFIX [RFC7011] and NetFlow [RFC3954]. The I2NSF does not mandate any specific method for a given data set, so it is up to each implementation.

5.1. Information Types and Acquisition Methods

In this document, most defined information types defined benefit from high visibility with respect to value changes, e.g., alarms and records. In contrast, values that change monotonically in a continuous way do not benefit from this high visibility. On the contrary, emitting each change would result in a useless amount of value updates. Hence, values, such as counter, are best acquired in periodic intervals.

The mechanisms provided by YANG Push [I-D.ietf-netconf-yang-push] and YANG Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] address exactly these set of requirements. YANG also enables semantically well-structured information, as well as subscriptions to datastores or event streams - by changes or periodically.

In consequence, this information model in this document is intended to support data models used in solicited or unsolicited event streams that potentially are facilitated by a subscription mechanism. A subset of information elements defined in the information model address this domain of application.

6. Basic Information Model for All Monitoring Data

As explained in the above section, there is a wealth of data available from the NSF that can be monitored. Firstly, there must be some general information with each monitoring message sent from an NSF that helps a consumer to identify meta data with that message, which are listed as below:

- o message_version: It indicates the version of the data format and is a two-digit decimal numeral starting from 01.
- o message_type: Event, Alert, Alarm, Log, Counter, etc.
- o time_stamp: It indicates the time when the message is generated.
- o vendor_name: The name of the NSF vendor.
- o NSF_name: The name (or IP) of the NSF generating the message.
- o Module_name: The module name outputting the message.
- o Severity: It indicates the level of the logs. There are total eight levels, from 0 to 7. The smaller the numeral is, the higher the severity is.

7. Extended Information Model for Monitoring Data

This section covers the additional information associated with the system messages. The extended information model is only for the structured data such as alarm. Any unstructured data is specified with basic information model only.

7.1. System Alarm

Characteristics:

- o acquisition_method: subscription
- o emission_type: on-change
- o dampening_type: no-dampening

7.1.1. Memory Alarm

The following information should be included in a Memory Alarm:

- o event_name: MEM_USAGE_ALARM
- o module_name: It indicates the NSF module responsible for generating this alarm.
- o usage: specifies the amount of memory used.
- o threshold: The threshold triggering the alarm
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: The memory usage exceeded the threshold

7.1.2. CPU Alarm

The following information should be included in a CPU Alarm:

- o event_name: CPU_USAGE_ALARM
- o usage: Specifies the amount of CPU used.
- o threshold: The threshold triggering the event
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: The CPU usage exceeded the threshold.

7.1.3. Disk Alarm

The following information should be included in a Disk Alarm:

- o event_name: DISK_USAGE_ALARM
- o usage: Specifies the amount of disk space used.
- o threshold: The threshold triggering the event
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: The disk usage exceeded the threshold.

7.1.4. Hardware Alarm

The following information should be included in a Hardware Alarm:

- o event_name: HW_FAILURE_ALARM
- o component_name: It indicates the HW component responsible for generating this alarm.
- o threshold: The threshold triggering the alarm
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: The HW component has failed or degraded.

7.1.5. Interface Alarm

The following information should be included in a Interface Alarm:

- o event_name: IFNET_STATE_ALARM
- o interface_Name: The name of interface
- o interface_state: UP, DOWN, CONGESTED
- o threshold: The threshold triggering the event
- o severity: The severity of the alarm such as critical, high, medium, low
- o message: Current interface state

7.2. System Events

Characteristics:

- o acquisition_method: subscription
- o emission_type: on-change
- o dampening_type: on-repetition

7.2.1. Access Violation

The following information should be included in this event:

- o event_name: ACCESS_DENIED

- o user: Name of a user
- o group: Group to which a user belongs
- o login_ip_address: Login IP address of a user
- o authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, Single Sign-On (SSO) Authentication
- o message: access is denied.

7.2.2. Configuration Change

The following information should be included in this event:

- o event_name: CONFIG_CHANGE
- o user: Name of a user
- o group: Group to which a user belongs
- o login_ip_address: Login IP address of a user
- o authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- o message: Configuration is modified.

7.3. System Log

Characteristics:

- o acquisition_method: subscription
- o emission_type: on-change
- o dampening_type: on-repetition

7.3.1. Access Logs

Access logs record administrators' login, logout, and operations on a device. By analyzing them, security vulnerabilities can be identified. The following information should be included in an operation report:

- o Administrator: Administrator that operates on the device

- o login_ip_address: IP address used by an administrator to log in
- o login_mode: Specifies the administrator logs in mode e.g. root, user
- o operation_type: The operation type that the administrator execute, e.g., login, logout, and configuration.
- o result: Command execution result
- o content: Operation performed by an administrator after login.

7.3.2. Resource Utilization Logs

Running reports record the device system's running status, which is useful for device monitoring. The following information should be included in running report:

- o system_status: The current system's running status
- o CPU_usage: Specifies the CPU usage.
- o memory_usage: Specifies the memory usage.
- o disk_usage: Specifies the disk usage.
- o disk_left: Specifies the available disk space left.
- o session_number: Specifies total concurrent sessions.
- o process_number: Specifies total number of system processes.
- o in_traffic_rate: The total inbound traffic rate in pps
- o out_traffic_rate: The total outbound traffic rate in pps
- o in_traffic_speed: The total inbound traffic speed in bps
- o out_traffic_speed: The total outbound traffic speed in bps

7.3.3. User Activity Logs

User activity logs provide visibility into users' online records (such as login time, online/lockout duration, and login IP addresses) and the actions that users perform. User activity reports are helpful to identify exceptions during a user's login and network access activities.

- o user: Name of a user
- o group: Group to which a user belongs
- o login_ip_address: Login IP address of a user
- o authentication_mode: User authentication mode. e.g., Local Authentication, Third-Party Server Authentication, Authentication Exemption, SSO Authentication
- o access_mode: User access mode. e.g., PPP, SVN, LOCAL
- o online_duration: Online duration
- o lockout_duration: Lockout duration
- o type: User activities. e.g., Successful User Login, Failed Login attempts, User Logout, Successful User Password Change, Failed User Password Change, User Lockout, User Unlocking, Unknown
- o cause: Cause of a failed user activity

7.4. System Counters

Characteristics:

- o acquisition_method: subscription or query
- o emission_type: periodical
- o dampening_type: none

7.4.1. Interface counters

Interface counters provide visibility into traffic into and out of an NSF, and bandwidth usage.

- o interface_name: Network interface name configured in NSF
- o in_total_traffic_pkts: Total inbound packets
- o out_total_traffic_pkts: Total outbound packets
- o in_total_traffic_bytes: Total inbound bytes
- o out_total_traffic_bytes: Total outbound bytes
- o in_drop_traffic_pkts: Total inbound drop packets

- o out_drop_traffic_pkts: Total outbound drop packets
- o in_drop_traffic_bytes: Total inbound drop bytes
- o out_drop_traffic_bytes: Total outbound drop bytes
- o in_traffic_ave_rate: Inbound traffic average rate in pps
- o in_traffic_peak_rate: Inbound traffic peak rate in pps
- o in_traffic_ave_speed: Inbound traffic average speed in bps
- o in_traffic_peak_speed: Inbound traffic peak speed in bps
- o out_traffic_ave_rate: Outbound traffic average rate in pps
- o out_traffic_peak_rate: Outbound traffic peak rate in pps
- o out_traffic_ave_speed: Outbound traffic average speed in bps
- o out_traffic_peak_speed: Outbound traffic peak speed in bps

7.5. NSF Events

Characteristics:

- o acquisition_method: subscription
- o emission_type: on-change
- o dampening_type: none

7.5.1. DDoS Event

The following information should be included in a DDoS Event:

- o event_name: SEC_EVENT_DDoS
- o sub_attack_type: Any one of SYN flood, ACK flood, SYN-ACK flood, FIN/RST flood, TCP Connection flood, UDP flood, ICMP flood, HTTPS flood, HTTP flood, DNS query flood, DNS reply flood, SIP flood, and etc.
- o dst_ip: The IP address of a victim under attack
- o dst_port: The port number that the attack traffic aims at.
- o start_time: The time stamp indicating when the attack started

- o `end_time`: The time stamp indicating when the attack ended. If the attack is still undergoing when sending out the alarm, this field can be empty.
- o `attack_rate`: The PPS of attack traffic
- o `attack_speed`: the bps of attack traffic
- o `rule_id`: The ID of the rule being triggered
- o `rule_name`: The name of the rule being triggered
- o `profile`: Security profile that traffic matches.

7.5.2. Session Table Event

The following information should be included in a Session Table Event:

- o `event_name`: `SESSION_USAGE_HIGH`
- o `current`: The number of concurrent sessions
- o `max`: The maximum number of sessions that the session table can support
- o `threshold`: The threshold triggering the event
- o `message`: The number of session table exceeded the threshold.

7.5.3. Virus Event

The following information should be included in a Virus Event:

- o `event_Name`: `SEC_EVENT_VIRUS`
- o `virus_type`: Type of the virus. e.g., trojan, worm, macro virus type
- o `virus_name`: Name of the virus
- o `dst_ip`: The destination IP address of the packet where the virus is found
- o `src_ip`: The source IP address of the packet where the virus is found
- o `src_port`: The source port of the packet where the virus is found

- o `dst_port`: The destination port of the packet where the virus is found
- o `src_zone`: The source security zone of the packet where the virus is found
- o `dst_zone`: The destination security zone of the packet where the virus is found
- o `file_type`: The type of the file where the virus is hided within
- o `file_name`: The name of the file where the virus is hided within
- o `virus_info`: The brief introduction of the virus
- o `raw_info`: The information describing the packet triggering the event.
- o `rule_id`: The ID of the rule being triggered
- o `rule_name`: The name of the rule being triggered
- o `profile`: Security profile that traffic matches.

7.5.4. Intrusion Event

The following information should be included in an Intrusion Event:

- o `event_name`: The name of event. e.g., `SEC_EVENT_Intrusion`
- o `sub_attack_type`: Attack type, e.g., brutal force and buffer overflow
- o `src_ip`: The source IP address of the packet
- o `dst_ip`: The destination IP address of the packet
- o `src_port`: The source port number of the packet
- o `dst_port`: The destination port number of the packet
- o `src_zone`: The source security zone of the packet
- o `dst_zone`: The destination security zone of the packet
- o `protocol`: The employed transport layer protocol. e.g., TCP and UDP
- o `app`: The employed application layer protocol. e.g., HTTP and FTP

- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches
- o intrusion_info: Simple description of intrusion
- o raw_info: The information describing the packet triggering the event

7.5.5. Botnet Event

The following information should be included in a Botnet Event:

- o event_name: The name of event. e.g., SEC_EVENT_Botnet
- o botnet_name: The name of the detected botnet
- o src_ip: The source IP address of the packet
- o dst_ip: The destination IP address of the packet
- o src_port: The source port number of the packet
- o dst_port: The destination port number of the packet
- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o protocol: The employed transport layer protocol. e.g., TCP and UDP
- o app: The employed application layer protocol. e.g., HTTP and FTP
- o role: The role of the communicating parties within the botnet:
 1. The packet from the zombie host to the attacker
 2. The packet from the attacker to the zombie host
 3. The packet from the IRC/WEB server to the zombie host
 4. The packet from the zombie host to the IRC/WEB server
 5. The packet from the attacker to the IRC/WEB server
 6. The packet from the IRC/WEB server to the attacker

7. The packet from the zombie host to the victim

- o botnet_info: Simple description of Botnet
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered
- o profile: Security profile that traffic matches
- o raw_info: The information describing the packet triggering the event.

7.5.6. Web Attack Event

The following information should be included in a Web Attack Alarm:

- o event_name: The name of event. e.g., SEC_EVENT_WebAttack
- o sub_attack_type: Concret web attack type. e.g., SQL injection, command injection, XSS, CSRF
- o src_ip: The source IP address of the packet
- o dst_ip: The destination IP address of the packet
- o src_port: The source port number of the packet
- o dst_port: The destination port number of the packet
- o src_zone: The source security zone of the packet
- o dst_zone: The destination security zone of the packet
- o req_method: The method of requirement. For instance, "PUT" and "GET" in HTTP
- o req_url: Requested URL
- o url_category: Matched URL category
- o filtering_type: URL filtering type. e.g., Blacklist, Whitelist, User-Defined, Predefined, Malicious Category, and Unknown
- o rule_id: The ID of the rule being triggered
- o rule_name: The name of the rule being triggered

- o profile: Security profile that traffic matches

7.6. NSF Logs

Characteristics:

- o acquisition_method: subscription
- o emission_type: on-change
- o dampening_type: on_repetition

7.6.1. DDoS Logs

Besides the fields in a DDoS Alarm, the following information should be included in a DDoS Logs:

- o attack_type: DDoS
- o attack_ave_rate: The average pps of the attack traffic within the recorded time
- o attack_ave_speed: The average bps of the attack traffic within the recorded time
- o attack_pkt_num: The number of attack packets within the recorded time
- o attack_src_ip: The source IP addresses of attack traffics. If there are a large number of IP addresses, then pick a certain number of resources according to different rules.
- o action: Actions against DDoS attacks. e.g., Allow, Alert, Block, Discard, Declare, Block-ip, and Block-service.

7.6.2. Virus Logs

Besides the fields in a Virus Alarm, the following information should be included in a Virus Logs:

- o attack_type: Virus
- o protocol: The transport layer protocol
- o app: The name of the application layer protocol
- o times: The time of detecting the virus

- o action: The actions dealing with the virus. e.g., alert and block
- o os: The OS that the virus will affect. e.g., all, android, ios, unix, and windows

7.6.3. Intrusion Logs

Besides the fields in an Intrusion Alarm, the following information should be included in an Intrusion Logs:

- o attack_type: Intrusion
- o times: The times of intrusions happened in the recorded time
- o os: The OS that the intrusion will affect. e.g., all, android, ios, unix, and windows
- o action: The actions dealing with the intrusions. e.g., Allow, Alert, Block, Discard, Declare, Block-ip, and Block-service
- o attack_rate: NUM the pps of attack traffic
- o attack_speed: NUM the bps of attack traffic

7.6.4. Botnet Logs

Besides the fields in a Botnet Alarm, the following information should be included in a Botnet Logs:

- o attack_type: Botnet
- o botnet_pkt_num: The number of the packets sent to or from the detected botnet
- o action: The actions dealing with the detected packets. e.g., Allow, Alert, Block, Discard, Declare, Block-ip, and Block-service.
- o os: The OS that the attack aims at. e.g., all, android, ios, unix, and windows.

7.6.5. DPI Logs

DPI Logs provide statistics on uploaded and downloaded files and data, sent and received emails, and alert and block records on websites. It is helpful to learn risky user behaviors and why access to some URLs is blocked or allowed with an alert record.

- o type: DPI action types. e.g., File Blocking, Data Filtering, and Application Behavior Control
- o file_name: The file name
- o file_type: The file type
- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of traffic
- o dst_region: Destination region of traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic
- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o action: Action defined in the file blocking rule, data filtering rule, or application behavior control rule that traffic matches.

7.6.6. Vulnerability Scanning Logs

Vulnerability scanning logs record the victim host and its related vulnerability information that should to be fixed. The following information should be included in the report:

- o victim_ip: IP address of the victim host which has vulnerabilities
- o vulnerability_id: The vulnerability id
- o vulnerability_level: The vulnerability level. e.g., high, middle, and low

- o OS: The operating system of the victim host
- o service: The service which has vulnerability in the victim host
- o protocol: The protocol type. e.g., TCP and UDP
- o port: The port number
- o vulnerability_info: The information about the vulnerability
- o fix_suggestion: The fix suggestion to the vulnerability.

7.6.7. Web Attack Logs

Besides the fields in an Web Attack Alarm, the following information should be included in a Web Attack Report:

- o attack_type: Web Attack
- o rsp_code: Response code
- o req_clientapp: The client application
- o req_cookies: Cookies
- o req_host: The domain name of the requested host
- o raw_info: The information describing the packet triggering the event.

7.7. NSF Counters

Characteristics:

- o acquisition_method: subscription or query
- o emission_type: periodical
- o dampening_type: none

7.7.1. Firewall counters

Firewall counters provide visibility into traffic signatures, bandwidth usage, and how the configured security and bandwidth policies have been applied.

- o src_zone: Source security zone of traffic

- o `dst_zone`: Destination security zone of traffic
- o `src_region`: Source region of traffic
- o `dst_region`: Destination region of traffic
- o `src_ip`: Source IP address of traffic
- o `src_user`: User who generates traffic
- o `dst_ip`: Destination IP address of traffic
- o `src_port`: Source port of traffic
- o `dst_port`: Destination port of traffic
- o `protocol`: Protocol type of traffic
- o `app`: Application type of traffic
- o `policy_id`: Security policy id that traffic matches
- o `policy_name`: Security policy name that traffic matches
- o `in_interface`: Inbound interface of traffic
- o `out_interface`: Outbound interface of traffic
- o `total_traffic`: Total traffic volume
- o `in_traffic_ave_rate`: Inbound traffic average rate in pps
- o `in_traffic_peak_rate`: Inbound traffic peak rate in pps
- o `in_traffic_ave_speed`: Inbound traffic average speed in bps
- o `in_traffic_peak_speed`: Inbound traffic peak speed in bps
- o `out_traffic_ave_rate`: Outbound traffic average rate in pps
- o `out_traffic_peak_rate`: Outbound traffic peak rate in pps
- o `out_traffic_ave_speed`: Outbound traffic average speed in bps
- o `out_traffic_peak_speed`: Outbound traffic peak speed in bps.

7.7.2. Policy Hit Counters

Policy Hit Counters record the security policy that traffic matches and its hit count. It can check if policy configurations are correct.

- o src_zone: Source security zone of traffic
- o dst_zone: Destination security zone of traffic
- o src_region: Source region of the traffic
- o dst_region: Destination region of the traffic
- o src_ip: Source IP address of traffic
- o src_user: User who generates traffic
- o dst_ip: Destination IP address of traffic
- o src_port: Source port of traffic
- o dst_port: Destination port of traffic
- o protocol: Protocol type of traffic
- o app: Application type of traffic
- o policy_id: Security policy id that traffic matches
- o policy_name: Security policy name that traffic matches
- o hit_times: The hit times that the security policy matches the specified traffic.

8. NSF Monitoring Management in I2NSF

A standard model for monitoring data is required for an administrator to check the monitoring data generated by an NSF. The administrator can check the monitoring data through the following process. When the NSF monitoring data that is under the standard format is generated, the NSF forwards it to the security controller. The security controller delivers it to I2NSF Consumer or Developer's Management System (DMS) so that the administrator can know the state of the I2NSF framework.

In order to communicate with other components, an I2NSF framework [RFC8329] requires the interfaces. The three main interfaces in I2NSF framework are used for sending monitoring data as follows:

- o I2NSF Consumer-Facing Interface
[I-D.ietf-i2nsf-consumer-facing-interface-dm]: When an I2NSF User makes a security policy and forwards it to the Security Controller via Consumer-Facing Interface, it can specify the threat-feed for threat prevention, the custom list, the malicious code scan group, and the event map group. They can be used as an event to be monitored by an NSF.
- o I2NSF Registration Interface
[I-D.ietf-i2nsf-registration-interface-dm]: The Network Functions Virtualization (NFV) architecture provides the lifecycle management of a Virtual Network Function (VNF) via the Ve-Vnfm interface. The role of Ve-Vnfm is to request VNF lifecycle management (e.g., the instantiation and de-instantiation of an NSF, and load balancing among NSFs), exchange configuration information, and exchange status information for a network service. In the I2NSF framework, the DMS manages data about resource states and network traffic for the lifecycle management of an NSF. Therefore, the generated monitoring data from NSFs are delivered from the Security Controller to the DMS via Registration Interface. These data are delivered from the DMS to the VNF Manager in the Management and Orchestration (MANO) in the NFV system [I-D.yang-i2nsf-nfv-architecture].
- o I2NSF NSF-Facing Interface
[I-D.ietf-i2nsf-nsf-facing-interface-dm]: After a high-level security policy from I2NSF User is translated by security policy translator [I-D.yang-i2nsf-security-policy-translation] in the Security Controller, the translated security policy (i.e., low-level policy) is applied to an NSF via NSF-Facing Interface. The monitoring data model specifies the list of events that can trigger Event-Condition-Action (ECA) policies via NSF-Facing Interface.

9. Tree Structure

The tree structure of the NSF monitoring YANG module is provided below:

```

module: ietf-i2nsf-monitor
  +--rw counters
    +--rw system-interface
      +--rw acquisition-method?          identityref
      +--rw emission-type?              identityref

```

```

+---rw dampening-type?          identityref
+---rw interface-name?          string
+---rw in-total-traffic-pkts?    uint32
+---rw out-total-traffic-pkts?   uint32
+---rw in-total-traffic-bytes?   uint32
+---rw out-total-traffic-bytes?  uint32
+---rw in-drop-traffic-pkts?     uint32
+---rw out-drop-traffic-pkts?    uint32
+---rw in-drop-traffic-bytes?    uint32
+---rw out-drop-traffic-bytes?   uint32
+---rw total-traffic?            uint32
+---rw in-traffic-ave-rate?       uint32
+---rw in-traffic-peak-rate?     uint32
+---rw in-traffic-ave-speed?     uint32
+---rw in-traffic-peak-speed?    uint32
+---rw out-traffic-ave-rate?     uint32
+---rw out-traffic-peak-rate?    uint32
+---rw out-traffic-ave-speed?    uint32
+---rw out-traffic-peak-speed?   uint32
+---rw message?                  string
+---rw time-stamp?               yang:date-and-time
+---rw vendor-name?              string
+---rw nsf-name?                  string
+---rw module-name?              string
+---rw severity?                  severity
+---rw nsf-firewall
+---rw acquisition-method?        identityref
+---rw emission-type?             identityref
+---rw dampening-type?            identityref
+---rw src-ip?                    inet:ipv4-address
+---rw dst-ip?                    inet:ipv4-address
+---rw src-port?                  inet:port-number
+---rw dst-port?                  inet:port-number
+---rw src-zone?                  string
+---rw dst-zone?                  string
+---rw src-region?                string
+---rw dst-region?                string
+---rw policy-id?                 uint8
+---rw policy-name?               string
+---rw src-user?                  string
+---rw protocol?                  identityref
+---rw app?                       string
+---rw total-traffic?             uint32
+---rw in-traffic-ave-rate?        uint32
+---rw in-traffic-peak-rate?       uint32
+---rw in-traffic-ave-speed?       uint32
+---rw in-traffic-peak-speed?     uint32
+---rw out-traffic-ave-rate?       uint32

```



```

    |   +---rw out-traffic-peak-rate?      uint32
    |   +---rw out-traffic-ave-speed?      uint32
    |   +---rw out-traffic-peak-speed?     uint32
+---rw nsf-policy-hits
    |   +---rw acquisition-method?         identityref
    |   +---rw emission-type?              identityref
    |   +---rw dampening-type?             identityref
    |   +---rw src-ip?                     inet:ipv4-address
    |   +---rw dst-ip?                     inet:ipv4-address
    |   +---rw src-port?                   inet:port-number
    |   +---rw dst-port?                   inet:port-number
    |   +---rw src-zone?                   string
    |   +---rw dst-zone?                   string
    |   +---rw src-region?                 string
    |   +---rw dst-region?                 string
    |   +---rw policy-id?                  uint8
    |   +---rw policy-name?                string
    |   +---rw src-user?                   string
    |   +---rw protocol?                   identityref
    |   +---rw app?                        string
    |   +---rw message?                    string
    |   +---rw time-stamp?                 yang:date-and-time
    |   +---rw vendor-name?                string
    |   +---rw nsf-name?                   string
    |   +---rw module-name?                string
    |   +---rw severity?                   severity
    |   +---rw hit-times?                   uint32
notifications:
+---n system-detection-alarm
    |   +---ro alarm-catagory?              identityref
    |   +---ro acquisition-method?           identityref
    |   +---ro emission-type?                identityref
    |   +---ro dampening-type?               identityref
    |   +---ro usage?                        uint8
    |   +---ro threshold?                    uint8
    |   +---ro message?                      string
    |   +---ro time-stamp?                   yang:date-and-time
    |   +---ro vendor-name?                  string
    |   +---ro nsf-name?                     string
    |   +---ro module-name?                  string
    |   +---ro severity?                     severity
+---n system-detection-event
    |   +---ro event-catagory?               identityref
    |   +---ro acquisition-method?            identityref
    |   +---ro emission-type?                 identityref
    |   +---ro dampening-type?                identityref
    |   +---ro user                          string

```

```

+---ro group                string
+---ro login-ip-addr        inet:ipv4-address
+---ro authentication?      identityref
+---ro message?             string
+---ro time-stamp?          yang:date-and-time
+---ro vendor-name?         string
+---ro nsf-name?            string
+---ro module-name?         string
+---ro severity?            severity
+---n nsf-detection-flood
+---ro event-name?          identityref
+---ro dst-ip?              inet:ipv4-address
+---ro dst-port?            inet:port-number
+---ro rule-id              uint8
+---ro rule-name            string
+---ro profile?             string
+---ro raw-info?            string
+---ro sub-attack-type?     identityref
+---ro start-time           yang:date-and-time
+---ro end-time             yang:date-and-time
+---ro attack-rate?         uint32
+---ro attack-speed?        uint32
+---ro message?             string
+---ro time-stamp?          yang:date-and-time
+---ro vendor-name?         string
+---ro nsf-name?            string
+---ro module-name?         string
+---ro severity?            severity
+---n nsf-detection-session-table
+---ro current-session?     uint8
+---ro maximum-session?     uint8
+---ro threshold?           uint8
+---ro message?             string
+---ro time-stamp?          yang:date-and-time
+---ro vendor-name?         string
+---ro nsf-name?            string
+---ro module-name?         string
+---ro severity?            severity
+---n nsf-detection-virus
+---ro src-ip?              inet:ipv4-address
+---ro dst-ip?              inet:ipv4-address
+---ro src-port?            inet:port-number
+---ro dst-port?            inet:port-number
+---ro src-zone?            string
+---ro dst-zone?            string
+---ro rule-id              uint8
+---ro rule-name            string
+---ro profile?             string

```

```

+---ro raw-info?          string
+---ro virus?             identityref
+---ro virus-name?       string
+---ro file-type?        string
+---ro file-name?        string
+---ro message?          string
+---ro time-stamp?       yang:date-and-time
+---ro vendor-name?      string
+---ro nsf-name?         string
+---ro module-name?      string
+---ro severity?         severity
+---n nsf-detection-intrusion
+---ro src-ip?           inet:ipv4-address
+---ro dst-ip?           inet:ipv4-address
+---ro src-port?         inet:port-number
+---ro dst-port?         inet:port-number
+---ro src-zone?         string
+---ro dst-zone?         string
+---ro rule-id           uint8
+---ro rule-name         string
+---ro profile?          string
+---ro raw-info?         string
+---ro protocol?         identityref
+---ro app?              string
+---ro sub-attack-type?  identityref
+---ro message?          string
+---ro time-stamp?       yang:date-and-time
+---ro vendor-name?      string
+---ro nsf-name?         string
+---ro module-name?      string
+---ro severity?         severity
+---n nsf-detection-botnet
+---ro src-ip?           inet:ipv4-address
+---ro dst-ip?           inet:ipv4-address
+---ro src-port?         inet:port-number
+---ro dst-port?         inet:port-number
+---ro src-zone?         string
+---ro dst-zone?         string
+---ro rule-id           uint8
+---ro rule-name         string
+---ro profile?          string
+---ro raw-info?         string
+---ro attack-type?      identityref
+---ro protocol?         identityref
+---ro botnet-name?      string
+---ro role?             string
+---ro message?          string
+---ro time-stamp?       yang:date-and-time

```

```

+---ro vendor-name?    string
+---ro nsf-name?       string
+---ro module-name?    string
+---ro severity?       severity
+---n nsf-detection-web-attack
+---ro src-ip?         inet:ipv4-address
+---ro dst-ip?         inet:ipv4-address
+---ro src-port?       inet:port-number
+---ro dst-port?       inet:port-number
+---ro src-zone?       string
+---ro dst-zone?       string
+---ro rule-id         uint8
+---ro rule-name       string
+---ro profile?        string
+---ro raw-info?       string
+---ro sub-attack-type? identityref
+---ro request-method? identityref
+---ro req-uri?        string
+---ro uri-category?   string
+---ro filtering-type* identityref
+---ro message?        string
+---ro time-stamp?     yang:date-and-time
+---ro vendor-name?    string
+---ro nsf-name?       string
+---ro module-name?    string
+---ro severity?       severity
+---n system-access-log
+---ro login-ip        inet:ipv4-address
+---ro administrator?  string
+---ro login-mode?     login-mode
+---ro operation-type? operation-type
+---ro result?         string
+---ro content?        string
+---ro acquisition-method? identityref
+---ro emission-type?  identityref
+---ro dampening-type? identityref
+---n system-res-util-log
+---ro system-status?  string
+---ro cpu-usage?      uint8
+---ro memory-usage?   uint8
+---ro disk-usage?     uint8
+---ro disk-left?      uint8
+---ro session-num?    uint8
+---ro process-num?    uint8
+---ro in-traffic-rate? uint32
+---ro out-traffic-rate? uint32
+---ro in-traffic-speed? uint32
+---ro out-traffic-speed? uint32

```

```

    +---ro acquisition-method?    identityref
    +---ro emission-type?        identityref
    +---ro dampening-type?       identityref
+---n system-user-activity-log
    +---ro acquisition-method?    identityref
    +---ro emission-type?        identityref
    +---ro dampening-type?       identityref
    +---ro user                   string
    +---ro group                  string
    +---ro login-ip-addr          inet:ipv4-address
    +---ro authentication?       identityref
    +---ro access?               identityref
    +---ro online-duration?      string
    +---ro logout-duration?      string
    +---ro additional-info?      string
+---n nsf-log-ddos
    +---ro attack-type?          identityref
    +---ro attack-ave-rate?      uint32
    +---ro attack-ave-speed?     uint32
    +---ro attack-pkt-num?       uint32
    +---ro attack-src-ip?        inet:ipv4-address
    +---ro action?               log-action
    +---ro acquisition-method?    identityref
    +---ro emission-type?        identityref
    +---ro dampening-type?       identityref
    +---ro message?              string
    +---ro time-stamp?           yang:date-and-time
    +---ro vendor-name?          string
    +---ro nsf-name?             string
    +---ro module-name?          string
    +---ro severity?             severity
+---n nsf-log-virus
    +---ro attack-type?          identityref
    +---ro action?               log-action
    +---ro os?                   string
    +---ro time                  yang:date-and-time
    +---ro acquisition-method?    identityref
    +---ro emission-type?        identityref
    +---ro dampening-type?       identityref
    +---ro message?              string
    +---ro time-stamp?           yang:date-and-time
    +---ro vendor-name?          string
    +---ro nsf-name?             string
    +---ro module-name?          string
    +---ro severity?             severity
+---n nsf-log-intrusion
    +---ro attack-type?          identityref
    +---ro action?               log-action

```

```

    +--ro time                               yang:date-and-time
    +--ro attack-rate?                       uint32
    +--ro attack-speed?                     uint32
    +--ro acquisition-method?               identityref
    +--ro emission-type?                   identityref
    +--ro dampening-type?                  identityref
    +--ro message?                         string
    +--ro time-stamp?                      yang:date-and-time
    +--ro vendor-name?                     string
    +--ro nsf-name?                        string
    +--ro module-name?                    string
    +--ro severity?                        severity
+---n nsf-log-botnet
    +--ro attack-type?                     identityref
    +--ro action?                          log-action
    +--ro botnet-pkt-num?                  uint8
    +--ro os?                             string
    +--ro acquisition-method?              identityref
    +--ro emission-type?                   identityref
    +--ro dampening-type?                  identityref
    +--ro message?                         string
    +--ro time-stamp?                      yang:date-and-time
    +--ro vendor-name?                     string
    +--ro nsf-name?                        string
    +--ro module-name?                    string
    +--ro severity?                        severity
+---n nsf-log-dpi
    +--ro attack-type?                     dpi-type
    +--ro acquisition-method?              identityref
    +--ro emission-type?                   identityref
    +--ro dampening-type?                  identityref
    +--ro src-ip?                          inet:ipv4-address
    +--ro dst-ip?                          inet:ipv4-address
    +--ro src-port?                        inet:port-number
    +--ro dst-port?                        inet:port-number
    +--ro src-zone?                        string
    +--ro dst-zone?                        string
    +--ro src-region?                      string
    +--ro dst-region?                      string
    +--ro policy-id?                       uint8
    +--ro policy-name?                     string
    +--ro src-user?                        string
    +--ro protocol?                        identityref
    +--ro app?                             string
    +--ro message?                         string
    +--ro time-stamp?                      yang:date-and-time
    +--ro vendor-name?                     string
    +--ro nsf-name?                        string

```

```

|   +---ro module-name?           string
|   +---ro severity?             severity
+---n nsf-log-vuln-scan
|   +---ro vulnerability-id?      uint8
|   +---ro victim-ip?            inet:ipv4-address
|   +---ro protocol?             identityref
|   +---ro port-num?             inet:port-number
|   +---ro level?                severity
|   +---ro os?                   string
|   +---ro vulnerability-info?    string
|   +---ro fix-suggestion?        string
|   +---ro service?              string
|   +---ro acquisition-method?    identityref
|   +---ro emission-type?        identityref
|   +---ro dampening-type?       identityref
|   +---ro message?              string
|   +---ro time-stamp?            yang:date-and-time
|   +---ro vendor-name?          string
|   +---ro nsf-name?             string
|   +---ro module-name?          string
|   +---ro severity?             severity
+---n nsf-log-web-attack
|   +---ro attack-type?          identityref
|   +---ro rsp-code?             string
|   +---ro req-clientapp?        string
|   +---ro req-cookies?          string
|   +---ro req-host?             string
|   +---ro raw-info?             string
|   +---ro acquisition-method?    identityref
|   +---ro emission-type?        identityref
|   +---ro dampening-type?       identityref
|   +---ro message?              string
|   +---ro time-stamp?            yang:date-and-time
|   +---ro vendor-name?          string
|   +---ro nsf-name?             string
|   +---ro module-name?          string
|   +---ro severity?             severity

```

Figure 1: Information Model for NSF Monitoring

10. YANG Data Model

This section introduces a YANG data model for the information model of the NSF monitoring information model.

```

<CODE BEGINS> file "ietf-i2nsf-monitor@2019-03-11.yang"
module ietf-i2nsf-monitor {
  yang-version 1.1;

```

```
namespace
  "urn:ietf:params:xml:ns:yang:ietf-i2nsf-monitor";
prefix
  iim;
import ietf-inet-types{
  prefix inet;
  reference
    "Section 4 of RFC 6991";
}
import ietf-yang-types {
  prefix yang;
  reference
    "Section 3 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: Linda Dunbar
  <mailto:Linda.dunbar@huawei.com>

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

  Editor: Chaehong Chung
  <mailto:darkhong@skku.edu>";

description
  "This module is a YANG module for monitoring NSFs.

  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 6087; see
  the RFC itself for full legal notices.";

revision "2019-03-11" {
  description "First revision";
```



```
    reference
      "RFC XXXX: I2NSF NSF Monitoring YANG Data Model";
  }

typedef severity {
  type enumeration {
    enum high {
      description
        "high-level";
    }
    enum middle {
      description
        "middle-level";
    }
    enum low {
      description
        "low-level";
    }
  }
  description
    "An indicator representing severity";
}

typedef log-action {
  type enumeration {
    enum allow {
      description
        "If action is allow";
    }
    enum alert {
      description
        "If action is alert";
    }
    enum block {
      description
        "If action is block";
    }
    enum discard {
      description
        "If action is discard";
    }
    enum declare {
      description
        "If action is declare";
    }
    enum block-ip {
      description
        "If action is block-ip";
    }
  }
}
```

```
        enum block-service{
            description
                "If action is block-service";
        }
    }
    description
        "This is used for protocol";
}
typedef dpi-type{
    type enumeration {
        enum file-blocking{
            description
                "DPI for blocking file";
        }
        enum data-filtering{
            description
                "DPI for filtering data";
        }
        enum application-behavior-control{
            description
                "DPI for controlling application behavior";
        }
    }
    description
        "This is used for dpi type";
}
typedef operation-type{
    type enumeration {
        enum login{
            description
                "Login operation";
        }
        enum logout{
            description
                "Logout operation";
        }
        enum configuration{
            description
                "Configuration operation";
        }
    }
    description
        "An indicator representing operation-type";
}
typedef login-mode{
    type enumeration {
        enum root{
            description
```

```
        "Root login-mode";
    }
    enum user{
        description
            "User login-mode";
    }
    enum guest{
        description
            "Guest login-mode";
    }
}
description
    "An indicator representing login-mode";
}

identity characteristics {
    description
        "Base identity for monitoring information
        characteristics";
}
identity acquisition-method {
    base characteristics;
    description
        "The type of acquisition-method. Can be multiple
        types at once.";
}
identity subscription {
    base acquisition-method;
    description
        "The acquisition-method type is subscription";
}
identity query {
    base acquisition-method;
    description
        "The acquisition-method type is query";
}
identity emission-type {
    base characteristics;
    description
        "The type of emission-type.";
}
identity periodical {
    base emission-type;
    description
        "The emission-type type is periodical.";
}
identity on-change {
    base emission-type;
```

```
        description
        "The emission-type type is on-change.";
    }
    identity dampening-type {
        base characteristics;
        description
        "The type of dampening-type.";
    }
    identity no-dampening {
        base dampening-type;
        description
        "The dampening-type is no-dampening.";
    }
    identity on-repetition {
        base dampening-type;
        description
        "The dampening-type is on-repetition.";
    }
    identity none {
        base dampening-type;
        description
        "The dampening-type is none.";
    }
}

identity authentication-mode {
    description
    "User authentication mode types:
    e.g., Local Authentication,
    Third-Party Server Authentication,
    Authentication Exemption, or Single Sign-On (SSO)
    Authentication.";
}
identity local-authentication {
    base authentication-mode;
    description
    "Authentication-mode : local authentication.";
}
identity third-party-server-authentication {
    base authentication-mode;
    description
    "If authentication-mode is
    third-part-server-authentication";
}
identity exemption-authentication {
    base authentication-mode;
    description
    "If authentication-mode is
    exemption-authentication";
}
```

```
}
identity sso-authentication {
  base authentication-mode;
  description
    "If authentication-mode is
    sso-authentication";
}

identity alarm-type {
  description
    "Base identity for detectable alarm types";
}
identity MEM-USAGE-ALARM {
  base alarm-type;
  description
    "A memory alarm is alerted";
}
identity CPU-USAGE-ALARM {
  base alarm-type;
  description
    "A cpu alarm is alerted";
}
identity DISK-USAGE-ALARM {
  base alarm-type;
  description
    "A disk alarm is alerted";
}
identity HW-FAILURE-ALARM {
  base alarm-type;
  description
    "A hardware alarm is alerted";
}
identity IFNET-STATE-ALARM {
  base alarm-type;
  description
    "An interface alarm is alerted";
}
identity event-type {
  description
    "Base identity for detectable event types";
}
identity ACCESS-DENIED {
  base event-type;
  description
    "The system event is access-denied.";
}
identity CONFIG-CHANGE {
  base event-type;
```

```
    description
      "The system event is config-change.";
  }

  identity flood-type {
    description
      "Base identity for detectable flood types";
  }
  identity syn-flood {
    base flood-type;
    description
      "A SYN flood is detected";
  }
  identity ack-flood {
    base flood-type;
    description
      "An ACK flood is detected";
  }
  identity syn-ack-flood {
    base flood-type;
    description
      "An SYN-ACK flood is detected";
  }
  identity fin-rst-flood {
    base flood-type;
    description
      "A FIN-RST flood is detected";
  }
  identity tcp-con-flood {
    base flood-type;
    description
      "A TCP connection flood is detected";
  }
  identity udp-flood {
    base flood-type;
    description
      "A UDP flood is detected";
  }
  identity icmp-flood {
    base flood-type;
    description
      "An ICMP flood is detected";
  }
  identity https-flood {
    base flood-type;
    description
      "A HTTPS flood is detected";
  }
}
```

```
identity http-flood {
  base flood-type;
  description
    "A HTTP flood is detected";
}
identity dns-reply-flood {
  base flood-type;
  description
    "A DNS reply flood is detected";
}
identity dns-query-flood {
  base flood-type;
  description
    "A DNS query flood is detected";
}
identity sip-flood {
  base flood-type;
  description
    "A SIP flood is detected";
}

identity nsf-event-name {
  description
    "Base identity for detectable nsf event types";
}
identity SEC-EVENT-DDOS {
  base nsf-event-name;
  description
    "The nsf event is sec-event-ddos.";
}
identity SESSION-USAGE-HIGH {
  base nsf-event-name;
  description
    "The nsf event is session-usage-high";
}
identity SEC-EVENT-VIRUS {
  base nsf-event-name;
  description
    "The nsf event is sec-event-virus";
}
identity SEC-EVENT-INTRUSION {
  base nsf-event-name;
  description
    "The nsf event is sec-event-intrusion";
}
identity SEC-EVENT-BOTNET {
  base nsf-event-name;
  description
```

```
    "The nsf event is sec-event-botnet";
}
identity SEC-EVENT-WEBATTACK {
    base nsf-event-name;
    description
        "The nsf event is sec-event-webattack";
}
identity attack-type {
    description
        "The root ID of attack based notification
        in the notification taxonomy";
}
identity system-attack-type {
    base attack-type;
    description
        "This ID is intended to be used
        in the context of system events";
}
identity nsf-attack-type {
    base attack-type;
    description
        "This ID is intended to be used
        in the context of nsf event";
}
identity botnet-attack-type {
    base nsf-attack-type;
    description
        "This is a ID stub limited to indicating
        that this attack type is botnet.
        The usual semantic and taxonomy is missing
        and name is used.";
}
identity virus-type {
    base nsf-attack-type;
    description
        "The type of virus. Can be multiple types at once.
        This attack type is associated with a detected
        system-log virus-attack";
}
identity trojan {
    base virus-type;
    description
        "The detected virus type is trojan";
}
identity worm {
    base virus-type;
    description
        "The detected virus type is worm";
```



```
}
identity macro {
  base virus-type;
  description
    "The detected virus type is macro";
}
identity intrusion-attack-type {
  base nsf-attack-type;
  description
    "The attack type is associatied with
    a detectedsystem-log intrusion";
}
identity brute-force {
  base intrusion-attack-type;
  description
    "The intrusion type is brute-force";
}
identity buffer-overflow {
  base intrusion-attack-type;
  description
    "The intrusion type is buffer-overflow";
}
identity web-attack-type {
  base nsf-attack-type;
  description
    "The attack type associated with
    a detected system-log web-attack";
}
identity command-injection {
  base web-attack-type;
  description
    "The detected web attack type is command injection";
}
identity xss {
  base web-attack-type;
  description
    "The detected web attack type is XSS";
}
identity csrf {
  base web-attack-type;
  description
    "The detected web attack type is CSRF";
}
identity ddos-attack-type {
  base nsf-attack-type;
  description
    "The attack type is associated with a detected
    nsf-log event";
```

```
}

identity req-method {
  description
    "A set of request types (if applicable).
    For instance, PUT or GET in HTTP";
}
identity put-req {
  base req-method;
  description
    "The detected request type is PUT";
}
identity get-req {
  base req-method;
  description
    "The detected request type is GET";
}

identity filter-type {
  description
    "The type of filter used to detect, for example,
    a web-attack. Can be applicable to more than
    web-attacks. Can be more than one type.";
}
identity whitelist {
  base filter-type;
  description
    "The applied filter type is whitelist";
}
identity blacklist {
  base filter-type;
  description
    "The applied filter type is blacklist";
}
identity user-defined {
  base filter-type;
  description
    "The applied filter type is user-defined";
}
identity balicious-category {
  base filter-type;
  description
    "The applied filter is balicious category";
}
identity unknown-filter {
  base filter-type;
  description
    "The applied filter is unknown";
}
```

```
}

identity access-mode {
  description
    "Base identity for detectable access mode.";
}
identity ppp {
  base access-mode;
  description
    "Access-mode : ppp";
}
identity svn {
  base access-mode;
  description
    "Access-mode : svn";
}
identity local {
  base access-mode;
  description
    "Access-mode : local";
}

identity protocol-type {
  description
    "An identity used to enable type choices in leafs
    and leaflists wrt protocol metadata.";
}
identity tcp {
  base ipv4;
  base ipv6;
  description
    "TCP protocol type.";
  reference
    "RFC 793: Transmission Control Protocol";
}
identity udp {
  base ipv4;
  base ipv6;
  description
    "UDP protocol type.";
  reference
    "RFC 768: User Datagram Protocol";
}
identity icmp {
  base ipv4;
  base ipv6;
  description
    "General ICMP protocol type.";
```

```
        reference
            "RFC 792: Internet Control Message Protocol";
    }
    identity icmpv4 {
        base ipv4;
        description
            "ICMPv4 protocol type.";
    }
    identity icmpv6 {
        base ipv6;
        description
            "ICMPv6 protocol type.";
    }
    identity ip {
        base protocol-type;
        description
            "General IP protocol type.";
        reference
            "RFC 791: Internet Protocol
            RFC 2460: Internet Protocol, Version 6 (IPv6)";
    }
    identity ipv4 {
        base ip;
        description
            "IPv4 protocol type.";
        reference
            "RFC 791: Internet Protocol";
    }
    identity ipv6 {
        base ip;
        description
            "IPv6 protocol type.";
        reference
            "RFC 2460: Internet Protocol, Version 6 (IPv6)";
    }
    identity http {
        base tcp;
        description
            "HTPP protocol type.";
        reference
            "RFC 2616: Hypertext Transfer Protocol";
    }
    identity ftp {
        base tcp;
        description
            "FTP protocol type.";
        reference
            "RFC 959: File Transfer Protocol";
    }
```

```
}
grouping common-monitoring-data {
  description
    "The data set of common monitoring";
  leaf message {
    type string;
    description
      "This is a freetext annotation of
      monitoring notification content";
  }
  leaf time-stamp {
    type yang:date-and-time;
    description
      "Indicates the time of message generation";
  }
  leaf vendor-name {
    type string;
    description
      "The name of the NSF vendor";
  }
  leaf nsf-name {
    type string;
    description
      "The name (or IP) of the NSF
      generating the message";
  }
  leaf module-name {
    type string;
    description
      "The module name outputting the message";
  }
  leaf severity {
    type severity;
    description
      "The severity of the alarm such
      asvcritical, high, middle, low.";
  }
}
grouping characteristics{
  description
    "A set of monitoring information characteristics";
  leaf acquisition-method {
    type identityref {
      base acquisition-method;
    }
    description
      "The acquisition-method for characteristics";
  }
}
```

```
    leaf emission-type {
      type identityref {
        base emission-type;
      }
      description
        "The emission-type for characteristics";
    }
    leaf dampening-type {
      type identityref {
        base dampening-type;
      }
      description
        "The dampening-type for characteristics";
    }
  }
  grouping i2nsf-system-alarm-type-content {
    description
      "A set of system alarm type contents";
    leaf usage {
      type uint8;
      description
        "specifies the amount of usage";
    }
    leaf threshold {
      type uint8;
      description
        "The threshold triggering the alarm or the event";
    }
  }
  grouping i2nsf-system-event-type-content {
    description
      "System event metadata associated
      with system events caused by user activity.";
    leaf user {
      type string;
      mandatory true;
      description
        "Name of a user";
    }
    leaf group {
      type string;
      mandatory true;
      description
        "Group to which a user belongs.";
    }
    leaf login-ip-addr {
      type inet:ipv4-address;
      mandatory true;
    }
  }
}
```

```
        description
            "Login IP address of a user.";
    }
    leaf authentication {
        type identityref {
            base authentication-mode;
        }
        description
            "The authentication-mode for authentication";
    }
}
grouping i2nsf-nsf-event-type-content-extend {
    description
        "A set of common IPv4-related NSF event
        content elements";
    leaf src-ip {
        type inet:ipv4-address;
        description
            "The source IP address of the packet";
    }
    leaf dst-ip {
        type inet:ipv4-address;
        description
            "The destination IP address of the packet";
    }
    leaf src-port {
        type inet:port-number;
        description
            "The source port of the packet";
    }
    leaf dst-port {
        type inet:port-number;
        description
            "The destination port of the packet";
    }
    leaf src-zone {
        type string;
        description
            "The source security zone of the packet";
    }
    leaf dst-zone {
        type string;
        description
            "The destination security zone of the packet";
    }
    leaf rule-id {
        type uint8;
        mandatory true;
    }
}
```

```
        description
            "The ID of the rule being triggered";
    }
    leaf rule-name {
        type string;
        mandatory true;
        description
            "The name of the rule being triggered";
    }
    leaf profile {
        type string;
        description
            "Security profile that traffic matches.";
    }
    leaf raw-info {
        type string;
        description
            "The information describing the packet
            triggering the event.";
    }
}
grouping i2nsf-nsf-event-type-content {
    description
        "A set of common IPv4-related NSF event
        content elements";
    leaf dst-ip {
        type inet:ipv4-address;
        description
            "The destination IP address of the packet";
    }
    leaf dst-port {
        type inet:port-number;
        description
            "The destination port of the packet";
    }
    leaf rule-id {
        type uint8;
        mandatory true;
        description
            "The ID of the rule being triggered";
    }
    leaf rule-name {
        type string;
        mandatory true;
        description
            "The name of the rule being triggered";
    }
    leaf profile {
```



```
        type string;
        description
            "Security profile that traffic matches.";
    }
    leaf raw-info {
        type string;
        description
            "The information describing the packet
            triggering the event.";
    }
}
grouping traffic-rates {
    description
        "A set of traffic rates
        for statistics data";
    leaf total-traffic {
        type uint32;
        description
            "Total traffic";
    }
    leaf in-traffic-ave-rate {
        type uint32;
        description
            "Inbound traffic average rate in pps";
    }
    leaf in-traffic-peak-rate {
        type uint32;
        description
            "Inbound traffic peak rate in pps";
    }
    leaf in-traffic-ave-speed {
        type uint32;
        description
            "Inbound traffic average speed in bps";
    }
    leaf in-traffic-peak-speed {
        type uint32;
        description
            "Inbound traffic peak speed in bps";
    }
    leaf out-traffic-ave-rate {
        type uint32;
        description
            "Outbound traffic average rate in pps";
    }
    leaf out-traffic-peak-rate {
        type uint32;
        description
```

```
        "Outbound traffic peak rate in pps";
    }
    leaf out-traffic-ave-speed {
        type uint32;
        description
            "Outbound traffic average speed in bps";
    }
    leaf out-traffic-peak-speed {
        type uint32;
        description
            "Outbound traffic peak speed in bps";
    }
}
grouping i2nsf-system-counter-type-content{
    description
        "A set of system counter type contents";
    leaf interface-name {
        type string;
        description
            "Network interface name configured in NSF";
    }
    leaf in-total-traffic-pkts {
        type uint32;
        description
            "Total inbound packets";
    }
    leaf out-total-traffic-pkts {
        type uint32;
        description
            "Total outbound packets";
    }
    leaf in-total-traffic-bytes {
        type uint32;
        description
            "Total inbound bytes";
    }
    leaf out-total-traffic-bytes {
        type uint32;
        description
            "Total outbound bytes";
    }
    leaf in-drop-traffic-pkts {
        type uint32;
        description
            "Total inbound drop packets";
    }
    leaf out-drop-traffic-pkts {
        type uint32;
```

```
        description
          "Total outbound drop packets";
      }
      leaf in-drop-traffic-bytes {
        type uint32;
        description
          "Total inbound drop bytes";
      }
      leaf out-drop-traffic-bytes {
        type uint32;
        description
          "Total outbound drop bytes";
      }
      uses traffic-rates;
    }
    grouping i2nsf-nsf-counters-type-content {
      description
        "A set of nsf counters type contents";
      leaf src-ip {
        type inet:ipv4-address;
        description
          "The source IP address of the packet";
      }
      leaf dst-ip {
        type inet:ipv4-address;
        description
          "The destination IP address of the packet";
      }
      leaf src-port {
        type inet:port-number;
        description
          "The source port of the packet";
      }
      leaf dst-port {
        type inet:port-number;
        description
          "The destination port of the packet";
      }
      leaf src-zone {
        type string;
        description
          "The source security zone of the packet";
      }
      leaf dst-zone {
        type string;
        description
          "The destination security zone of the packet";
      }
    }
```

```
    leaf src-region {
        type string;
        description
            "Source region of the traffic";
    }
    leaf dst-region {
        type string;
        description
            "Destination region of the traffic";
    }
    leaf policy-id {
        type uint8;
        description
            "The ID of the policy being triggered";
    }
    leaf policy-name {
        type string;
        description
            "The name of the policy being triggered";
    }
    leaf src-user {
        type string;
        description
            "User who generates traffic";
    }
    leaf protocol {
        type identityref {
            base protocol-type;
        }
        description
            "Protocol type of traffic";
    }
    leaf app {
        type string;
        description
            "Application type of traffic";
    }
}

notification system-detection-alarm {
    description
        "This notification is sent, when a system alarm
        is detected.";
    leaf alarm-category {
        type identityref {
            base alarm-type;
        }
        description
```

```
        "The alarm category for
        system-detection-alarm notification";
    }
    uses characteristics;
    uses i2nsf-system-alarm-type-content;
    uses common-monitoring-data;
}
notification system-detection-event {
    description
        "This notification is sent, when a security-sensitive
        authentication action fails.";
    leaf event-category {
        type identityref {
            base event-type;
        }
        description
            "The event category for system-detection-event";
    }
    uses characteristics;
    uses i2nsf-system-event-type-content;
    uses common-monitoring-data;
}
notification nsf-detection-flood {
    description
        "This notification is sent,
        when a specific flood type is detected";
    leaf event-name {
        type identityref {
            base SEC-EVENT-DDOS;
        }
        description
            "The event name for nsf-detection-flood";
    }
    uses i2nsf-nsf-event-type-content;
    leaf sub-attack-type {
        type identityref {
            base flood-type;
        }
        description
            "Any one of Syn flood, ACK flood, SYN-ACK flood,
            FIN/RST flood, TCP Connection flood, UDP flood,
            Icmp flood, HTTPS flood, HTTP flood, DNS query flood,
            DNS reply flood, SIP flood, and etc.";
    }
    leaf start-time {
        type yang:date-and-time;
        mandatory true;
        description
```

```
        "The time stamp indicating when the attack started";
    }
    leaf end-time {
        type yang:date-and-time;
        mandatory true;
        description
            "The time stamp indicating when the attack ended";
    }
    leaf attack-rate {
        type uint32;
        description
            "The PPS rate of attack traffic";
    }
    leaf attack-speed {
        type uint32;
        description
            "The BPS speed of attack traffic";
    }
    uses common-monitoring-data;
}

notification nsf-detection-session-table {
    description
        "This notification is sent, when an a session table
        event is detected";
    leaf current-session {
        type uint8;
        description
            "The number of concurrent sessions";
    }
    leaf maximum-session {
        type uint8;
        description
            "The maximum number of sessions that the session
            table can support";
    }
    leaf threshold {
        type uint8;
        description
            "The threshold triggering the event";
    }
    uses common-monitoring-data;
}

notification nsf-detection-virus {
    description
        "This notification is sent, when a virus is detected";
    uses i2nsf-nsf-event-type-content-extend;
    leaf virus {
        type identityref {
```

```
        base virus-type;
    }
    description
        "The virus type for nsf-detection-virus notification";
}
leaf virus-name {
    type string;
    description
        "The name of the detected virus";
}

leaf file-type {
    type string;
    description
        "The type of file virus code
        is found in (if applicable).";
}
leaf file-name {
    type string;
    description
        "The name of file virus code
        is found in (if applicable).";
}
uses common-monitoring-data;
}
notification nsf-detection-intrusion {
    description
        "This notification is send, when an intrusion event
        is detected.";
    uses i2nsf-nsf-event-type-content-extend;
    leaf protocol {
        type identityref {
            base protocol-type;
        }
        description
            "The protocol type for
            nsf-detection-intrusion notification";
    }
    leaf app {
        type string;
        description
            "The employed application layer protocol";
    }
    leaf sub-attack-type {
        type identityref {
            base intrusion-attack-type;
        }
        description
```

```
        "The sub attack type for intrusion attack";
    }
    uses common-monitoring-data;
}
notification nsf-detection-botnet {
    description
        "This notification is send, when a botnet event is
        detected";
    uses i2nsf-nsf-event-type-content-extend;
    leaf attack-type {
        type identityref {
            base botnet-attack-type;
        }
        description
            "The attack type for botnet attack";
    }
    leaf protocol {
        type identityref {
            base protocol-type;
        }
        description
            "The protocol type for nsf-detection-botnet notification";
    }
    leaf botnet-name {
        type string;
        description
            "The name of the detected botnet";
    }
    leaf role {
        type string;
        description
            "The role of the communicating
            parties within the botnet";
    }
    uses common-monitoring-data;
}
notification nsf-detection-web-attack {
    description
        "This notification is send, when an attack event is
        detected";
    uses i2nsf-nsf-event-type-content-extend;
    leaf sub-attack-type {
        type identityref {
            base web-attack-type;
        }
        description
            "Concret web attack type, e.g., sql injection,
            command injection, XSS, CSRF";
    }
}
```



```
    }
    leaf request-method {
      type identityref {
        base req-method;
      }
      description
        "The method of requirement. For instance, PUT or
        GET in HTTP";
    }
    leaf req-uri {
      type string;
      description
        "Requested URI";
    }
    leaf uri-category {
      type string;
      description
        "Matched URI category";
    }
    leaf-list filtering-type {
      type identityref {
        base filter-type;
      }
      description
        "URL filtering type, e.g., Blacklist, Whitelist,
        User-Defined, Predefined, Malicious Category,
        Unknown";
    }
    uses common-monitoring-data;
  }
  notification system-access-log {
    description
      "The notification is send, if there is
      a new system log entry about
      a system access event";
    leaf login-ip {
      type inet:ipv4-address;
      mandatory true;
      description
        "Login IP address of a user";
    }
    leaf administrator {
      type string;
      description
        "Administrator that maintains the device";
    }
    leaf login-mode {
      type login-mode;
    }
  }
```

```
        description
            "Specifies the administrator log-in mode";
    }
    leaf operation-type {
        type operation-type;
        description
            "The operation type that the administrator execute";
    }
    leaf result {
        type string;
        description
            "Command execution result";
    }
    leaf content {
        type string;
        description
            "The Operation performed by an administrator
            after login";
    }
    uses characteristics;
}

notification system-res-util-log {
    description
        "This notification is send, if there is
        a new log entry representing ressource
        utilization updates.";
    leaf system-status {
        type string;
        description
            "The current systems
            running status";
    }
    leaf cpu-usage {
        type uint8;
        description
            "Specifies the relative amount of
            cpu usage wrt plattform ressources";
    }
    leaf memory-usage {
        type uint8;
        description
            "Specifies the amount of memory usage";
    }
    leaf disk-usage {
        type uint8;
        description
            "Specifies the amount of disk usage";
    }
}
```

```
    leaf disk-left {
        type uint8;
        description
            "Specifies the amount of disk left";
    }
    leaf session-num {
        type uint8;
        description
            "The total number of sessions";
    }
    leaf process-num {
        type uint8;
        description
            "The total number of process";
    }
    leaf in-traffic-rate {
        type uint32;
        description
            "The total inbound traffic rate in pps";
    }
    leaf out-traffic-rate {
        type uint32;
        description
            "The total outbound traffic rate in pps";
    }
    leaf in-traffic-speed {
        type uint32;
        description
            "The total inbound traffic speed in bps";
    }
    leaf out-traffic-speed {
        type uint32;
        description
            "The total outbound traffic speed in bps";
    }
    uses characteristics;
}
notification system-user-activity-log {
    description
        "This notification is send, if there is
        a new user activity log entry";
    uses characteristics;
    uses i2nsf-system-event-type-content;
    leaf access {
        type identityref {
            base access-mode;
        }
        description

```

```
        "The access type for
        system-user-activity-log notification";
    }
    leaf online-duration {
        type string;
        description
            "Online duration";
    }
    leaf logout-duration {
        type string;
        description
            "Lockout duration";
    }
    leaf additional-info {
        type string;
        description
            "User activities. e.g., Successful
            User Login, Failed Login attempts,
            User Logout, Successful User
            Password Change, Failed User
            Password Change, User Lockout,
            User Unlocking, Unknown";
    }
}
notification nsf-log-ddos {
    description
        "This notification is send, if there is
        a new DDoS event log entry in the nsf log";
    leaf attack-type {
        type identityref {
            base ddos-attack-type;
        }
        description
            "The ddos attack type for
            nsf-log-ddos notification";
    }
    leaf attack-ave-rate {
        type uint32;
        description
            "The ave PPS of attack traffic";
    }
    leaf attack-ave-speed {
        type uint32;
        description
            "the ave bps of attack traffic";
    }
    leaf attack-pkt-num {
        type uint32;
    }
}
```

```
        description
            "the number of attack packets";
    }
    leaf attack-src-ip {
        type inet:ipv4-address;
        description
            "The source IP addresses of attack
            traffics. If there are a large
            amount of IP addresses, then
            pick a certain number of resources
            according to different rules.";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
    }
    uses characteristics;
    uses common-monitoring-data;
}
notification nsf-log-virus {
    description
        "This notification is send, If there is
        a new virus event log enry in the nsf log";
    leaf attack-type {
        type identityref {
            base virus-type;
        }
        description
            "The virus type for nsf-log-virus notification";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
    }
    leaf os{
        type string;
        description
            "simple os information";
    }
    leaf time {
        type yang:date-and-time;
        mandatory true;
    }
}
```

```
        description
            "Indicate the time when the message
             is generated";
    }
    uses characteristics;
    uses common-monitoring-data;
}

notification nsf-log-intrusion {
    description
        "This notification is send, if there is
         a new intrusion event log entry in the nsf log";
    leaf attack-type {
        type identityref {
            base intrusion-attack-type;
        }
        description
            "The intrusion attack type for
             nsf-log-intrusion notification";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
             block, discard, declare,
             block-ip, block-service";
    }
    leaf time {
        type yang:date-and-time;
        mandatory true;
        description
            "Indicate the time when the message
             is generated";
    }
    leaf attack-rate {
        type uint32;
        description
            "The PPS of attack traffic";
    }
    leaf attack-speed {
        type uint32;
        description
            "The bps of attack traffic";
    }
    uses characteristics;
    uses common-monitoring-data;
}

notification nsf-log-botnet {
    description
```

```
        "This noticiation is send, if there is
        a new botnet event log in the nsf log";
    leaf attack-type {
        type identityref {
            base botnet-attack-type;
        }
        description
            "The botnet attack type for
            nsf-log-botnet notification";
    }
    leaf action {
        type log-action;
        description
            "Action type: allow, alert,
            block, discard, declare,
            block-ip, block-service";
    }
    leaf botnet-pkt-num{
        type uint8;
        description
            "The number of the packets sent to
            or from the detected botnet";
    }
    leaf os{
        type string;
        description
            "simple os information";
    }
    uses characteristics;
    uses common-monitoring-data;
}

notification nsf-log-dpi {
    description
        "This notification is send, if there is
        a new dpi event in the nsf log";
    leaf attack-type {
        type dpi-type;
        description
            "The type of the dpi";
    }
    uses characteristics;
    uses i2nsf-nsf-counters-type-content;
    uses common-monitoring-data;
}

notification nsf-log-vuln-scan {
    description
        "This notification is send, if there is
        a new vulnerability-scan report in the nsf log";
```

```
leaf vulnerability-id {
  type uint8;
  description
    "The vulnerability id";
}
leaf victim-ip {
  type inet:ipv4-address;
  description
    "IP address of the victim host
    which has vulnerabilities";
}
leaf protocol {
  type identityref {
    base protocol-type;
  }
  description
    "The protocol type for
    nsf-log-vuln-scan notification";
}
leaf port-num {
  type inet:port-number;
  description
    "The port number";
}
leaf level {
  type severity;
  description
    "The vulnerability severity";
}
leaf os {
  type string;
  description
    "simple os information";
}
leaf vulnerability-info {
  type string;
  description
    "The information about the vulnerability";
}
leaf fix-suggestion {
  type string;
  description
    "The fix suggestion to the vulnerability";
}
leaf service {
  type string;
  description
    "The service which has vulnerabillity in the victim host";
}
```



```
    }
    uses characteristics;
    uses common-monitoring-data;
  }
  notification nsf-log-web-attack {
    description
      "This notificatio is send, if there is
      a new web-attack event in the nsf log";
    leaf attack-type {
      type identityref {
        base web-attack-type;
      }
      description
        "The web attack type for
        nsf-log-web-attack notification";
    }
    leaf rsp-code {
      type string;
      description
        "Response code";
    }
    leaf req-clientapp {
      type string;
      description
        "The client application";
    }
    leaf req-cookies {
      type string;
      description
        "Cookies";
    }
    leaf req-host {
      type string;
      description
        "The domain name of the requested host";
    }
    leaf raw-info {
      type string;
      description
        "The information describing
        the packet triggering the event.";
    }
    uses characteristics;
    uses common-monitoring-data;
  }
  container counters {
    description
      "This is probably better covered by an import
```

```
as this will not be notifications.  
Counter are not very suitable as telemetry, maybe  
via periodic subscriptions, which would still  
violate principle of least surprise.";  
container system-interface {  
    description  
        "The system counter type is interface counter";  
    uses characteristics;  
    uses i2nsf-system-counter-type-content;  
    uses common-monitoring-data;  
}  
container nsf-firewall {  
    description  
        "The nsf counter type is firewall counter";  
    uses characteristics;  
    uses i2nsf-nsf-counters-type-content;  
    uses traffic-rates;  
}  
container nsf-policy-hits {  
    description  
        "The counters of policy hit";  
    uses characteristics;  
    uses i2nsf-nsf-counters-type-content;  
    uses common-monitoring-data;  
    leaf hit-times {  
        type uint32;  
        description  
            "The hit times for policy";  
    }  
}  
}  
  
<CODE ENDS>
```

Figure 2: Data Model of Monitoring

11. 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-monitor
Registrant Contact: The IESG.
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-monitor
  namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-monitor
  prefix: iim
  reference: RFC XXXX
```

12. Security Considerations

The YANG module described in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module which can be created, modified and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect framework operations. The monitoring YANG module should be protected by the secure communication channel, to ensure its confidentiality and integrity. In another side, the NSF and security controller can all be faked, which lead to undesirable results, i.e., leakage of an NSF's important operational information, faked NSF sending false information to mislead security controller. The mutual authentication is essential to protected against this kind of attack. The current mainstream security technologies (i.e., TLS, DTLS, IPSEC, X.509 PKI) can be employed appropriately to provide the above security functions.

In addition, to defend against the DDoS attack caused by a lot of NSFs sending massive notifications to the security controller, the rate limiting or similar mechanisms should be considered in an NSF and security controller, whether in advance or just in the process of DDoS attack.

13. References

13.1. Normative References

- [I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Event Notifications", draft-ietf-netconf-subscribed-notifications-23 (work in progress), February 2019.
- [I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <<https://www.rfc-editor.org/info/rfc3877>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<https://www.rfc-editor.org/info/rfc5424>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6587] Gerhards, R. and C. Lonvick, "Transmission of Syslog Messages over TCP", RFC 6587, DOI 10.17487/RFC6587, April 2012, <<https://www.rfc-editor.org/info/rfc6587>>.

- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [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>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

13.2. Informative References

- [I-D.ietf-i2nsf-capability]
Xia, L., Strassner, J., Basile, C., and D. Lopez,
"Information Model of NSF's Capabilities", draft-ietf-i2nsf-capability-04 (work in progress), October 2018.
- [I-D.ietf-i2nsf-consumer-facing-interface-dm]
Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares,
"I2NSF Consumer-Facing Interface YANG Data Model", draft-ietf-i2nsf-consumer-facing-interface-dm-02 (work in progress), November 2018.

- [I-D.ietf-i2nsf-nsf-facing-interface-dm]
Kim, J., Jeong, J., J., J., PARK, P., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-02 (work in progress), November 2018.
- [I-D.ietf-i2nsf-registration-interface-dm]
Hyun, S., Jeong, J., Roh, T., Wi, S., J., J., and P. PARK, "I2NSF Registration Interface YANG Data Model", draft-ietf-i2nsf-registration-interface-dm-01 (work in progress), November 2018.
- [I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-07 (work in progress), January 2019.
- [I-D.yang-i2nsf-nfv-architecture]
Yang, H., Kim, Y., Jeong, J., and J. Kim, "I2NSF on the NFV Reference Architecture", draft-yang-i2nsf-nfv-architecture-04 (work in progress), November 2018.
- [I-D.yang-i2nsf-security-policy-translation]
Yang, J., Jeong, J., and J. Kim, "Security Policy Translation in Interface to Network Security Functions", draft-yang-i2nsf-security-policy-translation-02 (work in progress), October 2018.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <<https://www.rfc-editor.org/info/rfc3954>>.
- [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>>.
- [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-hong-i2nsf-nsf-monitoring-data-model-06

The following changes are made from draft-hong-i2nsf-nsf-monitoring-data-model-06:

- o This version has reflected the comments from Tom Petch as follows.
- o In Editorial Note, RFC XXXX: I2NSF NSF Monitoring YANG Data Model is mentioned.
- o In Section 2, Requirements Language and Terminology are integrated and the explain for YANG Data Diagrams is moved to Terminology.
- o In Section 2.3, NMDA conformance is mentioned.
- o In Section 2.1, the reference [RFC8174] is added.
- o In Section 2.3, the reference [RFC8340] that specifies the format for tree diagrams is added for the tree diagrams.
- o In Section 10, the copyright of the YANG Module is added in description.
- o In Section 10, the YANG import statements includes reference statements.
- o In Section 10, the YANG Module includes RFC XXX to notify the RFC from which it comes.
- o In Section 10, the the identity for protocols includes reference statements.
- o In Section 11, for the YANG Module Names and URI in the IETF XML Registry, the section is added.
- o In Section 12,

Appendix B. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (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. The following are considered co-authors:

- o Jinyong Tim Kim (Sungkyunkwan University)
- o Dongjin Hong (Sungkyunkwan University)
- o Dacheng Zhang (Huawei)
- o Yi Wu (Aliababa Group)
- o Rakesh Kumar (Juniper Networks)
- o Anil Lohiya (Juniper Networks)

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>

Chaehong Chung
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 8541 7158
EMail: darkhong@skku.edu

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

EMail: Frank.xialiang@huawei.com

Henk Birkholz
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

EMail: henk.birkholz@sit.fraunhofer.de

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

S. Hyun
Chosun University
J. Jeong
T. Roh
S. Wi
Sungkyunkwan University
J. Park
ETRI
March 11, 2019

I2NSF Registration Interface YANG Data Model
draft-ietf-i2nsf-registration-interface-dm-02

Abstract

This document defines an information model and a YANG data model for Interface to Network Security Functions (I2NSF) Registration Interface between Security Controller and Developer's Management System (DMS). The objective of these information and data models is to support NSF capability registration and query via I2NSF Registration Interface.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

"This version of this YANG module is part of RFC XXXX;"

"RFC XXXX: I2NSF Registration Interface YANG Data Model"

"reference: RFC XXXX"

Please update the "revision" date of the YANG module.

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Terminology	3
4. Objectives	4
5. Information Model	5
5.1. NSF Capability Registration	5
5.1.1. NSF Capability Information	6
5.1.2. NSF Access Information	8
5.2. NSF Capability Query	8
6. Data Model	8
6.1. YANG Tree Diagram	8
6.1.1. Definition of Symbols in Tree Diagrams	9
6.1.2. I2NSF Registration Interface	9
6.1.3. NSF Capability Information	11
6.1.4. NSF Access Information	11
6.2. YANG Data Modules	12
7. IANA Considerations	16
8. Security Considerations	17
9. References	17
9.1. Normative References	17
9.2. Informative References	17
Appendix A. XML Example of Registration Interface Data Model . .	19
A.1. Example 1: Registration for Capabilities of General Firewall	19
A.2. Example 2: Registration for Capabilities of Time based Firewall	20

A.3.	Example 3: Registration for Capabilities of Web Filter	22
A.4.	Example 4: Registration for Capabilities of VoIP/VoLTE Filter	24
A.5.	Example 5: Registration for Capabilities of HTTP and HTTPS Flood Mitigation	26
A.6.	Example 6: Query for Capabilities of Time based Firewall	28
Appendix B.	NSF Lifecycle Managment in NFV Environments	29
Appendix C.	Changes from draft-ietf-i2nsf-registration-interface-dm-01	29
Appendix D.	Acknowledgments	29
Appendix E.	Contributors	30
Authors'	Addresses	30

1. Introduction

A number of network security functions may exist in Interface to Network Security Functions (I2NSF) framework [RFC8329]. Since these NSFs likely have different security capabilities, it is important to register the security capabilities of each NSF into the security controller. In addition, it is required to search NSFs of some required security capabilities on demand. As an example, if additional security capabilities are required to serve some security service request(s) from an I2NSF user, the security controller should be able to request the DMS for NSFs that have the required security capabilities.

This document describes an information model (see Section 5) and a YANG [RFC7950] data model (see Section 6) for the I2NSF Registration Interface [RFC8329] between the security controller and the developer's management system (DMS) to support NSF capability registration and query and NSF initiation request via the registration interface. It also describes the operations which should be performed by the security controller and the DMS via the Registration Interface using the defined model.

2. Requirements Language

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

3. Terminology

This document uses the following terms defined in [i2nsf-terminology], [capability-im], [RFC8329], [nsf-triggered-steering], [supa-policy-data-model], and [supa-policy-info-model]

- o Network Security Function (NSF): A function that is responsible for specific treatment of received packets. A Network Security Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). Sample Network Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.
[nsf-triggered-steering]
- o Data Model: A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol. [supa-policy-info-model]
- o Information Model: An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol.
[supa-policy-info-model]
- o YANG: 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. Objectives

- o Registering NSFs to I2NSF framework: Developer's Management System (DMS) in I2NSF framework is typically run by an NSF vendor, and uses Registration Interface to provide NSFs developed by the NSF vendor to Security Controller. DMS registers NSFs and their capabilities to I2NSF framework through Registration Interface. For the registered NSFs, Security Controller maintains a catalog of the capabilities of those NSFs.
- o Updating the capabilities of registered NSFs: After an NSF is registered into Security Controller, some modifications on the capability of the NSF may be required later. In this case, DMS uses Registration Interface to update the capability of the NSF, and this update should be reflected on the catalog of NSFs.
- o Querying DMS about some required capabilities: Security Controller may need some additional capabilities to serve the security service request from an I2NSF user, but none of the registered NSFs has the required capabilities. In this case, Security

Controller may query DMS about NSF(s) that can provide the required capabilities via Registration Interface.

5. Information Model

The I2NSF registration interface is used by Security Controller and Developer's Management System (DMS) in I2NSF framework. The following summarizes the operations done through the registration interface:

- 1) DMS registers NSFs and their capabilities to Security Controller via the registration interface. DMS also uses the registration interface to update the capabilities of the NSFs registered previously.
- 2) In case that Security Controller fails to find any registered NSF that can provide some required capabilities, Security Controller queries DMS about NSF(s) having the required capabilities via the registration interface.

Figure 1 shows the information model of the I2NSF registration interface, which consists of three submodels: NSF capability registration, and NSF capability query. Each submodel is used for the operations listed above. The remainder of this section will provide more in-depth explanations of each submodel.

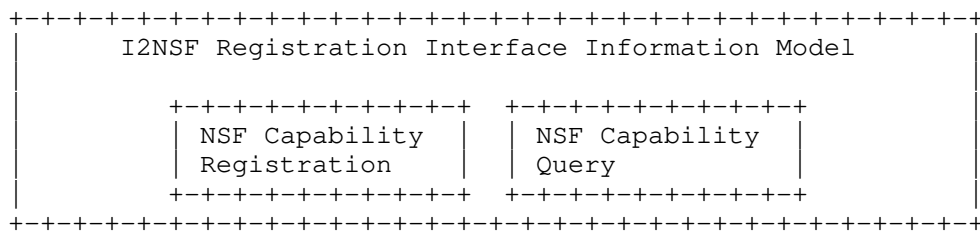


Figure 1: I2NSF Registration Interface Information Model

5.1. NSF Capability Registration

This submodel is used by DMS to register an NSF to Security Controller. Figure 2 shows how this submodel is constructed. The most important part in Figure 2 is the NSF capability, and this specifies the set of capabilities that the NSF to be registered can offer. The NSF Name contains a unique name of this NSF with the specified set of capabilities. When registering the NSF, DMS additionally includes the network access information of the NSF which is required to enable network communications with the NSF.

The following will further explain the NSF capability information and the NSF access information in more detail.

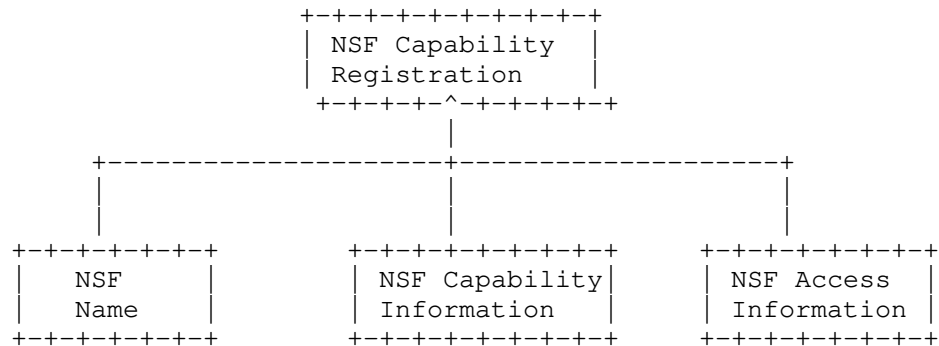


Figure 2: NSF Capability Registration Sub-Model

5.1.1. NSF Capability Information

NSF Capability Information basically describes the security capabilities of an NSF. In Figure 3, we show capability objects of an NSF. Following the information model of NSF capabilities defined in [capability-im], we share the same security capabilities: Network Security Capabilities, Content Security Capabilities, and Attack Mitigation Capabilities. Also, NSF Capability Information additionally contains the performance capabilities of an NSF as shown in Figure 3.

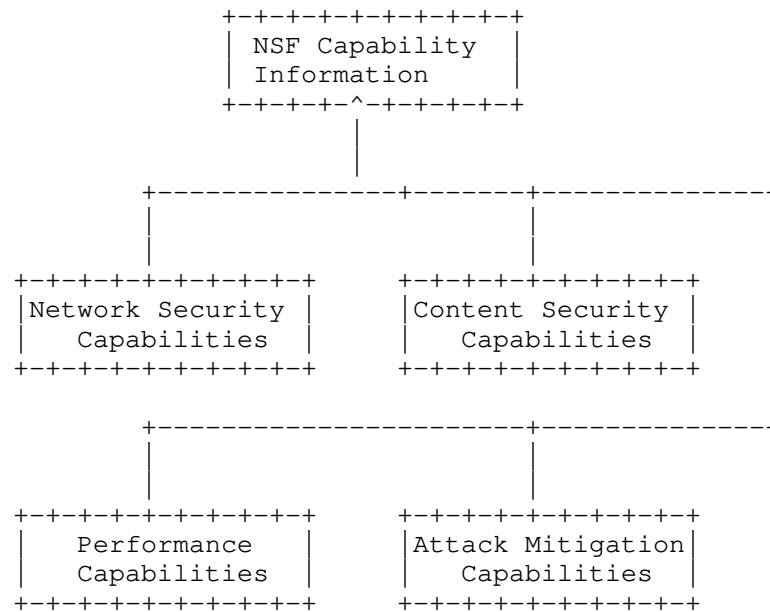


Figure 3: NSF Capability Information

5.1.1.1. Performance Capabilities

This information represents the processing capability of an NSF. This information can be used to determine whether the NSF is in congestion by comparing this with the workload that the NSF currently undergoes. Moreover, this information can specify an available amount of each type of resources such as processing power which are available on the NSF. (The registration interface can control the usages and limitations of the created instance and make the appropriate request according to the status.) As illustrated in Figure 4, this information consists of two items: Processing and Bandwidth. Processing information describes the NSF's available processing power. Bandwidth describes the information about available network amount in two cases, outbound, inbound. This two information can be used for the NSF's instance request.

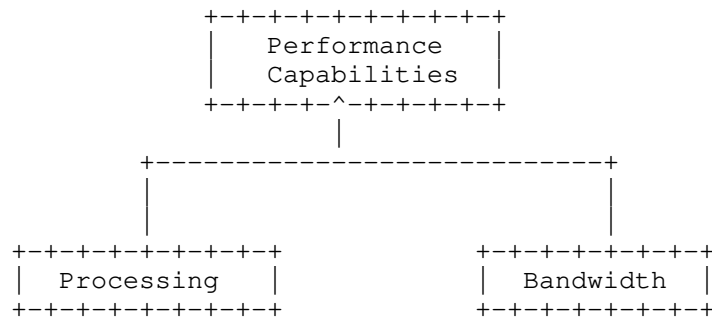


Figure 4: Performance Capability Overview

5.1.2. NSF Access Information

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

5.2. NSF Capability Query

Security Controller may require some additional capabilities to serve the security service request from an I2NSF user, but none of the registered NSFs has the required capabilities. In this case, Security Controller makes a description of the required capabilities by using the NSF capability information sub-model in Section 5.1.1, and sends DMS a query about which NSF(s) can provide these capabilities.

6. Data Model

6.1. YANG Tree Diagram

This section provides an overview of the YANG Tree diagram of the I2NSF registration interface.

6.1.1.1. Definition of Symbols in Tree Diagrams

A simplified graphical representation of the data model is used in this section. The meaning of the symbols used in the following diagrams [RFC8431] is as follows:

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

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

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

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

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

6.1.2. I2NSF Registration Interface

```
module : ietf-i2nsf-reg-interface
  +--rw nsf-capability-registration
  |   uses i2nsf-nsf-registrations

  rpcs :
    +---x nsf-capability-query
    |   uses i2nsf-nsf-capability-query
```

Figure 5: YANG tree of I2NSF Registration Interface

The I2NSF registration interface is used for the following purposes. Developer's Management System (DMS) registers NSFs and their capabilities into Security Controller via the registration interface. In case that Security Controller fails to find any NSF among the registered NSFs which can provide some required capabilities, Security Controller uses the registration interface to query DMS about NSF(s) having the required capabilities. The following sections describe the YANG data models to support these operations.

6.1.2.1. NSF Capability Registration

This section expands the i2nsf-nsf-registrations in Figure 5.

```

NSF Capability Registration
+--rw i2nsf-nsf-registrations
  +--rw i2nsf-nsf-capability-registration* [nsf-name]
    +--rw nsf-name                               string
    +--rw nsf-capability-info
      |   uses i2nsf-nsf-capability-info
    +--rw nsf-access-info
      |   uses i2nsf-nsf-access-info

```

Figure 6: YANG tree of NSF Capability Registration

When registering an NSF to Security Controller, DMS uses this module to describe what capabilities the NSF can offer. DMS includes the network access information of the NSF which is required to make a network connection with the NSF as well as the capability description of the NSF.

6.1.2.2. NSF Capability Query

This section expands the `i2nsf-nsf-capability-query` in Figure 5.

```

NSF Capability Query
+---x i2nsf-nsf-capability-query
  +---w input
  |   +---w query-i2nsf-capability-info
  |   |   uses ietf-i2nsf-capability
  +---ro output
      +---ro nsf-access-info
      |   uses i2nsf-nsf-access-info

```

Figure 7: YANG tree of NSF Capability Query

Security Controller may require some additional capabilities to provide the security service requested by an I2NSF user, but none of the registered NSFs has the required capabilities. In this case, Security Controller makes a description of the required capabilities using this module and then queries DMS about which NSF(s) can provide these capabilities. Use NETCONF RPCs to send a NSF capability query. Input data is `query-i2nsf-capability-info` and output data is `nsf-access-info`. In Figure 7, the `ietf-i2nsf-capability` refers to the module defined in `[i2nsf-capability-dm]`.

6.1.3. NSF Capability Information

This section expands the `i2nsf-nsf-capability-info` in Figure 6 and Figure 7.

```

NSF Capability Information
  +--rw i2nsf-nsf-capability-info
    +--rw i2nsf-capability
      |   uses ietf-i2nsf-capability
    +--rw nsf-performance-capability
      |   uses i2nsf-nsf-performance-capability

```

Figure 8: YANG tree of I2NSF NSF Capability Information

In Figure 8, the `ietf-i2nsf-capability` refers to the module defined in `[i2nsf-capability-dm]`. The `i2nsf-nsf-performance-capability` is used to specify the performance capability of an NSF.

6.1.3.1. NSF Performance Capability

This section expands the `i2nsf-nsf-performance-capability` in Figure 8.

```

NSF Performance Capability
  +--rw i2nsf-nsf-performance-capability
    +--rw processing
      |   +--rw processing-average   uint16
      |   +--rw processing-peak     uint16
    +--rw bandwidth
      |   +--rw outbound
      |     |   +--rw outbound-average   uint16
      |     |   +--rw outbound-peak     uint16
      |   +--rw inbound
      |     |   +--rw inbound-average   uint16
      |     |   +--rw inbound-peak     uint16

```

Figure 9: YANG tree of I2NSF NSF Performance Capability

This module is used to specify the performance capabilities of an NSF when registering or initiating the NSF.

6.1.4. NSF Access Information

This section expands the `i2nsf-nsf-access-info` in Figure 6.

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

Figure 10: YANG tree of I2NSF NSF Access Informantion

This module contains the network access information of an NSF that is required to enable network communications with the NSF.

6.2. YANG Data Modules

This section introduces a YANG data module for the information model of the required data for the registration interface between Security Controller and Developer's Management System, as defined in Section 5.

```
<CODE BEGINS> file "ietf-i2nsf-reg-interface@2019-03-11.yang"

module ietf-i2nsf-reg-interface{
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface";
  prefix "iiregi";

  import ietf-inet-types{
    prefix inet;
    reference "RFC 6991";
  }
  import ietf-i2nsf-capability{
    prefix capa;
    reference "draft-ietf-i2nsf-capability
      -data-model-02";
  }
  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: Linda Dunbar
    <mailto:Linda.dunbar@huawei.com>

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

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

Editor: Taekyun Roh
<mailto:tkroh0198@skku.edu>

Editor: Sarang Wi
<mailto:dnl9795@skku.edu>

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

description

"It defines a YANG data model for Registration 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 "The third revision";  
  reference  
    "RFC XXXX: I2NSF Registration Interface YANG Data Model";  
}  
rpc i2nsf-nsf-capability-query {  
  description  
    "Capability information that the  
    Security Controller  
    sends to the DMS";  
  input {  
    container query-i2nsf-capability-info {  
      description  
        "i2nsf capability information";  
      uses "capa:nsf-capabilities";  
      reference  
        "draft-ietf-i2nsf-capability  
        -data-model-02";  
    }  
  }
```

```
    }
    output{
        container nsf-access-info {
            description
                "nsf access information";
            uses i2nsf-nsf-access-info;
        }
    }
}
container i2nsf-nsf-registrations{
    description
        "i2nsf-nsf-registrations";
    list i2nsf-nsf-capability-registration {
        key "nsf-name";
        description
            "Requeired information for registration";
        leaf nsf-name {
            type string;
            mandatory true;
            description
                "nsf-name";
        }
        container nsf-capability-info {
            description
                "nsf-capability-information";
            uses i2nsf-nsf-capability-info;
        }
        container nsf-access-info {
            description
                "nsf-access-info";
            uses i2nsf-nsf-access-info;
        }
    }
}
grouping i2nsf-nsf-performance-capability {
    description
        "NSF performance capailities";
    container processing{
        description
            "processing info";
        leaf processing-average{
            type uint16;
            description
                "processing-average";
        }
        leaf processing-peak{
            type uint16;
        }
    }
}
```

```
        description
            "processing peak";
    }
}
container bandwidth{
    description
        "bandwidth info";
    container outbound{
        description
            "outbound";
        leaf outbound-average{
            type uint16;
            description
                "outbound-average";
        }
        leaf outbound-peak{
            type uint16;
            description
                "outbound-peak";
        }
    }
}
container inbound{
    description
        "inbound";
    leaf inbound-average{
        type uint16;
        description
            "inbound-average";
    }
    leaf inbound-peak{
        type uint16;
        description
            "inbound-peak";
    }
}
}
}
grouping i2nsf-nsf-capability-info {
    description
        "Detail information of an NSF";
    container i2nsf-capability {
        description
            "ietf i2nsf capability information";
        uses "capa:nsf-capabilities";
        reference "draft-ietf-i2nsf-capability
            -data-model-02";
    }
    container nsf-performance-capability {
```



```
        description
            "performance capability";
        uses i2nsf-nsf-performance-capability;
    }
}

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

<CODE ENDS>

Figure 11: Registration Interface YANG Data Model

7. 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-reg-interface
Registrant Contact: The IESG.
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-reg-interface
Namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface
Prefix: iiregi
Reference: RFC XXXX

8. Security Considerations

This document introduces no additional security threats and SHOULD follow the security requirements as stated in [RFC8329].

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs toIndicate Requirement Levels", RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.
- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.
- [RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.
- [RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.
- [RFC8340] Bjorklund, M. and L. Berger, "YANG Tree Diagrams", RFC 8340, March 2018.

9.2. Informative References

- [capability-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-i2nsf-capability-04 (work in progress), October 2018.
- [draft-ietf-nvo3-vxlan-gpe] Maino, Ed., F., Kreeger, Ed., L., and U. Elzur, Ed., "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.

- [i2nsf-capability-dm]
Hares, S., Jeong, J., Kim, J., Moskowitz, R., and Q. Lin,
"I2NSF Capability YANG Data Model", draft-ietf-i2nsf-
capability-data-model-02 (work in progress), November
2018.
- [i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H.
Birkholz, "Interface to Network Security Functions (I2NSF)
Terminology", draft-ietf-i2nsf-terminology-07 (work in
progress), January 2019.
- [nfv-framework]
"Network Functions Virtualisation (NFV); Architectural
Framework", ETSI GS NFV 002 ETSI GS NFV 002 V1.1.1,
October 2013.
- [nsf-triggered-steering]
Hyun, S., Jeong, J., Park, J., and S. Hares, "Service
Function Chaining-Enabled I2NSF Architecture", draft-hyun-
i2nsf-nsf-triggered-steering-06 (work in progress), July
2018.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R.
Kumar, "Framework for Interface to Network Security
Functions", RFC 8329, February 2018.
- [RFC8431] Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini,
S., and N. Bahadur, "A YANG Data Model for Routing
Information Base (RIB)", RFC 8431, September 2018.
- [supa-policy-data-model]
Halpern, J., Strassner, J., and S. van der Meer, "Generic
Policy Data Model for Simplified Use of Policy
Abstractions (SUPA)", draft-ietf-supa-generic-policy-data-
model-04 (work in progress), June 2017.
- [supa-policy-info-model]
Strassner, J., Halpern, J., and S. van der Meer, "Generic
Policy Information Model for Simplified Use of Policy
Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-
model-03 (work in progress), May 2017.

Appendix A. XML Example of Registration Interface Data Model

This section describes XML examples of the I2NSF Registration Interface data model in five NSF Registration examples and one NSF Capability Query example.

A.1. Example 1: Registration for Capabilities of General Firewall

This section shows a configuration example for capabilities registration of general firewall.

```
<i2nsf-nsf-registrations
  xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface"
  xmlns:capa="urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability">
  <i2nsf-nsf-capability-registration>
    <nsf-name>general_firewall_capability</nsf-name>
    <nsf-capability-info>
      <i2nsf-capability>
        <condition-capabilities>
          <generic-nsf-capabilities>
            <ipv4-capa>capa:ipv4-protocol</ipv4-capa>
            <ipv4-capa>capa:exact-ipv4-address</ipv4-capa>
            <ipv4-capa>capa:range-ipv4-address</ipv4-capa>
            <tcp-capa>capa:exact-tcp-port-num</tcp-capa>
            <tcp-capa>capa:range-tcp-port-num</tcp-capa>
          </generic-nsf-capabilities>
        </condition-capabilities>
        <action-capabilities>
          <ingress-action-capa>capa:pass</ingress-action-capa>
          <ingress-action-capa>capa:drop</ingress-action-capa>
          <ingress-action-capa>capa:alert</ingress-action-capa>
          <egress-action-capa>capa:pass</egress-action-capa>
          <egress-action-capa>capa:drop</egress-action-capa>
          <egress-action-capa>capa:alert</egress-action-capa>
        </action-capabilities>
      </i2nsf-capability>
    <nsf-performance-capability>
      <processing>
        <processing-average>1000</processing-average>
        <processing-peak>5000</processing-peak>
      </processing>
      <bandwidth>
        <outbound>
          <outbound-average>1000</outbound-average>
          <outbound-peak>5000</outbound-peak>
        </outbound>
        <inbound>
          <inbound-average>1000</inbound-average>
```

```

        <inbound-peak>5000</inbound-peak>
      </inbound>
    </bandwidth>
  </nsf-performance-capability>
</nsf-capability-info>
<nsf-access-info>
  <nsf-instance-name>general_firewall</nsf-instance-name>
  <nsf-address>221.159.112.100</nsf-address>
  <nsf-port-address>3000</nsf-port-address>
</nsf-access-info>
</i2nsf-nsf-capability-registration>
</i2nsf-nsf-registrations>

```

Figure 12: Configuration XML for Registration of General Firewall

Figure 12 shows the configuration XML for registration of general firewall and its capabilities are as follows.

1. The instance name of the NSF is `general_firewall`.
2. The NSF can inspect protocol, exact IPv4 address, and range IPv4 address for IPv4 packets.
3. The NSF can inspect exact port number and range port number for tcp packets.
4. The NSF can control whether the packets are allowed to pass, drop, or alert.
5. The NSF can have processing power and bandwidth.
6. The location of the NSF is 221.159.112.100.
7. The port of the NSF is 3000.

A.2. Example 2: Registration for Capabilities of Time based Firewall

This section shows a configuration example for capabilities registration of time based firewall.

```

<i2nsf-nsf-registrations
  xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface"
  xmlns:capa="urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability">
  <i2nsf-nsf-capability-registration>
    <nsf-name>time_based_firewall_capability</nsf-name>
    <nsf-capability-info>
      <i2nsf-capability>

```

```

    <time-capabilities>absolute-time</time-capabilities>
    <time-capabilities>periodic-time</time-capabilities>
    <condition-capabilities>
    <generic-nsf-capabilities>
      <ipv4-capa>capa:ipv4-protocol</ipv4-capa>
      <ipv4-capa>capa:exact-ipv4-address</ipv4-capa>
      <ipv4-capa>capa:range-ipv4-address</ipv4-capa>
    </generic-nsf-capabilities>
  </condition-capabilities>
  <action-capabilities>
    <ingress-action-capa>capa:pass</ingress-action-capa>
    <ingress-action-capa>capa:drop</ingress-action-capa>
    <ingress-action-capa>capa:alert</ingress-action-capa>
    <egress-action-capa>capa:pass</egress-action-capa>
    <egress-action-capa>capa:drop</egress-action-capa>
    <egress-action-capa>capa:alert</egress-action-capa>
  </action-capabilities>
</i2nsf-capability>
<nsf-performance-capability>
  <processing>
    <processing-average>1000</processing-average>
    <processing-peak>5000</processing-peak>
  </processing>
  <bandwidth>
    <outbound>
      <outbound-average>1000</outbound-average>
      <outbound-peak>5000</outbound-peak>
    </outbound>
    <inbound>
      <inbound-average>1000</inbound-average>
      <inbound-peak>5000</inbound-peak>
    </inbound>
  </bandwidth>
</nsf-performance-capability>
</nsf-capability-info>
<nsf-access-info>
  <nsf-instance-name>time_based_firewall</nsf-instance-name>
  <nsf-address>221.159.112.110</nsf-address>
  <nsf-port-address>3000</nsf-port-address>
</nsf-access-info>
</i2nsf-nsf-capability-registration>
</i2nsf-nsf-registrations>

```

Figure 13: Configuration XML for Registration of Time based Firewall

Figure 13 shows the configuration XML for registration of time based firewall and its capabilities are as follows.

1. The instance name of the NSF is `time_based_firewall`.
2. The NSF can execute the security policy rule according to absolute time and periodic time.
3. The NSF can inspect protocol, exact IPv4 address, and range IPv4 address for IPv4 packets.
4. The NSF can control whether the packets are allowed to pass, drop, or alert.
5. The NSF can have processing power and bandwidth.
6. The location of the NSF is 221.159.112.110.
7. The port of the NSF is 3000.

A.3. Example 3: Registration for Capabilities of Web Filter

This section shows a configuration example for capabilities registration of web filter.

```
<i2nsf-nsf-registrations
  xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface"
  xmlns:capa="urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability">
  <i2nsf-nsf-capability-registration>
    <nsf-name>web_filter_capability</nsf-name>
    <nsf-capability-info>
      <i2nsf-capability>
        <condition-capabilities>
          <advanced-nsf-capabilities>
            <http-capa>capa:url</http-capa>
          </advanced-nsf-capabilities>
        </condition-capabilities>
        <action-capabilities>
          <ingress-action-capa>capa:pass</ingress-action-capa>
          <ingress-action-capa>capa:drop</ingress-action-capa>
          <ingress-action-capa>capa:alert</ingress-action-capa>
          <egress-action-capa>capa:pass</egress-action-capa>
          <egress-action-capa>capa:drop</egress-action-capa>
          <egress-action-capa>capa:alert</egress-action-capa>
        </action-capabilities>
      </i2nsf-capability>
    <nsf-performance-capability>
      <processing>
        <processing-average>1000</processing-average>
        <processing-peak>5000</processing-peak>
      </processing>
      <bandwidth>
        <outbound>
          <outbound-average>1000</outbound-average>
          <outbound-peak>5000</outbound-peak>
        </outbound>
        <inbound>
          <inbound-average>1000</inbound-average>
          <inbound-peak>5000</inbound-peak>
        </inbound>
      </bandwidth>
    </nsf-performance-capability>
  </nsf-capability-info>
  <nsf-access-info>
    <nsf-instance-name>web_filter</nsf-instance-name>
    <nsf-address>221.159.112.120</nsf-address>
    <nsf-port-address>3000</nsf-port-address>
  </nsf-access-info>
</i2nsf-nsf-capability-registration>
</i2nsf-nsf-registrations>
```

Figure 14: Configuration XML for Registration of Web Filter

Figure 14 shows the configuration XML for registration of web filter and its capabilities are as follows.

1. The instance name of the NSF is web_filter.
2. The NSF can inspect url for http and https packets.
3. The NSF can control whether the packets are allowed to pass, drop, or alert.
4. The NSF can have processing power and bandwidth.
5. The location of the NSF is 221.159.112.120.
6. The port of the NSF is 3000.

A.4. Example 4: Registration for Capabilities of VoIP/VoLTE Filter

This section shows a configuration example for capabilities registration of VoIP/VoLTE filter.

```
<i2nsf-nsf-registrations
  xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface"
  xmlns:capa="urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability">
  <i2nsf-nsf-capability-registration>
    <nsf-name>voip_volte_filter_capability</nsf-name>
    <nsf-capability-info>
      <i2nsf-capability>
        <condition-capabilities>
          <advanced-nsf-capabilities>
            <voip-volte-capa>capa:voice-id</voip-volte-capa>
          </advanced-nsf-capabilities>
        </condition-capabilities>
        <action-capabilities>
          <ingress-action-capa>capa:pass</ingress-action-capa>
          <ingress-action-capa>capa:drop</ingress-action-capa>
          <ingress-action-capa>capa:alert</ingress-action-capa>
          <egress-action-capa>capa:pass</egress-action-capa>
          <egress-action-capa>capa:drop</egress-action-capa>
          <egress-action-capa>capa:alert</egress-action-capa>
        </action-capabilities>
      </i2nsf-capability>
    <nsf-performance-capability>
      <processing>
        <processing-average>1000</processing-average>
        <processing-peak>5000</processing-peak>
      </processing>
      <bandwidth>
        <outbound>
          <outbound-average>1000</outbound-average>
          <outbound-peak>5000</outbound-peak>
        </outbound>
        <inbound>
          <inbound-average>1000</inbound-average>
          <inbound-peak>5000</inbound-peak>
        </inbound>
      </bandwidth>
    </nsf-performance-capability>
  </nsf-capability-info>
  <nsf-access-info>
    <nsf-instance-name>voip_volte_filter</nsf-instance-name>
    <nsf-address>221.159.112.130</nsf-address>
    <nsf-port-address>3000</nsf-port-address>
  </nsf-access-info>
</i2nsf-nsf-capability-registration>
</i2nsf-nsf-registrations>
```

Figure 15: Configuration XML for Registration of VoIP/VoLTE Filter

Figure 15 shows the configuration XML for registration of VoIP/VoLTE filter and its capabilities are as follows.

1. The instance name of the NSF is `voip_volte_filter`.
2. The NSF can inspect voice id for VoIP/VoLTE packets.
3. The NSF can control whether the packets are allowed to pass, drop, or alert.
4. The NSF can have processing power and bandwidth.
5. The location of the NSF is 221.159.112.130.
6. The port of the NSF is 3000.

A.5. Example 5: Registration for Capabilities of HTTP and HTTPS Flood Mitigation

This section shows a configuration example for capabilities registration of http and https flood mitigation.

```
<i2nsf-nsf-registrations
  xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface"
  xmlns:capa="urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability">
  <i2nsf-nsf-capability-registration>
    <nsf-name>
      http_and_https_flood_mitigation_capability
    </nsf-name>
    <nsf-capability-info>
      <i2nsf-capability>
        <condition-capabilities>
          <advanced-nsf-capabilities>
            <antiddos-capa>capa:http-flood-action</antiddos-capa>
            <antiddos-capa>capa:https-flood-action</antiddos-capa>
          </advanced-nsf-capabilities>
        </condition-capabilities>
        <action-capabilities>
          <ingress-action-capa>capa:pass</ingress-action-capa>
          <ingress-action-capa>capa:drop</ingress-action-capa>
          <ingress-action-capa>capa:alert</ingress-action-capa>
          <egress-action-capa>capa:pass</egress-action-capa>
          <egress-action-capa>capa:drop</egress-action-capa>
          <egress-action-capa>capa:alert</egress-action-capa>
        </action-capabilities>
      </i2nsf-capability>
    </nsf-capability-info>
    <nsf-performance-capability>
      <processing>
```

```
        <processing-average>1000</processing-average>
        <processing-peak>5000</processing-peak>
    </processing>
    <bandwidth>
        <outbound>
            <outbound-average>1000</outbound-average>
            <outbound-peak>5000</outbound-peak>
        </outbound>
        <inbound>
            <inbound-average>1000</inbound-average>
            <inbound-peak>5000</inbound-peak>
        </inbound>
    </bandwidth>
</nsf-performance-capability>
</nsf-capability-info>
<nsf-access-info>
    <nsf-instance-name>
        http_and_https_flood_mitigation
    </nsf-instance-name>
    <nsf-address>221.159.112.140</nsf-address>
    <nsf-port-address>3000</nsf-port-address>
</nsf-access-info>
</i2nsf-nsf-capability-registration>
</i2nsf-nsf-registrations>
```

Figure 16: Configuration XML for Registration of of HTTP and HTTPS Flood Mitigation

Figure 16 shows the configuration XML for registration of VoIP/VoLTE filter and its capabilities are as follows.

1. The instance name of the NSF is http_and_https_flood_mitigation.
2. The NSF can control the amount of packets for http and https packets.
3. The NSF can control whether the packets are allowed to pass, drop, or alert.
4. The NSF can have processing power and bandwidth.
5. The location of the NSF is 221.159.112.140.
6. The port of the NSF is 3000.

A.6. Example 6: Query for Capabilities of Time based Firewall

This section shows a configuration example for capabilities query of Time based Firewall.

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <i2nsf-nsf-capability-query
    xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface"
    xmlns:capa="urn:ietf:params:xml:ns:yang:ietf-i2nsf-capability">
    <query-i2nsf-capability-info>
      <time-capabilities>absolute-time</time-capabilities>
      <time-capabilities>periodic-time</time-capabilities>
      <condition-capabilities>
        <generic-nsf-capabilities>
          <ipv4-capa>capa:ipv4-protocol</ipv4-capa>
          <ipv4-capa>capa:exact-ipv4-address</ipv4-capa>
          <ipv4-capa>capa:range-ipv4-address</ipv4-capa>
        </generic-nsf-capabilities>
      </condition-capabilities>
      <action-capabilities>
        <ingress-action-capa>capa:pass</ingress-action-capa>
        <ingress-action-capa>capa:drop</ingress-action-capa>
        <ingress-action-capa>capa:alert</ingress-action-capa>
        <egress-action-capa>capa:pass</egress-action-capa>
        <egress-action-capa>capa:drop</egress-action-capa>
        <egress-action-capa>capa:alert</egress-action-capa>
      </action-capabilities>
    </query-i2nsf-capability-info>
  </i2nsf-nsf-capability-query>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nsf-access-info
    xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-reg-interface">
    <nsf-instance-name>time-based-firewall</nsf-instance-name>
    <nsf-address>221.159.223.250</nsf-address>
    <nsf-port-address>8080</nsf-port-address>
  </nsf-access-info>
</rpc-reply>
```

Figure 17: Configuration XML for Query of Time based Firewall

Figure 17 shows the configuration of input data and output data XML for nsf capability query of time based firewall.

Appendix B. NSF Lifecycle Managemet in NFV Environments

Network Functions Virtualization (NFV) can be used to implement I2NSF framework. In NFV environments, NSFs are deployed as virtual network functions (VNFs). Security Controller can be implemented as an Element Management (EM) of the NFV architecture, and is connected with the VNF Manager (VNFM) via the Ve-Vnfm interface [nfv-framework]. Security Controller can use this interface for the purpose of the lifecycle management of NSFs. If some NSFs need to be instantiated to enforce security policies in the I2NSF framework, Security Controller could request the VNFM to instantiate them through the Ve-Vnfm interface. Or if an NSF, running as a VNF, is not used by any traffic flows for a time period, Security Controller may request deinstantiating it through the interface for efficient resource utilization.

Appendix C. Changes from draft-ietf-i2nsf-registration-interface-dm-01

The following changes have been made from draft-ietf-i2nsf-registration-interface-dm-01:

- o Section 4 has been revised to clarify major objectives of the I2NSF registration interface: NSF capability registration, NSF capability query.
- o Section 5 has been revised to describe the above-mentioned major operations of the I2NSF registration interface. Section 5.1 describes the information model for registering NSFs and their capabilities. Section 5.2 describes the information model for querying NSFs based on a description of required capabilities.
- o In section 6, the data model has been revised according to the revised information model.
- o Appendix A. has been revised to describe the XML examples of the registration interface data model in five NSF Registration examples and one NSF Capability Query example.

Appendix D. 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 E. Contributors

This document is made by the group effort of I2NSF working group. Many people actively contributed to this document. The following are considered co-authors:

- o Jinyong Tim Kim (Sungkyunkwan University)
- o Susan Hares (Huawei)
- o Diego R. Lopez (Telefonica)
- o Chung, Chaehong (Sungkyunkwan University)

Authors' Addresses

Sangwon Hyun
Department of Computer Engineering
Chosun University
309, Pilmun-daero, Dong-gu
Gwangju, Jeollanam-do 61452
Republic of Korea

EMail: shyun@chosun.ac.kr

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

Phone: +82 31 299 4957

Fax: +82 31 290 7996

EMail: pauljeong@skku.edu

URI: <http://iotlab.skku.edu/people-jaehoon-jeong.php>

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: tkroh0198@skku.edu

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: dnl9795@skku.edu

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

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

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

J. Yang
J. Jeong
J. Kim
Sungkyunkwan University
March 11, 2019

Security Policy Translation in Interface to Network Security Functions
draft-yang-i2nsf-security-policy-translation-03

Abstract

This document proposes a scheme of security policy translation (i.e., Security Policy Translator) in Interface to Network Security Functions (I2NSF) Framework. When I2NSF User delivers a high-level security policy for a security service, Security Policy Translator in Security Controller translates it into a low-level security policy for Network Security Functions (NSFs).

Status of This Memo

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

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

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Necessity for Policy Translator	3
4. Design of Policy Translator	4
4.1. Overall Structure of Policy Translator	4
4.2. DFA-based Data Extractor	6
4.2.1. Design of DFA-based Data Extractor	6
4.2.2. Example Scenario for Data Extractor	7
4.3. Data Converter	9
4.3.1. Role of Data Converter	9
4.3.2. NSF Database	10
4.3.3. Data Conversion in Data Converter	10
4.3.4. Policy Provisioning	12
4.4. CFG-based Policy Generator	13
4.4.1. Content Production	13
4.4.2. Structure Production	14
4.4.3. Generator Construction	14
5. Implementation Considerations	18
5.1. Data Model Auto-adaptation	18
5.2. Data Conversion	19
5.3. Policy Provisioning	19
6. Features of Policy Translator Design	19
7. Security Considerations	20
8. Acknowledgments	20
9. References	20
9.1. Normative References	20
9.2. Informative References	21
Appendix A. Changes from draft-yang-i2nsf-security-policy-translation-02	22
Authors' Addresses	22

1. Introduction

This document defines a scheme of a security policy translation in Interface to Network Security Functions (I2NSF) Framework [RFC8329]. First of all, this document explains the necessity of a security policy translator (shortly called policy translator) in the I2NSF framework.

The policy translator resides in Security Controller in the I2NSF framework and translates a high-level security policy to a low-level security policy for Network Security Functions (NSFs). A high-level policy is specified by I2NSF User in the I2NSF framework and is

delivered to Security Controller via Consumer-Facing Interface [consumer-facing-inf-dm]. It is translated into a low-level policy by Policy Translator in Security Controller and is delivered to NSFs to execute the rules corresponding to the low-level policy via NSF-Facing Interface [nsf-facing-inf-dm].

2. Terminology

This document uses the terminology specified in [i2nsf-terminology] [RFC8329].

3. Necessity for Policy Translator

Security Controller acts as a coordinator between I2NSF User and NSFs. Also, Security Controller has capability information of NSFs that are registered via Registration Interface [registration-inf-dm] by Developer's Management System [RFC8329]. As a coordinator, Security Controller needs to generate a low-level policy in the form of security rules intended by the high-level policy, which can be understood by the corresponding NSFs.

A high-level security policy is specified by RESTCONF/YANG [RFC8040][RFC6020], and a low-level security policy is specified by NETCONF/YANG [RFC6241][RFC6020]. The translation from a high-level security policy to the corresponding low-level security policy will be able to rapidly elevate I2NSF in real-world deployment. A rule in a high-level policy can include a broad target object, such as employees in a company for a security service (e.g., firewall and web filter). Such employees may be from human resource (HR) department, software engineering department, and advertisement department. A keyword of employee needs to be mapped to these employees from various departments. This mapping needs to be handled by a policy translator in a flexible way while understanding the intention of a policy specification. Let us consider the following two policies:

- o Block my son's computers from malicious websites.
- o Drop packets from the IP address 10.0.0.1 and 10.0.0.3 to harm.com and illegal.com

The above two sentences are examples of policies for blocking malicious websites. Both policies are for the same operation. However, NSF cannot understand the first policy, because the policy does not have any specified information for NSF. To set up the policy at an NSF, the NSF MUST receive at least the source IP address and website address for an operation. It means that the first sentence is NOT compatible for an NSF policy. Conversely, when I2NSF Users request a security policy to the system, they never make a

security policy like the second example. For generating a security policy like the second sentence, the user MUST know that the NSF needs to receive the specified information, source IP address and website address. It means that the user understands the NSF professionally, but there are not many professional users in a small size of company or at a residential area. In conclusion, the I2NSF User prefers to issue a security policy in the first sentence, but an NSF will require the same policy as the second sentence with specific information. Therefore, an advanced translation scheme of security policy is REQUIRED in I2NSF.

This document proposes an approach using Automata theory [Automata] for the policy translation, such as Deterministic Finite Automaton (DFA) and Context Free Grammar (CFG). Note that Automata theory is the foundation of programming language and compiler. Thus, with this approach, I2NSF User can easily specify a high-level security policy that will be enforced into the corresponding NSFs with a compatibly low-level security policy with the help of Policy Translator. Also, for easy management, a modularized translator structure is proposed.

4. Design of Policy Translator

Commonly used security policies are created as XML(Extensible Markup Language) [XML] files. A popular way to change the format of an XML file is to use an XSLT (Extensible Stylesheet Language Transformation) [XSLT] document. XSLT is an XML-based language to transform an input XML file into another output XML file. However, the use of XSLT makes it difficult to manage the policy translator and to handle the registration of new capabilities of NSFs. With the necessity for a policy translator, this document describes a policy translator based on Automata theory.

4.1. Overall Structure of Policy Translator

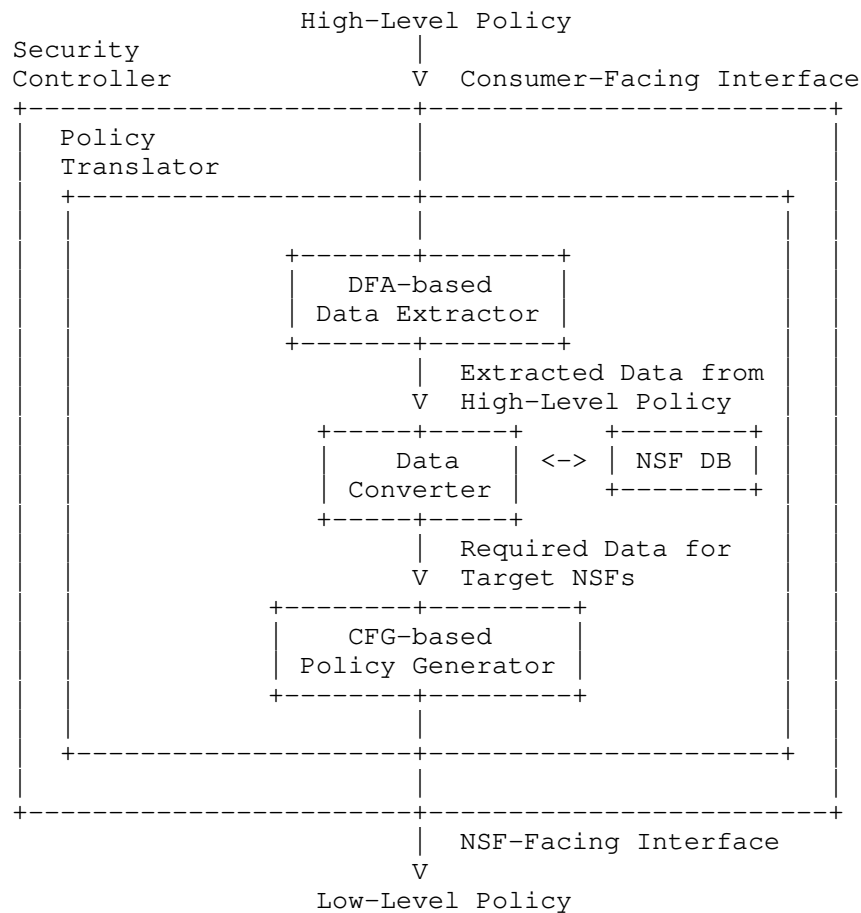


Figure 1: The Overall Design of Policy Translator

Figure 1 shows the overall design for Policy Translator in Security Controller. There are three main components for Policy Translator: Data Extractor, Data Converter, and Policy Generator.

Extractor is a DFA-based module for extracting data from a high-level policy which I2NSF User delivered via Consumer-Facing Interface. Data Converter converts the extracted data to the capabilities of target NSFs for a low-level policy. It refers to NSF Database (DB) in order to convert an abstract subject or object (e.g., IP address and website URL). Policy Generator generates a low-level policy which will execute the NSF capabilities from Converter.

4.2. DFA-based Data Extractor

4.2.1. Design of DFA-based Data Extractor

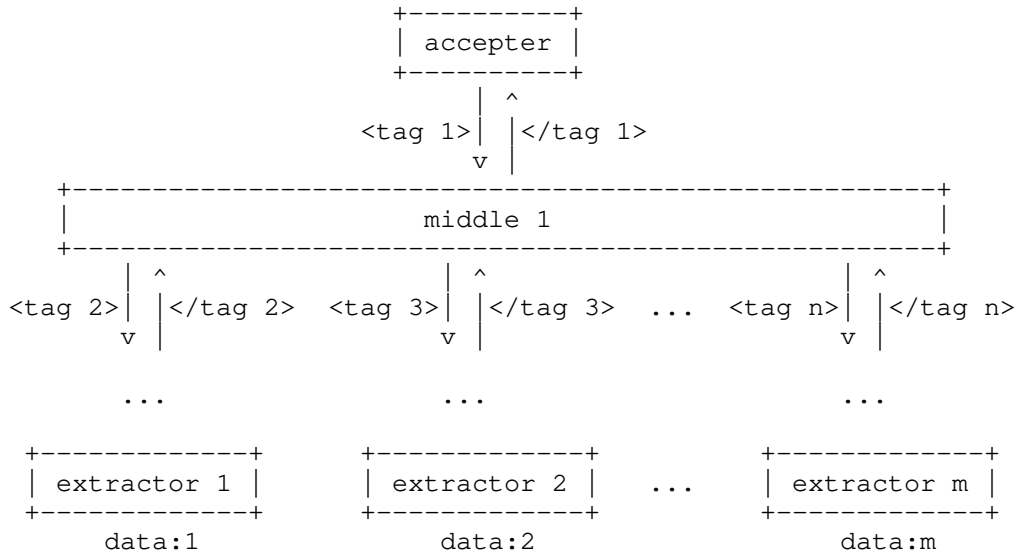


Figure 2: DFA Architecture of Data Extractor

Figure 2 shows a design for Data Extractor in the policy translator. If a high-level policy contains data along the hierarchical structure of the standard Consumer-Facing Interface YANG data model [consumer-facing-inf-dm], data can be easily extracted using the state transition machine, such as DFA. The extracted data can be processed and used by an NSF to understand it. Extractor can be constructed by designing a DFA with the same hierarchical structure as a YANG data model.

After constructing a DFA, Data Extractor can extract all of data in the entered high-level policy by using state transitions. Also, the DFA can easily detect the grammar errors of the high-level policy. The extracting algorithm of Data Extractor is as follows:

1. Start from the 'accepter' state.
2. Read the next tag from the high-level policy.
3. Transit to the corresponding state.
4. If the current state is in 'extractor', extract the corresponding data, and then go back to step 2.

5. If the current state is in 'middle', go back to step 2.
6. If there is no possible transition and arrived at 'accepter' state, the policy has no grammar error. Otherwise, there is a grammar error, so stop the process with failure.

4.2.2. Example Scenario for Data Extractor

```

<I2NSF>
  <name>block_web</name>
  <cond>
    <src>Son's_PC</src>
    <dest>malicious_websites</dest>
  </cond>
  <action>block</action>
</I2NSF>

```

Figure 3: The Example of High-level Policy

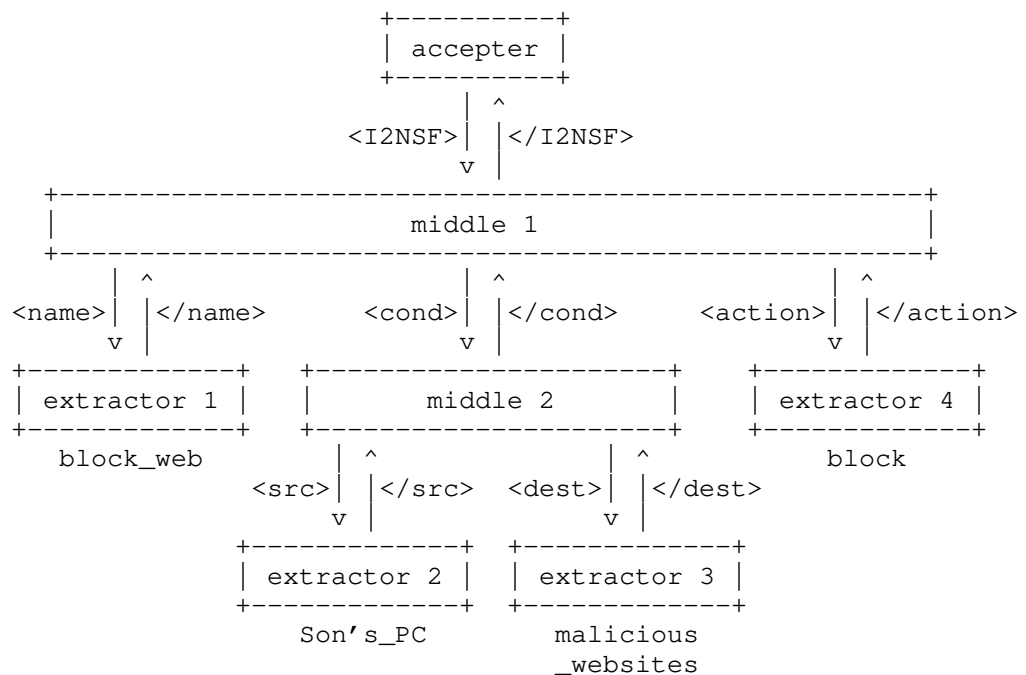


Figure 4: The Example of Data Extractor

To explain the Data Extractor process by referring to an example scenario, assume that Security Controller received a high-level

policy for a web-filtering as shown in Figure 3. Then we can construct DFA-based Data Extractor by using the design as shown in Figure 2. Figure 4 shows the architecture of Data Extractor that is based on the architecture in Figure 2 along with the input high-level policy in Figure 3. Data Extractor can automatically extract all of data in the high-level policy according to the following process:

1. Start from the 'accepter' state.
2. Read the first opening tag called '<I2NSF>', and transit to the 'middle 1' state.
3. Read the second opening tag called '<name>', and transit to the 'extractor 1' state.
4. The current state is an 'extractor' state. Extract the data of 'name' field called 'block_web'.
5. Read the second closing tag called '</name>', and go back to the 'middle 1' state.
6. Read the third opening tag called '<cond>', and transit to the 'middle 2' state.
7. Read the fourth opening tag called '<src>', and transit to the 'extractor 2' state.
8. The current state is an 'extractor' state. Extract the data of 'src' field called 'Son's_PC'.
9. Read the fourth closing tag called '</src>', and go back to the 'middle 2' state.
10. Read the fifth opening tag called '<dest>', and transit to the 'extractor 3' state.
11. The current state is an 'extractor' state. Extract the data of 'dest' field called 'malicious_websites'.
12. Read the fifth closing tag called '</dest>', and go back to the 'middle 2' state.
13. Read the third closing tag called '</cond>', and go back to the 'middle 1' state.
14. Read the sixth opening tag called '<action>', and transit to the 'extractor 4' state.

15. The current state is an 'extractor' state. Extract the data of 'action' field called 'block'.
16. Read the sixth closing tag called '</action>', and go back to the 'middle 1' state.
17. Read the first closing tag called '</I2NSF>', and go back to the 'accepter' state.
18. There is no further possible transition, and the state is finally on 'accepter' state. There is no grammar error in Figure 3 so the scanning for data extraction is finished.

The above process is constructed by an extracting algorithm. After finishing all the steps of the above process, Data Extractor can extract all of data in Figure 3, 'block_web', 'Son's_PC', 'malicious', and 'block'.

Since the translator is modularized into a DFA structure, a visual understanding is feasible. Also, The performance of Data Extractor is excellent compared to one-to-one searching of data for a particular field. In addition, the management is efficient because the DFA completely follows the hierarchy of Consumer-Facing Interface. If I2NSF User wants to modify the data model of a high-level policy, it only needs to change the connection of the relevant DFA node.

4.3. Data Converter

4.3.1. Role of Data Converter

Every NSF has its own unique capabilities. The capabilities of an NSF are registered into Security Controller by a Developer's Management System, which manages the NSF, via Registration Interface. Therefore, Security Controller already has all information about the capabilities of NSFs. This means that Security Controller can find target NSFs with only the data (e.g., subject and object for a security policy) of the high-level policy by comparing the extracted data with all capabilities of each NSF. This search process for appropriate NSFs is called by policy provisioning, and it eliminates the need for I2NSF User to specify the target NSFs explicitly in a high-level security policy.

Data Converter selects target NSFs and converts the extracted data into the capabilities of selected NSFs. If Security Controller uses this data convertor, it can provide the policy provisioning function to I2NSF User automatically. Thus, the translator design provides big benefits to the I2NSF Framework.

4.3.2. NSF Database

The NSF Database contains all the information needed to convert high-level policy data to low-level policy data. The contents of NSF Database are classified as the following two: "endpoint information" and "NSF capability information".

The first is "endpoint information". Endpoint information is necessary to convert an abstract high-level policy data such as Son's_PC, malicious to a specific low-level policy data such as 10.0.0.1, illegal.com. In the high-level policy, the range of endpoints for applying security policy MUST be provided abstractly. Thus, endpoint information is needed to specify the abstracted high-level policy data. Endpoint information is provided by I2NSF User as the high-level policy through Consumer-Facing Interface, and Security Controller builds NSF Database based on received information.

The second is "NSF capability information". Since capability is information that allows NSF to know what features it can support, NSF capability information is used in policy provisioning process to search the appropriate NSFs through the security policy. NSF capability information is provided by Developer's Management System (DMS) through Registration Interface, and Security Controller builds NSF Database based on received information. In addition, if the NSF sends monitoring information such as initiating information to Security Controller through NSF-Facing Interface, Security Controller can modify NSF Database accordingly.

4.3.3. Data Conversion in Data Converter

High-level Policy Data		Low-level Policy Data	
Rule Name	The Same value	Rule Name	
block_web		block_web	
Source	Conversion into User's IP address	Source IPv4	
Son's_PC		[10.0.0.1, 10.0.0.3]	
Destination	Conversion into malicious websites	URL Category	
malicious_websites		[harm.com, illegal.com]	
Action	Conversion into NSF Capability	Log Action	Drop Action
block		True	True

Figure 5: Example of Data Conversion

Figure 5 shows an example for describing a data conversion in Data Converter. High-level policy data MUST be converted into low-level policy data which are compatible with NSFs. If a system administrator attaches a database to Data Converter, it can convert contents by referring to the database with SQL queries. Data conversion in Figure 5 is based on the following list:

- o 'Rule Name' field does NOT need the conversion.
- o 'Source' field SHOULD be converted into a list of target IPv4 addresses.
- o 'Destination' field SHOULD be converted into a URL category list of malicious websites.
- o 'Action' field SHOULD be converted into the corresponding action(s) in NSF capabilities.

4.3.4. Policy Provisioning

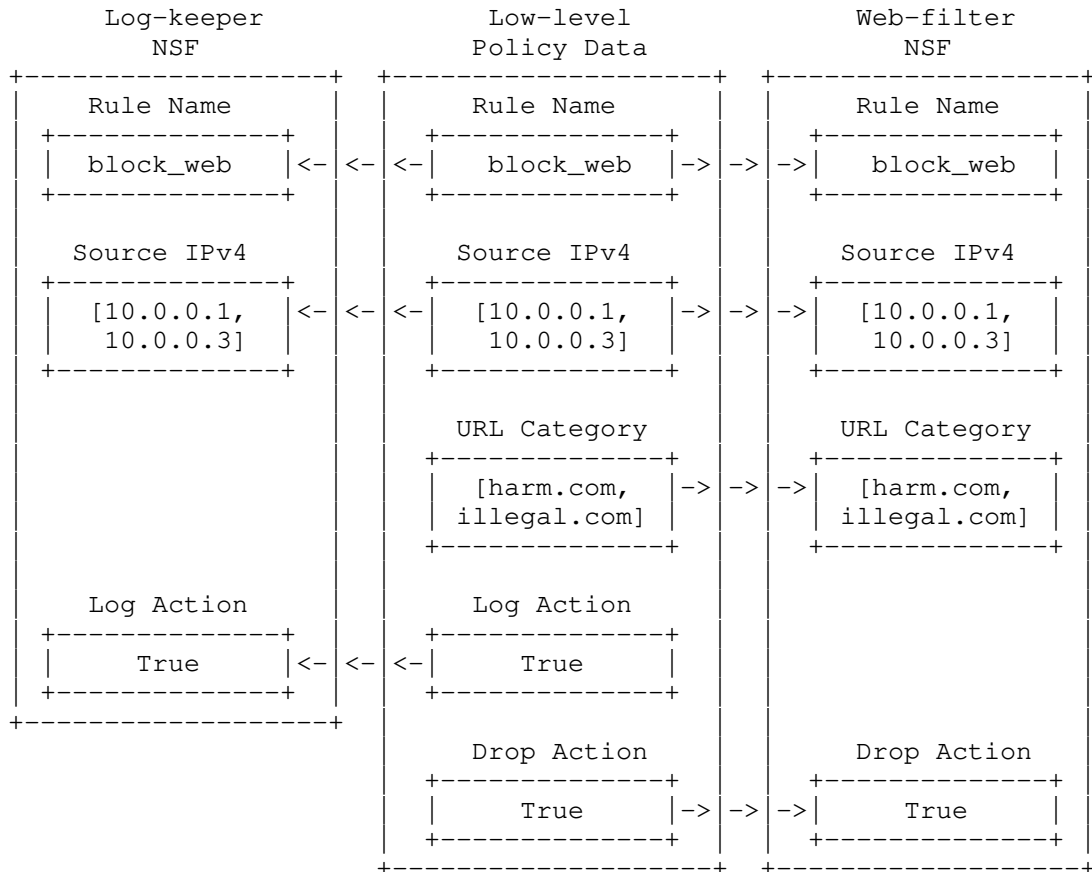


Figure 6: Example of Policy Provisioning

Generator searches for proper NSF's which can cover all of capabilities in the high-level policy. Generator searches for target NSF's by comparing only NSF capabilities which is registered by Vendor Management System. This process is called by "policy provisioning" because Generator finds proper NSF's by using only the policy. If target NSF's are found by using other data which is not included in a user's policy, it means that the user already knows the specific knowledge of an NSF in the I2NSF Framework. Figure 6 shows an example of policy provisioning. In this example, log-keeper NSF and web-filter NSF are selected for covering capabilities in the security policy. All of capabilities can be covered by two selected NSF's.

4.4. CFG-based Policy Generator

Generator makes low-level security policies for each target NSF with the extracted data. We constructed Generator by using Context Free Grammar (CFG). CFG is a set of production rules which can describe all possible strings in a given formal language (e.g., programming language). The low-level policy also has its own language based on a YANG data model of NSF-Facing Interface. Thus, we can construct the productions based on the YANG data model. The productions that makes up the low-level security policy are categorized into two types, 'Content Production' and 'Structure Production'.

4.4.1. Content Production

Content Production is for injecting data into low-level policies to be generated. A security manager (i.e., a person (or software) to make productions for security policies) can construct Content Productions in the form of an expression as the following productions:

- o [cont_prod] -> [cont_prod][cont_prod] (Where duplication is allowed.)
- o [cont_prod] -> <cont_tag>[cont_data]</cont_tag>
- o [cont_data] -> data_1 | data_2 | ... | data_n

Square brackets mean non-terminal state. If there are no non-terminal states, it means that the string is completely generated. When the duplication of content tag is allowed, the security manager adds the first production for a rule. If there is no need to allow duplication, the first production can be skipped because it is an optional production.

The second production is the main production for Content Production because it generates the tag which contains data for low-level policy. Last, the third production is for injecting data into a tag which is generated by the second production. If data is changed for an NSF, the security manager needs to change "only the third production" for data mapping in each NSF.

For example, if the security manager wants to express a low-level policy for source IP address, Content Production can be constructed in the following productions:

- o [cont_ipv4] -> [cont_ipv4][cont_ipv4] (Allow duplication.)
- o [cont_ipv4] -> <ipv4>[cont_ipv4_data]</ipv4>

- o [cont_ipv4_data] -> 10.0.0.1 | 10.0.0.3

4.4.2. Structure Production

Structure Production is for grouping other tags into a hierarchy. The security manager can construct Structure Production in the form of an expression as the following production:

- o [struct_prod] -> <struct_tag>[prod_1]...[prod_n]</struct_tag>

Structure Production can be expressed as a single production. The above production means to group other tags by the name of a tag which is called by 'struct_tag'. [prod_x] is a state for generating a tag which wants to be grouped by Structure Production. [prod_x] can be both Content Production and Structure Production. For example, if the security manager wants to express the low-level policy for the I2NSF tag, which is grouping 'name' and 'rules', Structure Production can be constructed as the following production where [cont_name] is the state for Content Production and [struct_rule] is the state for Structure Production.

- o [struct_i2nsf] -> <I2NSF>[cont_name][struct_rules]</I2NSF>

4.4.3. Generator Construction

The security manager can build a generator by combining the two productions which are described in Section 4.4.1 and Section 4.4.2. Figure 7 shows the CFG-based Generator construction of the web-filter NSF. It is constructed based on the NSF-Facing Interface Data Model in [nsf-facing-inf-dm]. According to Figure 7, the security manager can express productions for each clause as in following CFG:

1. [cont_name] -> <rule-name>[cont_name_data]</rule-name>
2. [cont_name_data] -> block_web
3. [cont_ipv4] -> [cont_ipv4][cont_ipv4] (Allow duplication)
4. [cont_ipv4] -> <ipv4>[cont_ipv4_data]</ipv4>
5. [cont_ipv4_data] -> 10.0.0.1 | 10.0.0.3
6. [cont_url] -> [cont_url][cont_url] (Allow duplication)
7. [cont_url] -> <url>[cont_url_data]</url>
8. [cont_url_data] -> harm.com | illegal.com

9. [cont_action] -> <action>[cont_action_data]</action>
10. [cont_action_data] -> drop
11. [struct_packet] -> <packet>[cont_ipv4]</packet>
12. [struct_payload] -> <payload>[cont_url]</payload>
13. [struct_cond] ->
 <condition>[struct_packet] [struct_payload]</condition>
14. [struct_rules] -> <rules>[struct_cond] [cont_action]</rules>
15. [struct_i2nsf] -> <I2NSF>[cont_name] [struct_rules]</I2NSF>

Then, Generator generates a low-level policy by using the above CFG. The low-level policy is generated by the following process:

1. Start: [struct_i2nsf]
2. Production 15: <I2NSF>[cont_name] [struct_rules]</I2NSF>
3. Production 1: <I2NSF><rule-name>[cont_name_data]</rule-name>[struct_rules]</I2NSF>
4. Production 2: <I2NSF><rule-name>block_web</rule-name>[struct_rules]</I2NSF>
5. Production 14: <I2NSF><rule-name>block_web</rule-name><rules>[struct_cond] [cont_action]</rules></I2NSF>
6. Production 13: <I2NSF><rule-name>block_web</rule-name><rules><condition>[struct_packet] [struct_payload]</condition>[cont_action]</rules></I2NSF>
7. Production 11: <I2NSF><rule-name>block_web</rule-name><rules><condition><packet>[cont_ipv4]</packet>[struct_payload]</condition>[cont_action]</rules></I2NSF>
8. Production 3: <I2NSF><rule-name>block_web</rule-name><rules><condition><packet>[cont_ipv4] [cont_ipv4]</packet>[struct_payload]</condition>[cont_action]</rules></I2NSF>
9. Production 4: <I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>[cont_ipv4_data]</ipv4><ipv4>[cont_ipv4_data]</ipv4></packet>[struct_payload]</condition>[cont_action]</rules></I2NSF>

10. Production 5: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet>[struct_payload]</condition>[cont_action]</rules></I2NSF>`
11. Production 12: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet><payload>[cont_url]</payload></condition>[cont_action]</rules></I2NSF>`
12. Production 6: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet><payload>[cont_url][cont_url]</payload></condition>[cont_action]</rules></I2NSF>`
13. Production 7: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet><payload><url>[cont_url_data]</url><url>[cont_url_data]</url></payload></condition>[cont_action]</rules></I2NSF>`
14. Production 8: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet><payload><url>harm.com</url><url>illegal.com</url></payload></condition>[cont_action]</rules></I2NSF>`
15. Production 9: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet><payload><url>harm.com</url><url>illegal.com</url></payload></condition><action>[cont_action_data]</action></rules></I2NSF>`
16. Production 10: `<I2NSF><rule-name>block_web</rule-name><rules><condition><packet><ipv4>10.0.0.1</ipv4><ipv4>10.0.0.3</ipv4></packet><payload><url>harm.com</url><url>illegal.com</url></payload></condition><action>drop</action></rules></I2NSF>`

The last production has no non-terminal state, and the low-level policy is completely generated. Figure 8 shows the generated low-level policy where tab characters and newline characters are added.

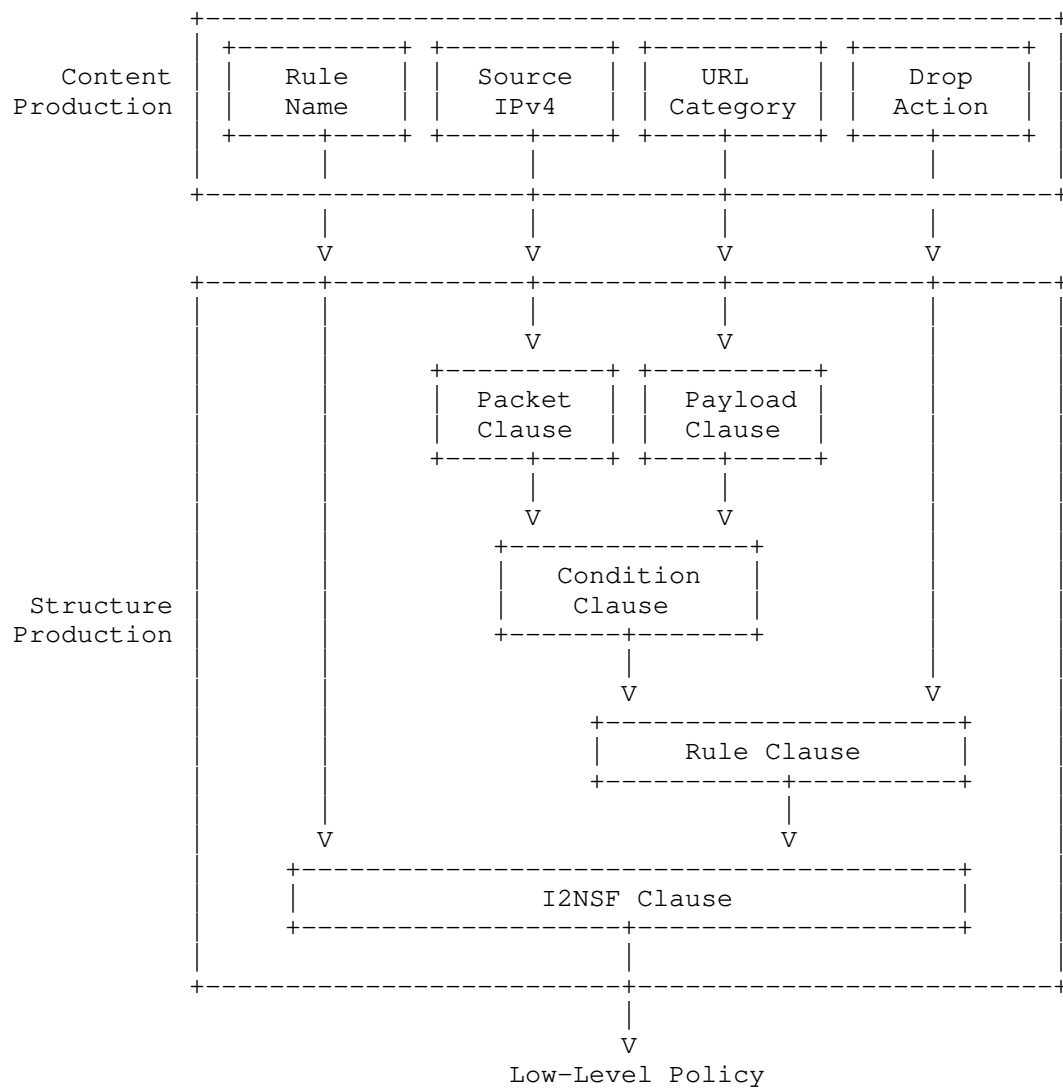


Figure 7: Generator Construction for Web-Filter NSF

```
<I2NSF>
  <rule-name>block_web</rule-name>
  <rules>
    <condition>
      <packet>
        <ipv4>10.0.0.1</ipv4>
        <ipv4>10.0.0.3</ipv4>
      </packet>
      <payload>
        <url>harm.com</url>
        <url>illegal.com</url>
      </payload>
    </condition>
    <action>drop</action>
  </rules>
</I2NSF>
```

Figure 8: Example of Low-Level Policy

5. Implementation Considerations

The implementation considerations in this document include the following three: "data model auto-adaptation", "data conversion", and "policy provisioning".

5.1. Data Model Auto-adaptation

Security Controller which acts as the intermediary MUST process the data according to the data model of the connected interfaces. However, the data model can be changed flexibly depending on the situation, and Security Controller may adapt to the change. Therefore, Security Controller can be implemented for convenience so that the security policy translator can easily adapt to the change of the data model.

The translator constructs and uses the DFA to adapt to Consumer-Facing Interface Data Model. In addition, the CFG is constructed and used to adapt to NSF-Facing Interface Data Model. Both the DFA and the CFG follow the same tree structure of YANG Data Model.

The DFA starts at the a node and expands operations by changing the state according to the input. Based on the YANG Data Model, a container node is defined as a middle state and a leaf node is defined as an extractor node. After that, if the nodes are connected in the same way as the hierarchical structure of the data model, Security Controller can automatically construct the DFA. The DFA can be conveniently built by investigating the link structure using the stack, starting with the root node.

The CFG starts at the leaf nodes and is grouped into clauses until all the nodes are merged into one node. A leaf node is defined as the content production, and a container node is defined as the structure production. After that, if the nodes are connected in the same way as the hierarchy of the data model, Security Controller can automatically construct the CFG. The CFG can be conveniently constructed by investigating the link structure using the priority queue data, starting with the leaf nodes.

5.2. Data Conversion

Security Controller requires the ability to materialize the abstract data in the high-level security policy and forward it to NSFs. Security Controller can receive endpoint information as keywords through the high-level security policy. At this time, if the endpoint information corresponding to the keyword is mapped and the query is transmitted to the NSF Database, the NSF Database can be conveniently registered with necessary information for data conversion. When a policy tries to establish a policy through the keyword, Security Controller searches the details corresponding to the keyword registered in the NSF Database and converts the keywords into the appropriate and specified data.

5.3. Policy Provisioning

This document stated that policy provisioning function is necessary to enable users without expert security knowledge to create policies. Policy provisioning is determined by the capability of the NSF. If the NSF has information about the capability in the policy, the probability of selection increases.

Most importantly, selected NSFs may be able to perform all capabilities in the security policy. This document recommends a study of policy provisioning algorithms that are highly efficient and can satisfy all capabilities in the security policy.

6. Features of Policy Translator Design

First, by showing a visualized translator structure, the security manager can handle various policy changes. Translator can be shown by visualizing DFA and Context-free Grammar so that the manager can easily understand the structure of Policy Translator.

Second, if I2NSF User only keeps the hierarchy of the data model, I2NSF User can freely create high-level policies. In the case of DFA, data extraction can be performed in the same way even if the order of input is changed. The design of the policy translator is

more flexible than the existing method that works by keeping the tag's position and order exactly.

Third, the structure of Policy Translator can be updated even while Policy Translator is operating. Because Policy Translator is modularized, the translator can adapt to changes in the NSF capability while the I2NSF framework is running. The function of changing the translator's structure can be provided through Registration Interface.

7. Security Considerations

There is no security concern in the proposed security policy translator as long as the I2NSF interfaces (i.e., Consumer-Facing Interface, NSF-Facing Interface, and Registration Interface) are protected by secure communication channels.

8. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (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).

This work was supported in part by the MSIT under the ITRC (Information Technology Research Center) support program (IITP-2018-2017-0-01633) supervised by the IITP.

9. References

9.1. Normative References

- [Automata] Peter, L., "Formal Languages and Automata, 6th Edition", January 2016.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.
- [XML] W3C, "On Views and XML (Extensible Markup Language)", June 1999.

9.2. Informative References

- [consumer-facing-inf-dm]
Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares,
"I2NSF Consumer-Facing Interface YANG Data Model", draft-
ietf-i2nsf-consumer-facing-interface-dm-03 (work in
progress), March 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-07 (work in
progress), July 2019.
- [nsf-facing-inf-dm]
Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin,
"I2NSF Network Security Function-Facing Interface YANG
Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-03
(work in progress), March 2019.
- [registration-inf-dm]
Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF
Registration Interface YANG Data Model", draft-ietf-i2nsf-
registration-interface-dm-02 (work in progress), March
2019.
- [XSLT] W3C, "Extensible Stylesheet Language Transformations
(XSLT) Version 1.0", November 1999.

Appendix A. Changes from draft-yang-i2nsf-security-policy-translation-02

The following changes are made from draft-yang-i2nsf-security-policy-translation-02:

- o Section 4.3.2 is added for describing 'NSF Database'. This section reinforces the ambiguous description of the NSF Database.
- o Section 5 is added for describing 'Implementation Considerations'. This section provides guidelines for a convenient implementation of security policy translator.

Authors' Addresses

Jinhyuk Yang
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 8520 8039
EMail: jin.hyuk@skku.edu

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

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

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

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