

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2019

W. Pan
L. Xia
Huawei
October 15, 2018

Configuration of Advanced Security Functions with I2NSF Security
Controller
draft-dong-i2nsf-asf-config-01

Abstract

This draft defines a network security function (NSF-) facing interface of the security controller for the purpose of configuring some advanced security functions. These advanced security functions include antivirus, anti-ddos, and intrusion prevention system (IPS). The interface is presented in a YANG data model fashion and can be used to deploy a large amount of NSF blocks that all support above mentioned functions in the software defined network (SDN) based paradigm.

Status of This Memo

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

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

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

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

Copyright Notice

Copyright (c) 2018 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
2.1. Key Words	3
2.2. Definition of Terms	3
3. Tree Diagrams	3
4. Data Model Structure	3
4.1. Antivirus	3
4.2. Anti-ddos	4
4.3. Intrusion prevention system	6
5. YANG Modules	7
5.1. Antivirus	7
5.2. Anti-ddos	13
5.3. Intrusion prevention system	20
6. IANA Considerations	26
7. Security Considerations	26
8. Acknowledgements	26
9. References	26
9.1. Normative References	26
9.2. Informative References	26
Authors' Addresses	27

1. Introduction

I2NSF provides a technology and vendor independent way for a centralized security controller in a SDN environment to manage and configure the distributed NSFs [RFC8329]. The NSFs are automatically customized in a programmable manner via a standard interface. In the draft [I-D.ietf-i2nsf-nsf-facing-interface-dm], it proposed a generic NSF-facing interface to manage which action should be applied on which traffic. In addition, there is another draft that defined the NSF-facing interface for management, including configuration and monitoring, of IPsec SAs [I-D.ietf-i2nsf-sdn-ipsec-flow-protection]. In this document, we defined another NSF-facing interface for security controller to configure some advanced security functions including the antivirus, anti-ddos, and IPS profiles. With the variety and complexity of the advanced security functions, it is hardly to define all the interfaces to configure each advanced security function. The antivirus, anti-ddos and IPS profiles, these three functions are the most common and well-developed advanced security functions and have been widely used. Standardizing the interface of these three functions can minimize the cost of

management and configuration of the security controller with a vendor independent way.

2. Terminology

2.1. Key Words

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

2.2. Definition of Terms

This document uses the terms defined in [I-D.ietf-i2nsf-terminology].

3. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams 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. Data Model Structure

4.1. Antivirus

The following tree diagram shows the interface for configuring antivirus detections on incoming and outgoing files. The file transfer protocol type, direction of file transfer, and the action applied on the detected virus are able to be configured. In addition, this interface also supports to configure the application and signature exception features to apply specific actions on certain applications and detected virus respectively. The anti-virus also supports to configure a whitelist for trusted files.

```

module: ietf-i2nsf-asf-config-antivirus
+--rw antivirus
  +--rw profiles
    +--rw profile* [name]
      +--rw name string
      +--rw description? string
      +--rw detect* [protocol-type direction]
        | +--rw protocol-type detect-protocol
        | +--rw direction detect-direction
        | +--rw action? detect-action
      +--rw exception-application* [application-name]
        | +--rw application-name string
        | +--rw application-action? detect-action
      +--rw exception-signature* [signature-id]
        | +--rw signature-id uint64
        | +--rw signature-action? detect-action
      +--rw whitelists {antivirus-whitelists}?
        +--rw match-rules
          | +--rw match-rule* [scope type value]
          |   +--rw scope match-scope
          |   +--rw type match-type
          |   +--rw value string
          +--rw source-address* inet:ip-address
          +--rw source-address-range*
            [start-address end-address]
          | +--rw start-address inet:ip-address
          | +--rw end-address inet:ip-address
          +--rw destination-address* inet:ip-address
          +--rw destination-address-range*
            [start-address end-address]
          +--rw start-address inet:ip-address
          +--rw end-address inet:ip-address

```

4.2. Anti-ddos

The following tree diagram shows the configuration parameters of DDoS detection and prevention functions of different types of DDoS attacks.

* SYN flood: The total number of packets that have the same destination address are counted in a period of time. If the counted packets number exceeds a pre-defined threshold, the prevention function is triggered. The anti-ddos system will alert the user/administrator, and start up source address inspection or TCP proxy function as configured.

* UPD flood: The UDP flood packets normally have the same payload or the payload changes regularly. The anti-ddos system is able to

automatically learn this payload characteristics, which is so called fingerprint of the UDP flood attack packets. And then if a packet matches the learned fingerprint, it will be discarded. For some UDP flood attack that does not has a fingerprint, a threshold bandwidth will be configured to limit the UDP traffic. If the UDP packet is associated with some TCP packets, the anti-ddos system can trigger the TCP protection measures and use the generated white list to determine whether to discard the UDP packets.

* HTTP and HTTPS flood: The detection mechanisms for these two attacks are similar to SYN flood detection. The total number of packets that have the same destination address are counted in a period of time. A threshold is set for the purpose of alerting.

* DNS request flood: The anti-ddos system counts the number of DNS request packets that have the same destination address in a period of time. Once this number exceeds a configured threshold, the prevention function is triggered. The anti-ddos system sends a response to the client to ask for another request with a TCP connection, and then verify the source address.

* DNS reply flood: The anti-ddos system counts the number of DNS reply packets that have the same destination address in a period of time. Once this number exceeds a configured threshold, the source address inspection is triggered. The anti-ddos ask the sender to send the reply message again with a new query ID and port number. If the second reply message is received and the query ID and port number match with the asked one. This source address will be added into the white list.

* ICMP flood: A threshold is configured to limit the rate of ICMP traffic.

* SIP flood: The anti-ddos system counts the number of SIP request packets that have the same destination address in a period of time. If the counted packets number exceeds a pre-defined threshold, the source authentication is triggered. The anti-ddos system sends an OPTIONS request packet with a specific branch value to verify whether the source address exists. If the reply message is in response to the OPTIONS packet, this source address will be added into the white list.

```

module: ietf-i2nsf-asf-config-antiddos
+--rw antiddos
  +--rw profiles
    +--rw profile* [name]
      +--rw name                string
      +--rw description?        string
      +--rw syn-flood* [action]
        | +--rw action          syn-flood-action
        | +--rw alert-rate?     uint32
      +--rw udp-flood* [action]
        | +--rw action          udp-flood-action
        | +--rw alert-rate?     uint32
      +--rw http-flood* [action]
        | +--rw action          http-flood-action
        | +--rw alert-rate?     uint32
      +--rw https-flood* [action]
        | +--rw action          https-flood-action
        | +--rw alert-rate?     uint32
      +--rw dns-request-flood* [action]
        | +--rw action          dns-request-flood-action
        | +--rw alert-rate?     uint32
      +--rw dns-reply-flood* [action]
        | +--rw action          dns-reply-flood-action
        | +--rw alert-rate?     uint32
      +--rw icmp-flood * [action]
        | +--rw action          icmp-flood-action
        | +--rw alert-rate?     uint32
      +--rw sip-flood* [action]
        | +--rw action          sip-flood-action
        | +--rw alert-rate?     uint32
      +--rw detect-mode?        enumeration
      +--rw baseline-learn
        +--rw auto-apply?       boolean
        +--rw start?            boolean
        +--rw mode?             enumeration
        +--rw tolerance-value?  uint16
        +--rw learn-duration?   uint32
        +--rw learn-interval?   uint32

```

4.3. Intrusion prevention system

The following tree diagram shows the interface for configuring the IPS. This interface supports to configure a set of IPS signature-based filters to detect known type of attacks and to respond with user defined actions such as sending an alert or block the matched packets.

```

module: ietf-i2nsf-asf-config-ips
  +--rw ips
    +--rw profiles
      +--rw profile* [name]
        +--rw name string
        +--rw description? string
      +--rw signature-sets
        | +--rw signature-set* [name]
        |   +--rw name string
        |   +--rw action? action-type
        |   +--rw application
        |   | +--rw all-application boolean
        |   | +--rw specified-application* string
        |   +--rw target? target-type
        |   +--rw severity* severity-type
        |   +--rw operating-system* operating-system-type
        |   +--rw protocol
        |   | +--rw all-protocol boolean
        |   | +--rw specified-protocol* string
        |   +--rw category
        |   | +--rw all-category boolean
        |   | +--rw specified-category* [name]
        |   |   +--rw name string
        |   |   +--rw all-sub-category boolean
        |   |   +--rw sub-category* [name]
        |   |     +--rw name string
        |   +--rw exception-signatures
        |   | +--rw exception-signature* [id]
        |   |   +--rw id uint32
        |   |   +--rw action? action-type

```

5. YANG Modules

5.1. Antivirus

```

module ietf-i2nsf-asf-config-antivirus {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-asf-config-antivirus";
  prefix
    asf-config-antivirus;

  import ietf-inet-types{
    prefix inet;
  }

  organization
    "Huawei Technologies";

```

```
contact
  "Wei Pan: william.panwei@huawei.com
  Liang Xia: Frank.xialiang@huawei.com";

description
  "This module contains a collection of yang definitions
  for configuring antivirus.";

revision 2018-10-15 {
  description
    "Init revision.";
  reference "xxx.";
}

typedef detect-protocol {
  type enumeration {
    enum http {
      description "HTTP.";
    }
    enum ftp {
      description "FTP.";
    }
    enum smtp {
      description "SMTP.";
    }
    enum pop3 {
      description "POP3.";
    }
    enum imap {
      description "IMAP.";
    }
    enum nfs {
      description "NFS.";
    }
    enum smb {
      description "SMB.";
    }
  }
  description
    "This is detect protocol type in antivirus profile.";
}

typedef detect-direction {
  type enumeration {
    enum none {
      description "None.";
    }
    enum download {
```

```
        description "Download.";
    }
    enum upload {
        description "Upload.";
    }
    enum both {
        description "Both directions.";
    }
}
description
    "This is detect direction type in antivirus profile.";
}

typedef detect-action {
    type enumeration {
        enum alert {
            description "Permit files and generate virus logs.";
        }
        enum allow {
            description "Permit files.";
        }
        enum block {
            description "Block files and generate virus logs.";
        }
        enum declare {
            description
                "Permit virus-infected email messages, then add information to
                announce the detection of viruses and generate virus logs.";
        }
        enum delete-attachment {
            description
                "Permit virus-infected email messages with deleting there
                attachments, add information to announce the detection of
                viruses and generate virus logs.";
        }
    }
}
description
    "This is detect action type in antivirus profile.";
}

typedef match-scope {
    type enumeration {
        enum url {
            description "URL.";
        }
        enum host {
            description "Host.";
        }
    }
}
```

```
    enum referer {
        description "Referer.";
    }
}
description "This is antivirus whitelist match scope.";
}

typedef match-type {
    type enumeration {
        enum prefix {
            description "Prefix.";
        }
        enum suffix {
            description "Suffix.";
        }
        enum fuzzy {
            description "Fuzzy.";
        }
        enum exact {
            description "Exact.";
        }
    }
}
description "This is antivirus whitelist match type.";
}

feature antivirus-whitelists {
    description
        "This feature means the antivirus function supports
        whitelists.";
}

grouping address-range {
    description "Address range.";
    leaf start-address {
        type inet:ip-address;
        description
            "Start address.";
    }

    leaf end-address {
        type inet:ip-address;
        description
            "End address.";
    }
}

container antivirus {
    description "Antivirus.";
}
```

```
container profiles {
  description "Profiles.";
  list profile {
    key "name";
    description "Antivirus profile.";

    leaf name {
      type string;
      description "The name of the profile.";
    }

    leaf description {
      type string;
      description "The description of the profile.";
    }

    list detect {
      key "protocol-type direction";
      description "Antivirus detect.";

      leaf protocol-type {
        type detect-protocol;
        description "The protocol type of detect.";
      }

      leaf direction {
        type detect-direction;
        description "The direction of detect.";
      }

      leaf action {
        type detect-action;
        description "The action of detect.";
      }
    }

    list exception-application {
      key "application-name";
      description "Exceptional application.";

      leaf application-name {
        type string;
        description "The name of exceptional application.";
      }

      leaf application-action {
        type detect-action;
        description "The action of exceptional application.";
      }
    }
  }
}
```

```
    }
  }

  list exception-signature {
    key "signature-id";
    description "Exceptional signature.";

    leaf signature-id {
      type uint64;
      description "The exception id of antivirus signature.";
    }

    leaf signature-action {
      type detect-action;
      description "The action of exceptional signature.";
    }
  }

  container whitelists {
    if-feature antivirus-whitelists;
    description "The whitelist of antivirus.";

    container match-rules {
      description "The match rules of antivirus whitelist.";

      list match-rule {
        key "scope type value";
        description "The match rule of antivirus whitelist.";

        leaf scope {
          type match-scope;
          description
            "The scope of antivirus whitelist match rule.";
        }

        leaf type {
          type match-type;
          description
            "The type of antivirus whitelist match rule.";
        }

        leaf value {
          type string;
          description
            "The value of antivirus whitelist match rule.";
        }
      }
    }
  }
}
```

```
    leaf-list source-address {
      type inet:ip-address;
      description "The source-address of whitelist.";
    }

    list source-address-range {
      key "start-address end-address";
      description "The source-address range of whitelist.";
      uses address-range;
    }

    leaf-list destination-address {
      type inet:ip-address;
      description "The destination-address of whitelist.";
    }

    list destination-address-range {
      key "start-address end-address";
      description "The destination-address range of whitelist.";
      uses address-range;
    }
  }
}
}
```

5.2. Anti-ddos

```
module ietf-i2nsf-asf-config-antiddos {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-asf-config-antiddos";
  prefix
    asf-config-antiddos;

  organization
    "Huawei Technologies";

  contact
    "Wei Pan: william.panwei@huawei.com
     Liang Xia: Frank.xialiang@huawei.com";

  description
    "This module contains a collection of yang definitions
     for configuring anti-ddos.";

  revision 2018-10-15 {
```

```
description
  "Init revision.";
reference "xxx.";
}

typedef syn-flood-action {
  type enumeration {
    enum tcp-proxy {
      description
        "TCP proxy function.";
    }
    enum tcp-source-authentication {
      description
        "Authenticate the source addresses of TCP packets.";
    }
  }
  description
    "This is detect action type of syn-flood.";
}

typedef udp-flood-action {
  type enumeration {
    enum fingerprint-learning {
      description
        "Learn the fingerprint of UDP packets.";
    }
    enum udp-tcp-association {
      description
        "Authenticate the source addresses of TCP packets
        associated with UDP packets.";
    }
    enum traffic-limit {
      description
        "Limit the UDP traffic.";
    }
  }
  description
    "This is detect action type of udp-flood.";
}

typedef http-flood-action {
  type enumeration {
    enum source-authentication-meta-refresh {
      description
        "Authenticate the source addresses of HTTP packets by a way of
        meta-refresh.";
    }
    enum source-authentication-code-based {
```

```
        description
            "Authenticate the source addresses of HTTP packets by a way of
            code-based.";
    }
    enum source-authentication-302-redirect {
        description
            "Authenticate the source addresses of HTTP packets by a way of
            302-redirect.";
    }
}
description
    "This is detect action type of http-flood.";
}

typedef https-flood-action {
    type enumeration {
        enum source-authentication {
            description
                "Authenticate the source addresses of HTTPS packets.";
        }
    }
    description
        "This is detect action type of https-flood.";
}

typedef dns-request-flood-action {
    type enumeration {
        enum source-authentication-dns-cache-server {
            description
                "Authenticate the source addresses of DNS request packets for
                the DNS Cache Server.";
        }
        enum source-authentication-dns-authoritative-server {
            description
                "Authenticate the source addresses of DNS request packets for
                the DNS Authoritative Server.";
        }
    }
    description
        "This is detect action type of dns-request-flood.";
}

typedef dns-reply-flood-action {
    type enumeration {
        enum source-authentication {
            description
                "Authenticate the source addresses of DNS reply packets.";
        }
    }
}
```

```
    }
    description
      "This is detect action type of dns-reply-flood.";
  }

typedef icmp-flood-action {
  type enumeration {
    enum traffic-limit {
      description
        "Limit the ICMP traffic.";
    }
  }
  description
    "This is detect action type of icmp-flood.";
}

typedef sip-flood-action {
  type enumeration {
    enum source-authentication {
      description
        "Authenticate the source addresses of SIP packets.";
    }
  }
  description
    "This is detect action type of sip-flood.";
}

container antiddos {
  description "Anti-ddos.";
  container profiles {
    description "Profiles.";
    list profile {
      key "name";
      description "Anti-ddos profile.";

      leaf name {
        type string;
        description "The name of the profile.";
      }

      leaf description {
        type string;
        description "The description of the profile.";
      }
    }

    list syn-flood {
      key "action";
      description "SYN flood detect.";
    }
  }
}
```

```
    leaf action {
      type syn-flood-action;
      description "The action of syn-flood detect.";
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of syn-flood detect.";
    }
  }

list udp-flood {
  key "action";
  description "UDP flood detect.";

  leaf action {
    type udp-flood-action;
    description "The action of udp-flood detect.";
  }

  leaf alert-rate {
    type uint32;
    description "The alert rate of udp-flood detect.";
  }
}

list http-flood {
  key "action";
  description "HTTP flood detect.";

  leaf action {
    type http-flood-action;
    description "The action of http-flood detect.";
  }

  leaf alert-rate {
    type uint32;
    description "The alert rate of http-flood detect.";
  }
}

list https-flood {
  key "action";
  description "HTTPS flood detect.";

  leaf action {
    type https-flood-action;
    description "The action of https-flood detect.";
  }
}
```

```
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of https-flood detect.";
    }
  }

  list dns-request-flood {
    key "action";
    description "DNS request flood detect.";

    leaf action {
      type dns-request-flood-action;
      description "The action of dns-request-flood detect.";
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of dns-request-flood detect.";
    }
  }

  list dns-reply-flood {
    key "action";
    description "DNS reply flood detect.";

    leaf action {
      type dns-reply-flood-action;
      description "The action of dns-reply-flood detect.";
    }

    leaf alert-rate {
      type uint32;
      description "The alert rate of dns-reply-flood detect.";
    }
  }

  list icmp-flood {
    key "action";
    description "ICMP flood detect.";

    leaf action {
      type icmp-flood-action;
      description "The action of icmp-flood detect.";
    }

    leaf alert-rate {
```

```
        type uint32;
        description "The alert rate of icmp-flood detect.";
    }
}

list sip-flood {
    key "action";
    description "SIP flood detect.";

    leaf action {
        type sip-flood-action;
        description "The action of sip-flood detect.";
    }

    leaf alert-rate {
        type uint32;
        description "The alert rate of sip-flood detect.";
    }
}

leaf detect-mode {
    type enumeration {
        enum detect-clean {
            description
                "Detect DDoS attacks and defend against them.";
        }

        enum detect-only{
            description
                "Detect DDoS attacks only.";
        }
    }
    description "DDoS detect mode.";
}

container baseline-learn {
    description "Alert rate baseline learning.";

    leaf auto-apply {
        type boolean;
        description "Apply baseline learning results.";
    }

    leaf start {
        type boolean;
        description "Enable baseline learning.";
    }
}
```

```
    leaf mode {
      type enumeration {
        enum loop {
          description
            "Indicate that baseline learning is performed
            periodically.";
        }

        enum once {
          description
            "Indicate that baseline learning is performed once.";
        }
      }
      description "Indicate the baseline learning mode.";
    }

    leaf tolerance-value {
      type uint16;
      description
        "Indicate the baseline learning tolerance
        value.";
    }

    leaf learn-duration {
      type uint32;
      description "Indicate the baseline learning duration.";
    }

    leaf learn-interval {
      type uint32;
      description "Indicate the interval for baseline learning.";
    }
  }
}
}
```

5.3. Intrusion prevention system

```
module ietf-i2nsf-asf-config-ips {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-asf-config-ips";
  prefix
    asf-config-ips;

  organization
```

```
"Huawei Technologies";

contact
  "Wei Pan: william.panwei@huawei.com
  Liang Xia: Frank.xialiang@huawei.com";

description
  "This module contains a collection of yang definitions for
  configuring ips.";

revision 2018-10-15 {
  description
    "Init revision.";
  reference "xxx.";
}

typedef action-type {
  type enumeration {
    enum default-type {
      description "Default action type.";
    }
    enum alert {
      description "Alert.";
    }
    enum block {
      description "Block.";
    }
    enum allow {
      description "Allow.";
    }
  }
  description "The action type.";
}

typedef target-type {
  type enumeration {
    enum both {
      description "Both client and server.";
    }
    enum client {
      description "Client.";
    }
    enum server {
      description "Server.";
    }
  }
  description "The target type.";
}
```

```
typedef severity-type {
  type enumeration {
    enum high {
      description "High.";
    }
    enum medium {
      description "Medium.";
    }
    enum low {
      description "Low.";
    }
    enum information {
      description "Information.";
    }
  }
  description "The severity filter type.";
}

typedef operating-system-type {
  type enumeration {
    enum android {
      description "Android OS.";
    }
    enum ios {
      description "IOS.";
    }
    enum unix-like {
      description "UNIX-like OS.";
    }
    enum windows {
      description "Windows OS.";
    }
    enum other {
      description "Other OS.";
    }
  }
  description "The operating system type.";
}

container ips {
  description "Intrusion prevention system.";
  container profiles {
    description "Profiles.";
    list profile {
      key "name";
      description "IPS Profile.";

      leaf name {
```

```
    type string;
    description "The name of a profile.";
}

leaf description {
    type string;
    description "The description of a profile.";
}

container signature-sets {
    description "Signature sets.";
    list signature-set {
        key "name";
        description "Signature set.";

        leaf name {
            type string;
            description "The name of a signature set.";
        }

        leaf action {
            type action-type;
            description "The action for a signature set.";
        }

        container application {
            description "Application.";
            leaf all-application {
                type boolean;
                mandatory true;
                description
                    "The all application filtering conditions of the
                    signature set.";
            }

            leaf-list specified-application {
                when "../all-application = 'false'";
                type string;
                description
                    "The specified application filtering conditions of the
                    signature set.";
            }
        }

        leaf target {
            type target-type;
            description
                "The target type of a signature set.";
        }
    }
}
```

```
    }

    leaf-list severity {
      type severity-type;
      description
        "The severity type of a signature set.";
    }

    leaf-list operating-system {
      type operating-system-type;
      description
        "The operating system of a signature set.";
    }

    container protocol {
      description "Protocol.";
      leaf all-protocol {
        type boolean;
        mandatory true;
        description
          "The all protocol filtering conditions of a
          signature set.";
      }

      leaf-list specified-protocol {
        when "../all-protocol = 'false'";
        type string;
        description
          "The specified protocol filtering conditions of a
          signature set.";
      }
    }

    container category {
      description "Category.";
      leaf all-category {
        type boolean;
        mandatory true;
        description
          "The all category filtering conditions of t signature
          set.";
      }

      list specified-category {
        when "../all-category = 'false'";
        key "name";
        description "Specified category.";
      }
    }
  }
}
```

```
leaf name {
    type string;
    description
        "The specified name of category
        filtering conditions of a signature set.";
}

leaf all-sub-category {
    type boolean;
    mandatory true;
    description
        "The all sub-category filtering
        conditions of a signature set.";
}

list sub-category {
    when "../all-sub-category = 'false'";
    key "name";
    description "Sub category.";

    leaf name {
        type string;
        description
            "The specified name of sub-category filtering
            conditions of a signature set.";
    }
}

}
}
}

container exception-signatures {
    description "Exceptional signatures.";
    list exception-signature {
        key "id";
        description "Exceptional signature.";

        leaf id {
            type uint32;
            description "The ID of an exception signature.";
        }

        leaf action {
            type action-type;
            description
                "This action type of an exception signature.";
        }
    }
}
```

```
    }  
  }  
}  
}
```

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

TBD.

8. Acknowledgements

TBD

9. References

9.1. 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>>.

9.2. Informative References

[I-D.ietf-i2nsf-nsf-facing-interface-dm]
Kim, J., Jeong, J., Jung-Soo, P., Hares, S., and l. linqiushi@huawei.com, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-dm-00 (work in progress), March 2018.

[I-D.ietf-i2nsf-sdn-ipsec-flow-protection]
Lopez, R. and G. Lopez-Millan, "Software-Defined Networking (SDN)-based IPsec Flow Protection", draft-ietf-i2nsf-sdn-ipsec-flow-protection-01 (work in progress), March 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-05 (work in progress), January 2018.

[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>>.

Authors' Addresses

Wei Pan
Huawei

Email: william.panwei@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

D. Hong
J. Jeong
J. Kim
Sungkyunkwan University
S. Hares
L. Xia
Huawei
H. Birkholz
Fraunhofer SIT
October 22, 2018

YANG Data Model for Monitoring I2NSF Network Security Functions
draft-hong-i2nsf-nsf-monitoring-data-model-05

Abstract

This document proposes a 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 a data tree to specify an information model for monitoring NSFs, but also the corresponding YANG data model for monitoring 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 April 25, 2019.

Copyright Notice

Copyright (c) 2018 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 2
3. Terminology 3
3.1. Tree Diagrams 3
4. NSF Monitoring Management 3
5. Information Model Structure 4
6. YANG Data Model 12
7. Acknowledgments 46
8. References 46
8.1. Normative References 46
8.2. Informative References 46
Appendix A. Changes from draft-hong-i2nsf-nsf-monitoring-data-
model-04 48
Authors' Addresses 48

1. Introduction

This document defines a YANG [RFC6020] data model for monitoring Network Security Functions (NSFs). The monitoring means the aquisition of vital information about NSFs via notifications, records or counters. The data model for the monitoring presented in this document is derived from the information model for monitoring NSFs through the NSF-Facing Interface specified in [i2nsf-monitoring-im].

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-terminology][i2nsf-framework]. Especially, the following terms are from [i2nsf-monitoring-im].

- 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.
- 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.

3.1. Tree Diagrams

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

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

4. NSF Monitoring Management

A standard model for monitoring data is required for the administrator to check the monitoring data generated by 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 so that the administrator can know the state of the framework. In order to communicate with other component, it requires the interfaces. The three main interfaces in I2NSF framework are used for sending monitoring data like as follows:

- o I2NSF Consumer-Facing Interface : When an I2NSF user makes a security policy and forwards it to the security controller through the 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 NSF.
- o I2NSF Registration Interface : The NFV architecture provides VNF lifecycle management through the Ve-Vnfm interface. The role of Ve-Vnfm is to request VNF lifecycle management, exchange configuration information, and exchange status information for network service lifecycle management. For I2NSF framework case, the DMS manages data such as resource state and so on for lifecycle of NSF. Therefore the generated monitoring data need to be delivered to Vn-Vnfm via DMS in the environment that can support VNF.
- o I2NSF NSF-Facing Interface : After the policy is translated by security policy translator in security controller, the translated security policy(low-level) is applied to NSF via NSF-Facing Interface. The monitoring data model specifies the list of event that can trigger the ECA(Event-Condition-Action) policy at NSF-Facing Interface.

5. Information Model Structure

Figure 1 shows the overview of a structure tree of monitoring information based on the [i2nsf-monitoring-im].

```

module: ietf-i2nsf-nsf-monitoring-dm
  +--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

6. YANG Data Model

This section introduces a YANG data model for the information model of monitoring information based on [i2nsf-monitoring-im].

```

<CODE BEGINS> file "ietf-i2nsf-nsf-monitoring-dm@2018-10-22.yang"
module ietf-i2nsf-nsf-monitoring-dm {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-i2nsf-nsf-monitoring-dm";
  prefix
    monitoring-information;
  import ietf-inet-types{
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  organization
    "IETF I2NSF (Interface to Network Security Functions)

```

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

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

  Editor: Dongjin Hong
  <mailto:dong.jin@skku.edu>

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

description
  "This module defines a YANG data module for monitoring NSFs.";

revision "2018-07-02" {
  description "Fifth revision";
  reference
    "draft-zhang-i2nsf-info-model-monitoring-06";
}

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 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 associated with
       a detected system-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.";
}
identity udp {
    base ipv4;
```

```
    base ipv6;
    description
      "UDP protocol type.";
  }
  identity icmp {
    base ipv4;
    base ipv6;
    description
      "General ICMP protocol type.";
  }
  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.";
  }
  identity ipv4 {
    base ip;
    description
      "IPv4 protocol type.";
  }
  identity ipv6 {
    base ip;
    description
      "IPv6 protocol type.";
  }
  identity http {
    base tcp;
    description
      "HTTP protocol type.";
  }
  identity ftp {
    base tcp;
    description
      "FTP protocol type.";
  }
  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-catagory {
    type identityref {
      base alarm-type;
    }
    description
      "The alarm catagory 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-catagory {
    type identityref {
      base event-type;
    }
    description
      "The event catagory 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 deteced";
    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 notification 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

7. 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).

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

8.2. Informative References

[i2nsf-consumer-facing-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-01 (work in
progress), July 2018.

[i2nsf-framework]

Lopez,, D., Lopez,, E., Dunbar, L., Strassner, J., and R.
Kumar, "Framework for Interface to Network Security
Functions", RFC 8329, February 2018.

[i2nsf-monitoring-im]

Xia,, L., Zhang,, D., Wu, Y., Kumar, R., Lohiya, A., and
H. Birkholz, "An Information Model for the Monitoring of
Network Security Functions (NSF)", draft-zhang-i2nsf-info-
model-monitoring-06 (work in progress), May 2018.

[i2nsf-nsf-facing-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-01
(work in progress), July 2018.

[i2nsf-registration-im]

Hyun,, S., Jeong,, J., Roh, T., Wi, S., and J. Park,
"Registration Interface Information Model", draft-hyun-
i2nsf-registration-interface-im-06 (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-05
(work in progress), July 2018.

[i2rs-rib-data-model]

Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini,
S., and N. Bahadur, "A YANG Data Model for Routing
Information Base (RIB)", draft-ietf-i2rs-rib-data-model-10
(work in progress), February 2018.

Appendix A. Changes from draft-hong-i2nsf-nsf-monitoring-data-model-04

The following changes are made from draft-hong-i2nsf-nsf-monitoring-data-model-04:

1. The association with the three main interfaces of the I2NSF framework is described in Section 3.
2. Typos are corrected.

Authors' Addresses

Dongjin Hong
Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 10 7630 5473
EMail: dong.jin@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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

S. Hares
Huawei
J. Jeong
J. Kim
Sungkyunkwan University
R. Moskowitz
HTT Consulting
Q. Lin
Huawei
July 02, 2018

I2NSF Capability YANG Data Model
draft-ietf-i2nsf-capability-data-model-01

Abstract

This document defines a YANG data model for capabilities that enable an I2NSF user to control various Network Security Functions (NSFs) in the framework for 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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 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	4
3. Terminology	4
3.1. Tree Diagrams	4
4. Overview	4
5. The Structure and Objective of NSF Capabilities	6
5.1. Generic Network Security Function Identification	6
5.2. Event Capabilities	6
5.3. Condition Capabilities	7
5.4. Action Capabilities	7
5.5. Resolution Strategy Capabilities	7
5.6. Default Action Capabilities	7
5.7. RPC for Acquiring Appropriate Network Security Function	8
6. Data Model Structure	8
6.1. Network Security Function Identification	8
6.2. Capabilities of Generic Network Security Function	9
6.2.1. Event Capabilities	9
6.2.2. Condition Capabilities	11
6.2.3. Action Capabilities	14
6.2.4. Resolution Strategy Capabilities	15
6.2.5. Content Security Capabilities	15
6.2.6. Attack Mitigation Capabilities	16
6.2.7. RPC for Acquiring Appropriate Network Security Function	17
7. YANG Modules	18
7.1. I2NSF Capability YANG Data Module	18
8. IANA Considerations	52
9. Security Considerations	52
10. Acknowledgments	52
11. Contributors	53
12. References	53
12.1. Normative References	53
12.2. Informative References	53
Appendix A. Example: Extended VoIP-VoLTE Security Function Capabilities Module	55
Appendix B. Example: Configuration XML of Capability Module	56
B.1. Example: Configuration XML of Generic Network Security Function Capabilities	56
B.2. Example: Configuration XML of Extended VoIP/VoLTE Security Function Capabilities Module	58
Appendix C. Changes from draft-ietf-i2nsf-capability-data-	

model-01	59
Authors' Addresses	60

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 express capabilities of those security devices. This YANG data model is based on the information model for I2NSF NSF capabilities [i2nsf-nsf-cap-im]. 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]. After the capabilities of the NSFs are registered, this YANG data model can be used by the I2NSF user or Service Function Forwarder (SFF) [i2nsf-sfc] to acquire appropriate NSFs that can be controlled by the Network Operator Mgmt System.

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules. The "ietf-i2nsf-capability" YANG module defined in this document provides the following features:

- o Configuration of identification for generic network security function policy
- o Configuration of event capabilities for generic network security function policy
- o Configuration of condition capabilities for generic network security function policy
- o Configuration of action capabilities for generic network security function policy
- o Configuration of strategy capabilities for generic network security function policy
- o Configuration of default action capabilities for generic network security function policy
- o RPC for acquiring appropriate network security function according to type of NSF and/or target devices.

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-terminology][i2nsf-nsf-cap-im][i2rs-rib-data-model][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

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

3.1. Tree Diagrams

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

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

4. Overview

This section explains overview how the YANG data model can be used by I2NSF User, Developer's Mgmt System, and SFF. 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 device can support. To register NSFs in this way, the Developer's Mgmt System utilizes this standardized capabilities YANG data model through registration interface. Through this registration of capabilities, the a lot of problems [RFC8192] can be resolved. The following shows use cases.

Note [i2nsf-nsf-yang] is used to configure rules of NSFs in I2NSF Framework.

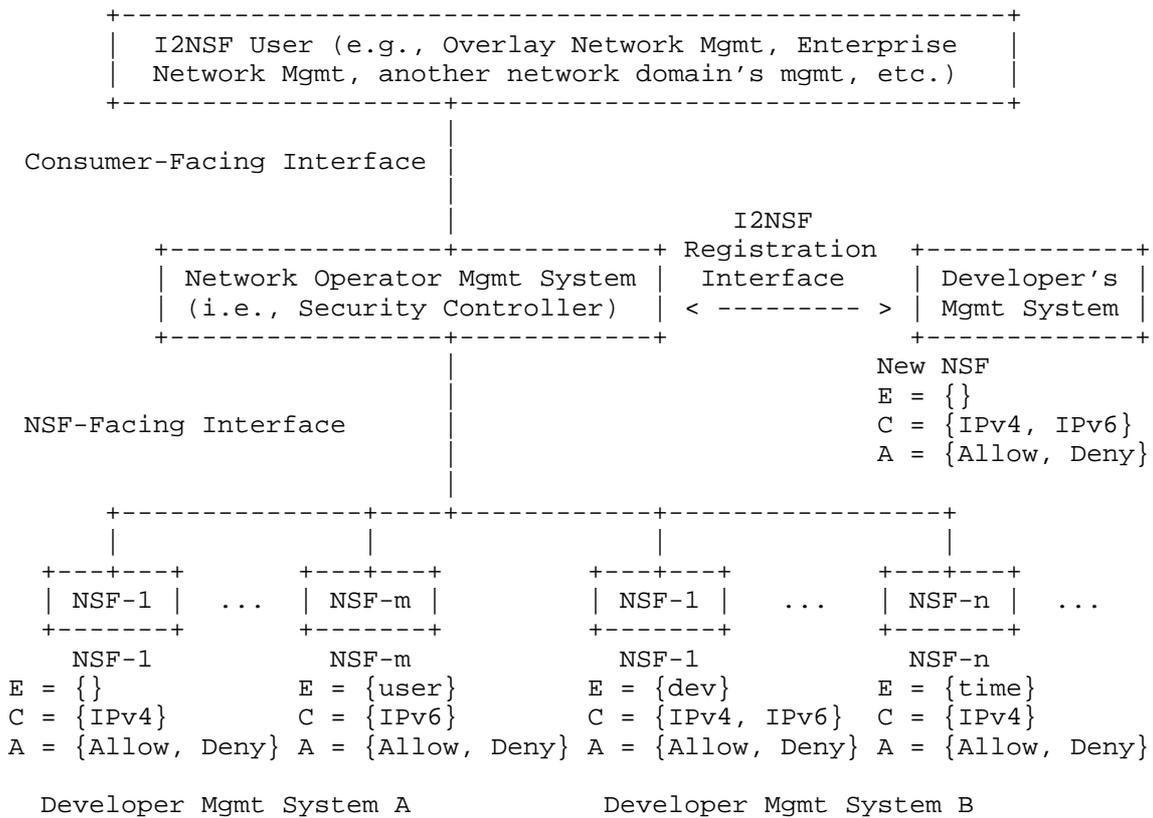


Figure 1: Capabilities of NSFs in I2NSF Framework

- o If I2NSF User wants to apply 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 standardizing the capabilities of NSFs. If I2NSF User wants to block malicious users with IPv6, I2NSF User sends the rules about blocking the users to Network Operator Mgmt System. When the Network Operator Mgmt System receives the rules, it sends that 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 I2NSF User to apply the rule about blocking the malicious packets to NSFs one by one. This problem can be resolved by standardizing the capabilities of NSFs. If NSFs find the malicious packets with IPv4, they can ask the Network Operator Mgmt System to alter specific rules and/or configurations. When the Network Operator Mgmt System receives the rules for malicious packets, it inspects whether the rules are reasonable and sends the rules 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 rules can be applied to appropriate NSFs without control of I2NSF User.
- o If NSFs of Service Function Chaining (SFC) [i2nsf-sfc] fail, it is a tremendous burden for I2NSF User to reconfigure the policy of SFC immediately. This problem can be resolved by periodically acquiring information of appropriate NSFs of SFC. If SFF needs information of Web Application Firewall for SFC, it can ask the Network Operator Mgmt System to acquire the location information of appropriate Web Application Firewall. When the Network Operator Mgmt System receives requested information from SFF, it sends location information of Web Application Firewall to the SFF. Therefore, the policy about the NSFs of SFC can be periodically updated without control of I2NSF User.

5. The Structure and Objective of NSF Capabilities

5.1. Generic Network Security Function Identification

This shows a identification for generic network security functions. These objects are defined as location information and target device information.

5.2. Event Capabilities

This shows a event capabilities for generic network security functions policy. This is used to specify capabilities about any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of I2NSF Policy Rules, it is used to determine whether the Condition clause of the I2NSF Policy Rule can be evaluated or not. These object of event capabilities is defined as user security event capabilities, device security event capabilities, system

security event capabilities, and time security event capabilities. These object of event capabilities can be extended according to specific vendor event features.

5.3. Condition Capabilities

This shows a condition capabilities for generic network security functions policy. This is used to specify capabilities about 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. These object of condition capabilities is defined as packet security condition capabilities, packet payload security condition capabilities, target security condition capabilities, user security condition capabilities, context condition capabilities, and generic context condition capabilities. These object of condition capabilities can be extended according to specific vendor condition features.

5.4. Action Capabilities

This shows a action capabilities for generic network security functions policy. This is used to specify capabilities 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. These object of action capabilities is defined as ingress action capabilities, egress action capabilities, and apply profile action capabilities. These object of action capabilities can be extended according to specific vendor action features.

5.5. Resolution Strategy Capabilities

This shows a resolution strategy capabilities for generic network security functions policy. This can be 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. These objects are defined as first-matching-rule capability and last-matching-rule capability. These objects can be extended according to specific vendor resolution strategy features.

5.6. Default Action Capabilities

This shows a default action policy for generic network security functions. This can be used to specify capabilities about a predefined action when no other alternative action was matched by the currently executing I2NSF Policy Rule.

5.7. RPC for Acquiring Appropriate Network Security Function

This shows a RPC for acquiring an appropriate network security function according to type of NSF and/or target devices. If the SFF [i2nsf-sfc] does not have the location information of network security functions that it should send in own cache table, this can be used to acquire the information. These objects are defined as input data (i.e., NSF type and target devices) and output data (i.e., location information of NSF).

6. Data Model Structure

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

6.1. Network Security Function Identification

The data model for network security function identification has the following structure:

```

module: ietf-i2nsf-capability
  +--rw nsf* [nsf-name]
    +--rw nsf-name                string
    +--rw nsf-type?               nsf-type
    +--rw nsf-address
      | +--rw (nsf-address-type)?
      |   +--:(ipv4-address)
      |     | +--rw ipv4-address    inet:ipv4-address
      |     +--:(ipv6-address)
      |       +--rw ipv6-address    inet:ipv6-address
    +--rw target-device
      | +--rw pc?                  boolean
      | +--rw mobile-phone?       boolean
      | +--rw voip-volte-phone?   boolean
      | +--rw tablet?             boolean
      | +--rw iot?                 boolean
      | +--rw vehicle?            boolean
    +--rw generic-nsf-capabilities
      | +--rw net-sec-capabilities
      |   uses net-sec-caps
    +--rw complete-nsf-capabilities
      | +--rw con-sec-control-capabilities
      |   | uses i2nsf-con-sec-control-caps
      | +--rw attack-mitigation-capabilities
      |   uses i2nsf-attack-mitigation-control-caps
  
```

Figure 2: Data Model Structure for NSF-Identification

This draft also utilizes the concepts originated in Basile, Liroy, Pitscheider, and Zhao[2015] concerning conflict resolution, use of external data, and target device. The authors are grateful to Cataldo for pointing out this excellent work.

The nsf-type object can be used for configuration about type of a NSF. The types of NSF consists of Network Firewall, Web Application Firewall, Anti-Virus, IDS, IPS, and DDoS Mitigator. The nsf-address object can be used for configuration about location of a NSF. The target-device object can be used for configuration about target devices. We will add additional type of a NSF for more generic network security functions.

6.2. Capabilities of Generic Network Security Function

The data model for Generic NSF capabilities has the following structure:

```

+--rw generic-nsf-capabilities
  +--rw net-sec-capabilities
    uses i2nsf-net-sec-caps
  
```

Figure 3: Data Model Structure for Capabilities of Network Security Function

6.2.1. Event Capabilities

The data model for event capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
  +--rw net-sec-capabilities
    +--rw time
      | +--rw time-zone
      | | +--rw time-zone-offset?  boolean
      | +--rw time-inteval
      | | +--rw absolute-time-inteval
      | | | +--rw start-time?  boolean
      | | | +--rw end-time?    boolean
      | | +--rw periodic-time-inteval
      | | | +--rw day?         boolean
      | | | +--rw month?      boolean
    +--rw event
      | +--rw usr-event
      | | +--rw usr-sec-event-content?  boolean
      | | +--rw usr-sec-event-format
      | | | +--rw unknown?  boolean
      | | | +--rw guid?     boolean
  
```



```

| | | |--rw sys-sec-event-type
| | | | |--rw unknown? boolean
| | | | |--rw audit-log-written-to? boolean
| | | | |--rw audit-log-cleared? boolean
| | | | |--rw policy-created? boolean
| | | | |--rw policy-edited? boolean
| | | | |--rw policy-deleted? boolean
| | | | |--rw policy-executed? boolean
| |--rw time-event
| | |--rw time-sec-event-begin? boolean
| | |--rw time-sec-event-end? boolean
| | |--rw time-sec-event-time-zone? boolean
+--rw condition
| ...
+--rw action
| ...
+--rw resolution-strategy
...

```

Figure 4: Data Model Structure for Event Capabilities of Network Security Function

These objects are defined as capabilities of user security event, device security event, system security event, and time security event. These objects can be extended according to specific vendor event features. We will add additional event objects for more generic network security functions.

6.2.2. Condition Capabilities

The data model for condition capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
  +--rw net-sec-capabilities
    +--rw time
      +--rw time-zone
        | |--rw time-zone-offset? boolean
      +--rw time-inteval
        +--rw absolute-time-inteval
          | |--rw start-time? boolean
          | |--rw end-time? boolean
        +--rw periodic-time-inteval
          +--rw day? boolean
          +--rw month? boolean
    
```

```

+--rw event
|
|  ...
+--rw condition
|
|  +--rw packet-security-condition
|  |
|  |  +--rw packet-security-mac-condition
|  |  |
|  |  |  +--rw pkt-sec-cond-mac-dest?          boolean
|  |  |  +--rw pkt-sec-cond-mac-src?          boolean
|  |  |  +--rw pkt-sec-cond-mac-8021q?       boolean
|  |  |  +--rw pkt-sec-cond-mac-ether-type?   boolean
|  |  |  +--rw pkt-sec-cond-mac-tci?         string
|  |  |
|  |  |  +--rw packet-security-ipv4-condition
|  |  |  |
|  |  |  |  +--rw pkt-sec-cond-ipv4-header-length?    boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-tos?              boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-total-length?     boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-id?               boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-fragment?         boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-fragment-offset?  boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-ttl?              boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-protocol?         boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-src?               boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-dest?              boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-ipopts?           boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-sameip?           boolean
|  |  |  |  +--rw pkt-sec-cond-ipv4-geoip?            boolean
|  |  |  |
|  |  |  |  +--rw packet-security-ipv6-condition
|  |  |  |  |
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-dscp?          boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-ecn?          boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-traffic-class?  boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-flow-label?    boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-payload-length? boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-next-header?   boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-hop-limit?    boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-src?           boolean
|  |  |  |  |  +--rw pkt-sec-cond-ipv6-dest?         boolean
|  |  |  |  |
|  |  |  |  |  +--rw packet-security-tcp-condition
|  |  |  |  |  |
|  |  |  |  |  |  +--rw pkt-sec-cond-tcp-src-port?    boolean
|  |  |  |  |  |  +--rw pkt-sec-cond-tcp-dest-port?   boolean
|  |  |  |  |  |  +--rw pkt-sec-cond-tcp-seq-num?     boolean
|  |  |  |  |  |  +--rw pkt-sec-cond-tcp-ack-num?     boolean
|  |  |  |  |  |  +--rw pkt-sec-cond-tcp-window-size?  boolean
|  |  |  |  |  |  +--rw pkt-sec-cond-tcp-flags?       boolean
|  |  |  |  |  |
|  |  |  |  |  |  +--rw packet-security-udp-condition
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +--rw pkt-sec-cond-udp-src-port?  boolean
|  |  |  |  |  |  |  +--rw pkt-sec-cond-udp-dest-port?  boolean
|  |  |  |  |  |  |  +--rw pkt-sec-cond-udp-length?    boolean
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +--rw packet-security-icmp-condition
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  +--rw pkt-sec-cond-icmp-type?  boolean
|  |  |  |  |  |  |  |  +--rw pkt-sec-cond-icmp-code?  boolean

```

```

|      +--rw pkt-sec-cond-icmp-seg-num?  boolean
+--rw packet-payload-condition
|   +--rw pkt-payload-content?  boolean
+--rw acl-number?                boolean
+--rw application-condition
|   +--rw application-object?  boolean
|   +--rw application-group?   boolean
|   +--rw application-label?   boolean
|   +--rw category
|       +--rw application-category?  boolean
+--rw target-condition
|   +--rw device-sec-context-cond?  boolean
+--rw users-condition
|   +--rw user
|       +--rw (user-name)?
|           +--:(tenant)
|               | +--rw tenant?  boolean
|               +--:(vn-id)
|                   +--rw vn-id?  boolean
|   +--rw group
|       +--rw (group-name)?
|           +--:(tenant)
|               | +--rw tenant?          boolean
|               +--:(vn-id)
|                   +--rw vn-id?        boolean
|       +--rw security-grup  boolean
+--rw url-category-condition
|   +--rw pre-defined-category?  boolean
|   +--rw user-defined-category?  boolean
+--rw context-condition
|   +--rw temp?  string
+--rw gen-context-condition
|   +--rw geographic-location
|       +--rw src-geographic-location?  boolean
|       +--rw dest-geographic-location?  boolean
+--rw action
|   ...
+--rw resolution-strategy
|   ...

```

Figure 5: Data Model Structure for Condition Capabilities of Network Security Function

These objects are defined as capabilities of packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. These objects can be extended according to

specific vendor condition features. We will add additional condition objects for more generic network security functions.

6.2.3. Action Capabilities

The data model for action capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
  +--rw net-sec-capabilities
    +--rw time
      +--rw time-zone
      |   +--rw time-zone-offset?  boolean
      +--rw time-inteval
        +--rw absolute-time-inteval
          +--rw start-time?  boolean
          +--rw end-time?    boolean
        +--rw periodic-time-inteval
          +--rw day?        boolean
          +--rw month?     boolean
    +--rw event
    |   ...
    +--rw condition
    |   ...
    +--rw action
      +--rw rule-log?          boolean
      +--rw session-log?      boolean
      +--rw ingress-action
      |   +--rw ingress-action-type
      |   |   +--rw pass?      boolean
      |   |   +--rw drop?     boolean
      |   |   +--rw reject?   boolean
      |   |   +--rw alert?    boolean
      |   |   +--rw mirror?   boolean
      +--rw egress-action
      |   +--rw egress-action-type
      |   |   +--rw invoke-signaling?  boolean
      |   |   +--rw tunnel-encapsulation?  boolean
      |   |   +--rw forwarding?         boolean
      |   |   +--rw redirection?       boolean
    +--rw resolution-strategy
    ...
  
```

Figure 6: Data Model Structure for Action Capabilities of Network Security Function

These objects are defined capabilities as ingress action, egress action, and apply profile action. These objects can be extended according to specific vendor action feature. We will add additional action objects for more generic network security functions.

6.2.4. Resolution Strategy Capabilities

The data model for resolution strategy capabilities has the following structure:

```

+--rw i2nsf-net-sec-caps
  +--rw net-sec-capabilities
    +--rw time
      | +--rw time-zone
      | | +--rw time-zone-offset?  boolean
      | +--rw time-inteval
      | | +--rw absolute-time-inteval
      | | | +--rw start-time?  boolean
      | | | +--rw end-time?    boolean
      | | +--rw periodic-time-inteval
      | | | +--rw day?        boolean
      | | | +--rw month?     boolean
    +--rw event
      | ...
    +--rw condition
      | ...
    +--rw action
      | ...
    +--rw resolution-strategy
      +--rw first-matching-rule?  boolean
      +--rw last-matching-rule?   boolean
  
```

Figure 7: Data Model Structure for Resolution Strategy Capabilities of Network Security Function

These objects are defined capabilities as first-matching-rule and last-matching-rule. These objects can be extended according to specific vendor resolution strategy features. We will add additional resolution strategy objects for more generic network security functions.

6.2.5. Content Security Capabilities

The data model for content security capabilities has the following structure:

```
+--rw complete-nsf-capabilities
  +--rw con-sec-control-capabilities
    |   +--rw anti-virus?          boolean
    |   +--rw ips?                boolean
    |   +--rw ids?                boolean
    |   +--rw url-filter?         boolean
    |   +--rw data-filter?        boolean
    |   +--rw mail-filter?        boolean
    |   +--rw sql-filter?         boolean
    |   +--rw file-blocking?      boolean
    |   +--rw file-isolate?       boolean
    |   +--rw pkt-capture?        boolean
    |   +--rw application-behavior? boolean
    |   +--rw voip-volte?         boolean
  +--rw attack-mitigation-capabilities
    ...
```

Figure 8: Data Model Structure for Content Security Capabilities of Network Security Function

Content security is composed of a number of distinct security Capabilities; each such Capability protects against a specific type of threat in the application layer. Content security is a type of Generic Network Security Function (GNSF), which summarizes a well-defined set of security Capabilities.

6.2.6. Attack Mitigation Capabilities

The data model for attack mitigation capabilities has the following structure:

```

+--rw complete-nsf-capabilities
  ...
+--rw attack-mitigation-capabilities
  +--rw (attack-mitigation-control-type)?
    +--:(ddos-attack)
      +--rw (ddos-attack-type)?
        +--:(network-layer-ddos-attack)
          +--rw network-layer-ddos-attack-types
            +--rw syn-flood-attack?          boolean
            +--rw udp-flood-attack?          boolean
            +--rw icmp-flood-attack?         boolean
            +--rw ip-fragment-flood-attack?  boolean
            +--rw ipv6-related-attack?       boolean
          +--:(app-layer-ddos-attack)
            +--rw app-layer-ddos-attack-types
              +--rw http-flood-attack?       boolean
              +--rw https-flood-attack?      boolean
              +--rw dns-flood-attack?        boolean
              +--rw dns-amp-flood-attack?    boolean
              +--rw ssl-flood-attack?        boolean
        +--:(single-packet-attack)
          +--rw (single-packet-attack-type)?
            +--:(scan-and-sniff-attack)
              +--rw ip-sweep-attack?         boolean
              +--rw port-scanning-attack?    boolean
            +--:(malformed-packet-attack)
              +--rw ping-of-death-attack?    boolean
              +--rw teardrop-attack?         boolean
            +--:(special-packet-attack)
              +--rw oversized-icmp-attack?   boolean
              +--rw tracert-attack?          boolean

```

Figure 9: Data Model Structure for Attack Mitigation Capabilities of Network Security Function

Attack mitigation is composed of a number of GNSFs; each one protects against a specific type of network attack. Attack Mitigation security is a type of GNSF, which summarizes a well-defined set of security Capabilities.

6.2.7. RPC for Acquiring Appropriate Network Security Function

The data model for RPC for Acquiring Appropriate Network Security Function has the following structure:

```

rpcs:
  +---x call-appropriate-nsf
    +---w input
      | +---w nsf-type          nsf-type
      | +---w target-device
      |   +---w pc?              boolean
      |   +---w mobile-phone?    boolean
      |   +---w voip-volte-phone? boolean
      |   +---w tablet?          boolean
      |   +---w iot?              boolean
      |   +---w vehicle?         boolean
    +--ro output
      +--ro nsf-address
        +--ro (nsf-address-type)?
          +--:(ipv4-address)
          | +--ro ipv4-address    inet:ipv4-address
          +--:(ipv6-address)
            +--ro ipv6-address    inet:ipv6-address

```

Figure 10: RPC for Acquiring Appropriate Network Security Function

This shows a RPC for acquiring an appropriate network security function according to type of NSF and/or target devices. If the SFF [i2nsf-sfc] does not have the location information of network security functions that it should send in own cache table, this can be used to acquire the information. These objects are defined as input data (i.e., NSF type and target devices) and output data (i.e., location information of NSF).

7. YANG Modules

7.1. I2NSF Capability YANG Data Module

This section introduces a YANG module for the information model of network security functions, as defined in the [i2nsf-nsf-cap-im].

```
<CODE BEGINS> file "ietf-i2nsf-capability@2018-07-02.yang"
```

```

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

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

```

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

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

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

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

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

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

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

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

```
revision "2018-07-02" {
  description "The fifth revision";
  reference
    "draft-ietf-i2nsf-capability-00";
}
```

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

```
        type inet:ipv4-address;
        mandatory true;
        description
            "nsf address type is ipv4";
    }
}
case ipv6-address {
    description
        "ipv6 case";
    leaf ipv6-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "nsf address type is ipv6";
    }
}
}
}
}

typedef nsf-type {
    type enumeration {
        enum network-firewall {
            description
                "If type of a NSF is Network Firewall.";
        }

        enum web-app-firewall {
            description
                "If type of a NSF is Web Application
                Firewall.";
        }

        enum anti-virus {
            description
                "If type of a NSF is Anti-Virus";
        }

        enum ids {
            description
                "If type of a NSF is IDS.";
        }

        enum ips {
            description
                "If type of a NSF is IPS.";
        }
    }
}
```

```
        enum ddos-mitigator {
            description
                "If type of a NSF is DDoS Mitigator.";
        }
    }
    description
        "This is used for type of NSF.";
}

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

        leaf pc {
            type boolean;
            description
                "If type of a device is PC.";
        }

        leaf mobile-phone {
            type boolean;
            description
                "If type of a device is mobile-phone.";
        }

        leaf voip-volte-phone {
            type boolean;
            description
                "If type of a device is voip-volte-phone.";
        }

        leaf tablet {
            type boolean;
            description
                "If type of a device is tablet.";
        }

        leaf iot {
            type boolean;
            description
                "If type of a device is Internet of Things.";
        }
    }
}
```

```
        leaf vehicle {
            type boolean;
            description
                "If type of a device is vehicle.";
        }
    }
}

grouping capabilities-information {
    description
        "This includes information of capabilities.";

    leaf nsf-type {
        type nsf-type;
        description
            "This is type of NSF.";
    }
    uses i2nsf-nsf-location;
    uses i2nsf-it-resources;
}

grouping i2nsf-net-sec-caps {
    description
        "i2nsf-net-sec-caps";
    container net-sec-capabilities {
        description
            "net-sec-capabilities";

        container time {
            description
                "This is capabilities for time";
            container time-zone {
                description
                    "This can be used to apply rules
                    according to time zone";

                leaf time-zone-offset {
                    type boolean;
                    description
                        "This is offset for UTC time zone";
                }
            }
        }

        container time-inteval {
            description
                "This can be used to apply rules
```

```
    according to time interval";
  container absolute-time-interval {
    description
      "This can be used to apply rules according to
      absolute time interval";
    leaf start-time {
      type boolean;
      description
        "This is start time for absolute time interval";
    }
    leaf end-time {
      type boolean;
      description
        "This is end time for absolute time interval";
    }
  }
  container periodic-time-interval {
    description
      "This can be used to apply rules according to
      periodic time interval";
    leaf day {
      type boolean;
      description
        "This is day for periodic time interval";
    }
    leaf month {
      type boolean;
      description
        "This is month for periodic time interval";
    }
  }
}

container event {
  description
    " This is abstract. 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.).";

  container usr-event {
    description "TBD";
  }
}
```

```
leaf usr-sec-event-content {
  type boolean;
  description
    "This is a mandatory string that contains the content
    of the UserSecurityEvent. The format of the content
    is specified in the usrSecEventFormat class
    attribute, and the type of event is defined in the
    usrSecEventType class attribute. An example of the
    usrSecEventContent attribute is a string hrAdmin,
    with the usrSecEventFormat set to 1 (GUID) and the
    usrSecEventType attribute set to 5 (new logon).";
}

container usr-sec-event-format {
  description
    "This is a mandatory uint 8 enumerated integer, which
    is used to specify the data type of the
    usrSecEventContent attribute. The content is
    specified in the usrSecEventContent class attribute,
    and the type of event is defined in the
    usrSecEventType class attribute. An example of the
    usrSecEventContent attribute is string hrAdmin,
    with the usrSecEventFormat attribute set to 1 (GUID)
    and the usrSecEventType attribute set to 5
    (new logon).";
  leaf unknown {
    type boolean;
    description
      "If SecEventFormat is unknown";
  }
  leaf guid {
    type boolean;
    description
      "If SecEventFormat is GUID
      (Generic Unique Identifier)";
  }
  leaf uuid {
    type boolean;
    description
      "If SecEventFormat is UUID
      (Universal Unique Identifier)";
  }
  leaf uri {
    type boolean;
    description
      "If SecEventFormat is URI
      (Uniform Resource Identifier)";
  }
}
```

```
leaf fqdn {
  type boolean;
  description
    "If SecEventFormat is FQDN
    (Fully Qualified Domain Name)";
}
leaf fqpn {
  type boolean;
  description
    "If SecEventFormat is FQPN
    (Fully Qualified Path Name)";
}
}

container usr-sec-event-type {
  leaf unknown {
    type boolean;
    description
      "If usrSecEventType is unknown";
  }
  leaf user-created {
    type boolean;
    description
      "If usrSecEventType is new user
      created";
  }
  leaf user-grp-created {
    type boolean;
    description
      "If usrSecEventType is new user
      group created";
  }
  leaf user-deleted {
    type boolean;
    description
      "If usrSecEventType is user
      deleted";
  }
  leaf user-grp-deleted {
    type boolean;
    description
      "If usrSecEventType is user
      group deleted";
  }
  leaf user-logon {
    type boolean;
    description
      "If usrSecEventType is user
```

```
        logon";
    }
    leaf user-logout {
        type boolean;
        description
            "If usrSecEventType is user
            logout";
    }
    leaf user-access-request {
        type boolean;
        description
            "If usrSecEventType is user
            access request";
    }
    leaf user-access-granted {
        type boolean;
        description
            "If usrSecEventType is user
            granted";
    }
    leaf user-access-violation {
        type boolean;
        description
            "If usrSecEventType is user
            violation";
    }
    }
description
    "This is a mandatory uint 8 enumerated integer, which
    is used to specify the type of event that involves
    this user. The content and format are specified in
    the usrSecEventContent and usrSecEventFormat class
    attributes, respectively. An example of the
    usrSecEventContent attribute is string hrAdmin,
    with the usrSecEventFormat attribute set to 1 (GUID)
    and the usrSecEventType attribute set to 5
    (new logon).";
}

}
container dev-event {
    description "TBD";

    leaf dev-sec-event-content {
        type boolean;
        mandatory true;
        description
            "This is a mandatory string that contains the content
```

```
    of the DeviceSecurityEvent. The format of the
    content is specified in the devSecEventFormat class
    attribute, and the type of event is defined in the
    devSecEventType class attribute. An example of the
    devSecEventContent attribute is alarm, with the
    devSecEventFormat attribute set to 1 (GUID), the
    devSecEventType attribute set to 5 (new logon).";
  }
```

```
container dev-sec-event-format {
  description
    "This is a mandatory uint 8 enumerated integer,
    which is used to specify the data type of the
    devSecEventContent attribute.";

  leaf unknown {
    type boolean;
    description
      "If SecEventFormat is unknown";
  }
  leaf guid {
    type boolean;
    description
      "If SecEventFormat is GUID
      (Generic Unique Identifier)";
  }
  leaf uuid {
    type boolean;
    description
      "If SecEventFormat is UUID
      (Universal Unique Identifier)";
  }
  leaf uri {
    type boolean;
    description
      "If SecEventFormat is URI
      (Uniform Resource Identifier)";
  }
  leaf fqdn {
    type boolean;
    description
      "If SecEventFormat is FQDN
      (Fully Qualified Domain Name)";
  }
  leaf fqpn {
    type boolean;
    description
      "If SecEventFormat is FQPN";
  }
}
```

```
        (Fully Qualified Path Name)";
    }
}

container dev-sec-event-type {
  description
    "This is a mandatory uint 8 enumerated integer,
    which is used to specify the type of event
    that was generated by this device.";

  leaf unknown {
    type boolean;
    description
      "If devSecEventType is unknown";
  }
  leaf comm-alarm {
    type boolean;
    description
      "If devSecEventType is communications
      alarm";
  }
  leaf quality-of-service-alarm {
    type boolean;
    description
      "If devSecEventType is quality of service
      alarm";
  }
  leaf process-err-alarm {
    type boolean;
    description
      "If devSecEventType is processing error
      alarm";
  }
  leaf equipment-err-alarm {
    type boolean;
    description
      "If devSecEventType is equipment error
      alarm";
  }
  leaf environmental-err-alarm {
    type boolean;
    description
      "If devSecEventType is environmental error
      alarm";
  }
}
}
```

```
container dev-sec-event-type-severity {
  description
    "This is a mandatory uint 8 enumerated integer,
    which is used to specify the perceived
    severity of the event generated by this
    Device.";

  leaf unknown {
    type boolean;
    description
      "If devSecEventType is unknown";
  }
  leaf cleared {
    type boolean;
    description
      "If devSecEventTypeSeverity is cleared";
  }
  leaf indeterminate {
    type boolean;
    description
      "If devSecEventTypeSeverity is
      indeterminate";
  }
  leaf critical {
    type boolean;
    description
      "If devSecEventTypeSeverity is critical";
  }
  leaf major {
    type boolean;
    description
      "If devSecEventTypeSeverity is major";
  }
  leaf minor {
    type boolean;
    description
      "If devSecEventTypeSeverity is minor";
  }
  leaf warning {
    type boolean;
    description
      "If devSecEventTypeSeverity is warning";
  }
}
}
container sys-event {
  description "TBD";
```

```
leaf sys-sec-event-content {
  type boolean;
  description
    "This is a mandatory string that contains a content
    of the SystemSecurityEvent. The format of a content
    is specified in a sysSecEventFormat class attribute,
    and the type of event is defined in the
    sysSecEventType class attribute. An example of the
    sysSecEventContent attribute is string sysadmin3,
    with the sysSecEventFormat attribute set to 1(GUID),
    and the sysSecEventType attribute set to 2
    (audit log cleared).";
}

container sys-sec-event-format {
  description
    "This is a mandatory uint 8 enumerated integer, which
    is used to specify the data type of the
    sysSecEventContent attribute.";

  leaf unknown {
    type boolean;
    description
      "If SecEventFormat is unknown";
  }
  leaf guid {
    type boolean;
    description
      "If SecEventFormat is GUID
      (Generic Unique Identifier)";
  }
  leaf uuid {
    type boolean;
    description
      "If SecEventFormat is UUID
      (Universal Unique Identifier)";
  }
  leaf uri {
    type boolean;
    description
      "If SecEventFormat is URI
      (Uniform Resource Identifier)";
  }
  leaf fqdn {
    type boolean;
    description
      "If SecEventFormat is FQDN
      (Fully Qualified Domain Name)";
  }
}
```

```
    }
    leaf fqpn {
        type boolean;
        description
            "If SecEventFormat is FQPN
            (Fully Qualified Path Name)";
    }
}

container sys-sec-event-type {
    description
        "This is a mandatory uint 8 enumerated integer, which
        is used to specify the type of event that involves
        this device.";

    leaf unknown {
        type boolean;
        description
            "If sysSecEventType is unknown";
    }
    leaf audit-log-written-to {
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that audit log is written to";
    }
    leaf audit-log-cleared {
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that audit log is cleared";
    }
    leaf policy-created {
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that policy is created";
    }
    leaf policy-edited{
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that policy is edited";
    }
    leaf policy-deleted{
        type boolean;
        description
            "If sysSecEventTypeSeverity
```

```
        is that policy is deleted";
    }
    leaf policy-executed{
        type boolean;
        description
            "If sysSecEventTypeSeverity
            is that policy is executed";
    }
}
}
container time-event {
    description "TBD";

    leaf time-sec-event-begin {
        type boolean;
        description
            "This is a mandatory DateTime attribute, and
            represents the beginning of a time period.
            It has a value that has a date and/or a time
            component (as in the Java or Python libraries).";
    }

    leaf time-sec-event-end {
        type boolean;
        description
            "This is a mandatory DateTime attribute, and
            represents the end of a time period. It has
            a value that has a date and/or a time component
            (as in the Java or Python libraries). If this is
            a single event occurrence, and not a time period
            when the event can occur, then the
            timeSecEventPeriodEnd attribute may be ignored.";
    }

    leaf time-sec-event-time-zone {
        type boolean;
        description
            "This is a mandatory string attribute, and defines a
            time zone that this event occurred in using the
            format specified in ISO8601.";
    }
}

}

container condition {
    description
        " This is abstract. 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.";

```
container packet-security-condition {
  description "TBD";

  container packet-security-mac-condition {
    description
      "The purpose of this Class is to represent packet MAC
      packet header information that can be used as part of
      a test to determine if the set of Policy Actions in
      this ECA Policy Rule should be execute or not.";

    leaf pkt-sec-cond-mac-dest {
      type boolean;
      description
        "The MAC destination address (6 octets long).";
    }

    leaf pkt-sec-cond-mac-src {
      type boolean;
      description
        "The MAC source address (6 octets long).";
    }

    leaf pkt-sec-cond-mac-8021q {
      type boolean;
      description
        "This is an optional string attribute, and defines
        The 802.1Q tag value (2 octets long).";
    }

    leaf pkt-sec-cond-mac-ether-type {
      type boolean;
      description
        "The EtherType field (2 octets long). Values up to
        and including 1500 indicate the size of the payload
        in octets; values of 1536 and above define which
        protocol is encapsulated in the payload of the
        frame.";
    }
  }
}
```

```
leaf pkt-sec-cond-mac-tci {
  type string;
  description
    "This is an optional string attribute, and defines
    the Tag Control Information. This consists of a 3
    bit user priority field, a drop eligible indicator
    (1 bit), and a VLAN identifier (12 bits).";
}
}

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

  leaf pkt-sec-cond-ipv4-header-length {
    type boolean;
    description
      "The IPv4 packet header consists of 14 fields,
      of which 13 are required.";
  }

  leaf pkt-sec-cond-ipv4-tos {
    type boolean;
    description
      "The ToS field could specify a datagram's priority
      and request a route for low-delay, high-throughput,
      or highly-reliable service..";
  }

  leaf pkt-sec-cond-ipv4-total-length {
    type boolean;
    description
      "This 16-bit field defines the entire packet size,
      including header and data, in bytes.";
  }

  leaf pkt-sec-cond-ipv4-id {
    type boolean;
    description
      "This field is an identification field and is
      primarily used for uniquely identifying
      the group of fragments of a single IP datagram.";
  }

  leaf pkt-sec-cond-ipv4-fragment {
```

```
type boolean;
description
  "IP fragmentation is an Internet Protocol (IP)
  process that breaks datagrams into smaller pieces
  (fragments), so that packets may be formed that
  can pass through a link with a smaller maximum
  transmission unit (MTU) than the original
  datagram size."
}

leaf pkt-sec-cond-ipv4-fragment-offset {
  type boolean;
  description
    "Fragment offset field along with Don't Fragment
    and More Fragment flags in the IP protocol
    header are used for fragmentation and reassembly
    of IP datagrams."
}

leaf pkt-sec-cond-ipv4-ttl {
  type boolean;
  description
    "The ttl keyword is used to check for a specific
    IP time-to-live value in the header of
    a packet."
}

leaf pkt-sec-cond-ipv4-protocol {
  type boolean;
  description
    "Internet Protocol version 4(IPv4) is the fourth
    version of the Internet Protocol (IP).";
}

leaf pkt-sec-cond-ipv4-src {
  type boolean;
  description
    "Defines the IPv4 Source Address."
}

leaf pkt-sec-cond-ipv4-dest {
  type boolean;
  description
    "Defines the IPv4 Destination Address."
}

leaf pkt-sec-cond-ipv4-ipopts {
  type boolean;
```

```
description
  "With the ipopts keyword you can check if
  a specific ip option is set. Ipopts has
  to be used at the beginning of a rule.";
}

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

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

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

  leaf pkt-sec-cond-ipv6-dscp {
    type boolean;
    description
      "Differentiated Services Code Point (DSCP)
      of ipv6.";
  }

  leaf pkt-sec-cond-ipv6-ecn {
    type boolean;
    description
      "ECN allows end-to-end notification of network
      congestion without dropping packets.";
  }
}
```

```
leaf pkt-sec-cond-ipv6-traffic-class {
  type boolean;
  description
    "The bits of this field hold two values. The 6
    most-significant bits are used for
    differentiated services, which is used to
    classify packets.";
}

leaf pkt-sec-cond-ipv6-flow-label {
  type boolean;
  description
    "The flow label when set to a non-zero value
    serves as a hint to routers and switches
    with multiple outbound paths that these
    packets should stay on the same path so that
    they will not be reordered.";
}

leaf pkt-sec-cond-ipv6-payload-length {
  type boolean;
  description
    "The size of the payload in octets,
    including any extension headers.";
}

leaf pkt-sec-cond-ipv6-next-header {
  type boolean;
  description
    "Specifies the type of the next header.
    This field usually specifies the transport
    layer protocol used by a packet's payload.";
}

leaf pkt-sec-cond-ipv6-hop-limit {
  type boolean;
  description
    "Replaces the time to live field of IPv4.";
}

leaf pkt-sec-cond-ipv6-src {
  type boolean;
  description
    "The IPv6 address of the sending node.";
}

leaf pkt-sec-cond-ipv6-dest {
  type boolean;
```

```
        description
            "The IPv6 address of the destination node(s).";
    }
}

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

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

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

    leaf pkt-sec-cond-tcp-seq-num {
        type boolean;
        description
            "If the SYN flag is set (1), then this is the
            initial sequence number.";
    }

    leaf pkt-sec-cond-tcp-ack-num {
        type boolean;
        description
            "If the ACK flag is set then the value of this
            field is the next sequence number that the sender
            is expecting.";
    }

    leaf pkt-sec-cond-tcp-window-size {
        type boolean;
        description
            "The size of the receive window, which specifies
            the number of windows size units (by default, bytes)
            (beyond the segment identified by the sequence
```

```
        number in the acknowledgment field) that the sender
        of this segment is currently willing to receive.";
    }

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

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

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

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

    leaf pkt-sec-cond-udp-length {
        type boolean;
        description
            "This is a mandatory string attribute, and defines
            the length in bytes of the UDP header and data
            (16 bits).";
    }
}

container packet-security-icmp-condition {
    description
        "The internet control message protocol condition.";

    leaf pkt-sec-cond-icmp-type {
        type boolean;
    }
}
```

```
        description
          "ICMP type, see Control messages.>";
      }

      leaf pkt-sec-cond-icmp-code {
        type boolean;
        description
          "ICMP subtype, see Control messages.>";
      }

      leaf pkt-sec-cond-icmp-seg-num {
        type boolean;
        description
          "The icmp Sequence Number.>";
      }
    }
  }

  container packet-payload-condition {
    description "TBD";
    leaf pkt-payload-content {
      type boolean;
      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 acl-number {
    type boolean;
    description
      "This is acl-number.>";
  }

  container application-condition {
    description
      "TBD";

    leaf application-object {
      type boolean;
      description
        "This is application object.>";
    }

    leaf application-group {
      type boolean;
      description
        "This is application group.>";
    }
  }
}
```

```
    }
    leaf application-label {
        type boolean;
        description
            "This is application label.";
    }
    container category {
        description
            "TBD";
        leaf application-category {
            type boolean;
            description
                "TBD";
        }
    }
}

container target-condition {
    description "TBD";

    leaf device-sec-context-cond {
        type boolean;
        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.";
    }
}

container users-condition {
    description "TBD";

    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 boolean;
    description
      "User's tenant information.";
  }
}

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

  leaf vn-id {
    type boolean;
    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 boolean;
            description
                "User's tenant information.";
        }
    }

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

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

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

}

container url-category-condition {
    description
        "TBD";
    leaf pre-defined-category {
        type boolean;
        description
            "This is pre-defined-category.";
    }
    leaf user-defined-category {
        type boolean;
        description
            "This user-defined-category.";
    }
}

container context-condition {
    description "TBD";
    leaf temp {
        type string;
        description
            "This is temp for context condition.";
    }
}
```

```
    }
  }
  container gen-context-condition {
    description "TBD";

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

  leaf rule-log {
    type boolean;
    description
      "rule-log";
  }
  leaf session-log {
    type boolean;
    description

```

```
    "session-log";
  }

  container ingress-action {
    description "TBD";

    container ingress-action-type {
      description
        "Ingress action type: permit, deny, and mirror.";
      leaf pass {
        type boolean;
        description
          "If ingress action is pass";
      }
      leaf drop {
        type boolean;
        description
          "If ingress action is drop";
      }
      leaf reject {
        type boolean;
        description
          "If ingress action is reject";
      }
      leaf alert {
        type boolean;
        description
          "If ingress action is alert";
      }
      leaf mirror {
        type boolean;
        description
          "If ingress action is mirror";
      }
    }
  }

  container egress-action {
    description "TBD";

    container egress-action-type {
      description
        "Egress-action-type: invoke-signaling,
        tunnel-encapsulation, and forwarding.";
      leaf invoke-signaling {
        type boolean;
        description
          "If egress action is invoke signaling";
      }
    }
  }
}
```

```
        leaf tunnel-encapsulation {
            type boolean;
            description
                "If egress action is tunnel encapsulation";
        }
        leaf forwarding {
            type boolean;
            description
                "If egress action is forwarding";
        }
        leaf redirection {
            type boolean;
            description
                "If egress action is redirection";
        }
    }
}

}
container resolution-strategy {
    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";

    leaf first-matching-rule {
        type boolean;
        description
            "If the resolution strategy is first matching rule";
    }

    leaf last-matching-rule {
        type boolean;
        description
            "If the resolution strategy is last matching rule";
    }
}
}
}

grouping i2nsf-con-sec-control-caps {
    description
        "i2nsf-con-sec-control-caps";

    container con-sec-control-capabilities {
        description
            "content-security-control-capabilities";
    }
}
```

```
leaf anti-virus {
  type boolean;
  description
    "antivirus";
}
leaf ips {
  type boolean;
  description
    "ips";
}

leaf ids {
  type boolean;
  description
    "ids";
}

leaf url-filter {
  type boolean;
  description
    "url-filter";
}
leaf data-filter {
  type boolean;
  description
    "data-filter";
}
leaf mail-filter {
  type boolean;
  description
    "mail-filter";
}
leaf sql-filter {
  type boolean;
  description
    "sql-filter";
}
leaf file-blocking {
  type boolean;
  description
    "file-blocking";
}
leaf file-isolate {
  type boolean;
  description
    "file-isolate";
}
leaf pkt-capture {
```

```
        type boolean;
        description
            "pkt-capture";
    }
    leaf application-behavior {
        type boolean;
        description
            "application-behavior";
    }
    leaf voip-volte {
        type boolean;
        description
            "voip-volte";
    }
}

}

grouping i2nsf-attack-mitigation-control-caps {
    description
        "i2nsf-attack-mitigation-control-caps";

    container attack-mitigation-capabilities {
        description
            "attack-mitigation-capabilities";
        choice attack-mitigation-control-type {
            description
                "attack-mitigation-control-type";
            case ddos-attack {
                description
                    "ddos-attack";
                choice ddos-attack-type {
                    description
                        "ddos-attack-type";
                    case network-layer-ddos-attack {
                        description
                            "network-layer-ddos-attack";
                        container network-layer-ddos-attack-types {
                            description
                                "network-layer-ddos-attack-type";
                            leaf syn-flood-attack {
                                type boolean;
                                description
                                    "syn-flood-attack";
                            }
                            leaf udp-flood-attack {
                                type boolean;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
        description
            "udp-flood-attack";
    }
    leaf icmp-flood-attack {
        type boolean;
        description
            "icmp-flood-attack";
    }
    leaf ip-fragment-flood-attack {
        type boolean;
        description
            "ip-fragment-flood-attack";
    }
    leaf ipv6-related-attack {
        type boolean;
        description
            "ip-fragment-flood-attack";
    }
}
}
}
case app-layer-ddos-attack {
    description
        "app-layer-ddos-attack";
    container app-layer-ddos-attack-types {
        description
            "app-layer-ddos-attack-types";
        leaf http-flood-attack {
            type boolean;
            description
                "http-flood-attack";
        }
        leaf https-flood-attack {
            type boolean;
            description
                "https-flood-attack";
        }
        leaf dns-flood-attack {
            type boolean;
            description
                "dns-flood-attack";
        }
        leaf dns-amp-flood-attack {
            type boolean;
            description
                "dns-amp-flood-attack";
        }
        leaf ssl-flood-attack {
            type boolean;
        }
    }
}
```

```
        description
          "ssl-flood-attack";
      }
    }
  }
}

case single-packet-attack {
  description
    "single-packet-attack";
  choice single-packet-attack-type {
    description
      "single-packet-attack-type";
    case scan-and-sniff-attack {
      description
        "scan-and-sniff-attack";
      leaf ip-sweep-attack {
        type boolean;
        description
          "ip-sweep-attack";
      }
      leaf port-scanning-attack {
        type boolean;
        description
          "port-scanning-attack";
      }
    }
    case malformed-packet-attack {
      description
        "malformed-packet-attack";
      leaf ping-of-death-attack {
        type boolean;
        description
          "ping-of-death-attack";
      }
      leaf teardrop-attack {
        type boolean;
        description
          "teardrop-attack";
      }
    }
  }
  case special-packet-attack {
    description
      "special-packet-attack";
    leaf oversized-icmp-attack {
      type boolean;
      description

```

```
        "oversized-icmp-attack";
    }
    leaf tracert-attack {
        type boolean;
        description
            "tracert-attack";
    }
}
}
}
}
}
}
}
}

list nsf {
    key "nsf-name";
    description
        "nsf-name";
    leaf nsf-name {
        type string;
        mandatory true;
        description
            "nsf-name";
    }
}

uses capabilities-information;

container generic-nsf-capabilities {
    description
        "generic-nsf-capabilities";
    uses i2nsf-net-sec-caps;
}

container complete-nsf-capabilities {
    description
        "generic-nsf-capabilities";
    uses i2nsf-con-sec-control-caps;
    uses i2nsf-attack-mitigation-control-caps;
}

}

rpc call-appropriate-nsf {
    description
        "We can acquire appropriate NSF that we want
```

If we give type of NSF that we want to use,
we acquire the location information of NSF";

```
input {
  leaf nsf-type {
    type nsf-type;
    mandatory true;
    description
      "This is used to acquire NSF
       This is mandatory";
  }
  uses i2nsf-it-resources;
}
output {
  uses i2nsf-nsf-location;
}
}
```

<CODE ENDS>

Figure 11: YANG Data Module of I2NSF Capability

8. IANA Considerations

No IANA considerations exist for this document at this time. URL will be added.

9. Security Considerations

This document introduces no additional security threats and SHOULD follow the security requirements as stated in [RFC8329].

10. 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).

11. 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 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)

12. References

12.1. 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.
- [RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.
- [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.

12.2. Informative References

- [i2nsf-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-01 (work in progress), April 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-00 (work in progress), March 2018.

[i2nsf-sfc]

Hyun, S., Jeong, J., Park, J., and S. Hares, "Service Function Chaining-Enabled I2NSF Architecture", draft-hyun-i2nsf-nsf-triggered-steering-05 (work in progress), March 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-05 (work in progress), January 2018.

[i2rs-rib-data-model]

Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-12 (work in progress), April 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. Example: Extended VoIP-VoLTE Security Function Capabilities Module

This section gives a simple example of how VoIP-VoLTE Security Function Capabilities module could be extended.

```
module
ex-voip-volte-cap {
  namespace "http://example.com/voip-volte-cap";
  prefix "voip-volte-cap";

  import ietf-i2nsf-capability {
    prefix capa;
  }

  augment "/capa:nsf/capa:generic-nsf-capabilities/"
    + "capa:net-sec-control-capabilities/"
    + "capa:condition/capa:condition-type" {
  case voice-condition {
    leaf sip-header-method {
      type boolean;
      description
        "SIP header method.";
    }

    leaf sip-header-uri {
      type boolean;
      description
        "SIP header URI.";
    }

    leaf sip-header-from {
      type boolean;
      description
        "SIP header From.";
    }

    leaf sip-header-to {
      type boolean;
      description
        "SIP header To.";
    }

    leaf sip-header-expire-time {
      type boolean;
      description
        "SIP header expire time.";
    }
  }
}
```

```
    }  
    leaf sip-header-user-agent {  
      type boolean;  
      description  
        "SIP header user agent.";  
    }  
  }  
}
```

Figure 12: Example: Extended VoIP-VoLTE Security Function Capabilities Module

Appendix B. Example: Configuration XML of Capability Module

This section gives a xml examples for a configuration of Capability module according to a requirement.

B.1. Example: Configuration XML of Generic Network Security Function Capabilities

This section gives a xml example for generic network security function capability configuration according to a requirement.

Requirement: Register packet filter according to requirements.

1. The location of the NSF is 221.159.112.150.
2. The NSF can obtain the best effect if the packet was generated by PC or IoT.
3. The NSF can apply policies according to time.
4. The NSF should be able to block the source packets or destination packets with IPv4 address.
5. The NSF should be able to pass, reject, or alert packets.
6. Here is XML example for the generic network security function capability configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
    <running />
  </target>
<config>
  <nsf xmlns="urn:ietf:params:xml:ns:yang:" +
    "ietf-i2nsf-capability">
    <nsf-name>Huawei-Firewall</nsf-name>
    <nsf-address>
      <ipv4-address>221.159.112.150</ipv4-address>
    </nsf-address>
    <target-device>
      <pc>true</pc>
    </target-device>
    <target-device>
      <iot>true</iot>
    </target-device>
    <generic-nsf-capabilities>
      <net-sec-control-capabilities>
        <nsf-capabilities-name>ipv4-packet-filter</nsf-capabilities-name>
        <time-zone>
          <start-time>true</start-time>
          <end-time>true</end-time>
        </time-zone>
        <condition>
          <packet-security-ipv4-condition>
            <pkt-sec-cond-ipv4-src>true</pkt-sec-cond-ipv4-src>
            <pkt-sec-cond-ipv4-dest>true</pkt-sec-cond-ipv4-dest>
          </packet-security-ipv4-condition>
        </condition>
        <action>
          <ingress-action-type>
            <pass>true</pass>
            <reject>true</reject>
            <alert>true</alert>
          </ingress-action-type>
        </action>
      </net-sec-control-capabilities>
    </generic-nsf-capabilities>
  </nsf>
</config>
</edit-config>
</rpc>
```

Figure 13: Example: Configuration XML for Generic Network Security Function Capability

B.2. Example: Configuration XML of Extended VoIP/VoLTE Security Function Capabilities Module

This section gives a xml example for extended VoIP-VoLTE security function capabilities (See Figure 12) configuration according to a requirement.

Requirement: Register VoIP/VoLTe security function according to requirements.

1. The location of the NSF is 221.159.112.151.
2. The NSF can obtain the best effect if the packet was generated by VoIP-VoLTE phone.
3. The NSF should be able to block the malicious sip packets with user agent.
4. The NSF should be able to pass, reject, or alert packets.

Here is XML example for the VoIP-VoLTE security function capabilities configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
    <running />
  </target>
<config>
  <nsf xmlns="urn:ietf:params:xml:ns:yang:" +
    "ietf-i2nsf-capability">
    <nsf-name>Cisco-VoIP-VoLTE</nsf-name>
    <nsf-address>
      <ipv4-address>221.159.112.151</ipv4-address>
    </nsf-address>
    <generic-nsf-capabilities>
      <net-sec-control-capabilities>
        <nsc-capabilities-name>sip-packet-filter<nsc-capabilities-name>
          <condition>
            <sip-header-user-agent>true</sip-header-user-agent>
          </condition>
          <action>
            <ingress-action-type>
              <pass>true</pass>
              <reject>true</reject>
              <alert>true</alert>
            </ingress-action-type>
          </action>
        </net-sec-control-capabilities>
      </generic-nsf-capabilities>
    </nsf>
  </config>
</edit-config>
</rpc>
```

Figure 14: Example: Configuration XML for Extended VoIP/VoLTE Security Function Capabilities

Appendix C. Changes from draft-ietf-i2nsf-capability-data-model-01

The following changes are made from draft-ietf-i2nsf-capability-data-model-00:

1. We have clarified and simplified capabilities.
2. We added additional condition capabilities for application and url.
3. We replaced unnecessary leaf-list component to leaf component.

4. We replaced the list component to the container component for net-sec-capabilities.
5. We modified the choice-case structure into a container structure to allow for the selection of multiple catalogues for condition and action clauses.
6. We added complete-nsf-capabilities such as content capabilities and attack mitigation capabilities.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

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

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

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

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

Robert Moskowitz
HTT Consulting
Oak Park, MI
USA

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

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

EMail: linqiushi@huawei.com

I2NSF Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 8, 2019

J. Jeong
E. Kim
Sungkyunkwan University
T. Ahn
Korea Telecom
R. Kumar
Juniper Networks
S. Hares
Huawei
November 4, 2018

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

Abstract

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 May 8, 2019.

Copyright Notice

Copyright (c) 2018 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
4. Data Modeling for Security Policies for Consumer-Facing Interface	3
4.1. YANG Data Model for Security Policies for Consumer-Facing Interface	8
5. Use Case: Policy Instance Example for VoIP/VoLTE Security Services	33
5.1. Policy Instance YANG Example for VoIP/VoLTE Security Services	35
6. Example XML Output for Various Use Cases	45
6.1. Case 1: VoIP Security Service	45
6.2. Case 2: DDoS-Attack Mitigation	47
6.3. Case 3: Time-Based Firewall	48
6.4. Case 4: Time-Based Web-Filter	49
6.5. Case 5: Threat-Feed Configuration	50
7. Security Considerations	51
8. References	51
8.1. Normative References	51
8.2. Informative References	52
Appendix A. Changes from draft-ietf-i2nsf-consumer-facing-interface-dm-01	53
Appendix B. Acknowledgments	53
Appendix C. Contributors	53
Authors' Addresses	53

1. Introduction

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

administrative domain. This document defines a YANG data model based on the information model of I2NSF Consumer-Facing Interface [client-facing-inf-im].

Data models are defined at a lower level of abstraction and provide many details. They provide details about the implementation of a protocol's specification, e.g., rules that explain how to map managed objects onto lower-level protocol constructs. Since conceptual models can be implemented in different ways, multiple data models can be derived by a single information model.

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

2. Requirements Language

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

3. Terminology

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

4. Data Modeling for 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.

Multi-tenancy in this document 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 departments. Thus, we need an object which defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. You can think of it as a way to assign policy users to a job function or a set of permissions within the organization. The policy-role object SHALL have Name, Date and access-profile to grant or deny permissions for the purpose of security policy management.

```

module: policy-general
  +--rw policy
    |
    |   +--rw rule* [rule-id]
    |   |   +--rw rule-id          uint16
    |   |   +--rw name?           string
    |   |   +--rw date?          yang:date-and-time
    |   |   +--rw case?          string
    |   |   +--rw event* [event-id]
    |   |   |   +--rw event-id      string
    |   |   |   +--rw name?        string
    |   |   |   +--rw date?        yang:date-and-time
    |   |   |   +--rw event-type?  string
    |   |   |   +--rw time-information? string
    |   |   |   +--rw event-map-group? -> /threat-feed/event-map-group
    |   |   |   |   /event-map-group-id
    |   |   |   +--rw enable?      boolean
    |   |   +--rw condition* [condition-id]
    |   |   |   +--rw condition-id  string
    |   |   |   +--rw source?      string
    |   |   |   +--rw destination? string
    |   |   |   +--rw match?       boolean
    |   |   |   +--rw match-direction? string
    |   |   |   +--rw exception?   string
    |   |   +--rw policy-action* [policy-action-id]
    |   |   |   +--rw policy-action-id string
    |   |   |   +--rw name?        string
    |   |   |   +--rw date?        yang:date-and-time
    |   |   |   +--rw primary-action? string
    |   |   |   +--rw secondary-action? string
    |   |   |   +--rw owner?      string

```

```

+--rw multi-tenancy
  +--rw policy-domain* [policy-domain-id]
    +--rw policy-domain-id      uint16
    +--rw name                   string
    +--rw address?              string
    +--rw contact                string
    +--rw date                   yang:date-and-time
    +--rw policy-tenant* [policy-tenant-id]
      +--rw policy-tenant-id    uint16
      +--rw name                 string
      +--rw date                 yang:date-and-time
      +--rw domain?             -> /multi-tenancy
                                /policy-domain
                                /policy-domain-id
    +--rw authentication-method? -> /multi-tenancy
                                    /policy-mgmt-auth-method
                                    /policy-mgmt-auth-method-id
  +--rw policy-role* [policy-role-id]
    +--rw policy-role-id        uint16
    +--rw name                   string
    +--rw date                   yang:date-and-time
    +--rw access-profile         string
  +--rw policy-user* [policy-user-id]
    +--rw policy-user-id        uint16
    +--rw name                   string
    +--rw date                   yang:date-and-time
    +--rw password               string
    +--rw email                  string
    +--rw scope-type?            string
    +--rw scope-reference?       string
    +--rw role                    string
  +--rw policy-mgmt-auth-method* [policy-mgmt-auth-method-id]
    +--rw policy-mgmt-auth-method-id uint16
    +--rw name                     string
    +--rw date                       yang:date-and-time
    +--rw authentication-method      enumeration
    +--rw mutual-authentication      boolean
    +--rw token-server                inet:ipv4-address
    +--rw certificate-server          inet:ipv4-address
    +--rw single-sing-on-server       inet:ipv4-address
+--rw endpoint-group
  +--rw meta-data-source* [meta-data-source-id]
    +--rw meta-data-source-id    uint16
    +--rw name                     string
    +--rw date                       yang:date-and-time
    +--rw tag-type?                boolean
    +--rw tag-server-information?   inet:ipv4-address
    +--rw tag-application-protocol? string

```

```

|   +--rw tag-server-credential?      string
+--rw user-group* [user-group-id]
|   +--rw user-group-id              uint16
|   +--rw name?                      string
|   +--rw date?                      yang:date-and-time
|   +--rw group-type?                enumeration
|   +--rw meta-data-server?          inet:ipv4-address
|   +--rw group-member?              string
|   +--rw risk-level?                uint16
+--rw device-group* [device-group-id]
|   +--rw device-group-id            uint16
|   +--rw name?                      string
|   +--rw date?                      yang:date-and-time
|   +--rw group-type?                enumeration
|   +--rw meta-data-server?          inet:ipv4-address
|   +--rw group-member?              string
|   +--rw risk-level?                uint16
+--rw application-group* [application-group-id]
|   +--rw application-group-id        uint16
|   +--rw name?                      string
|   +--rw date?                      yang:date-and-time
|   +--rw group-type?                enumeration
|   +--rw meta-data-server?          inet:ipv4-address
|   +--rw group-member?              string
|   +--rw risk-level?                uint16
+--rw location-group* [location-group-id]
|   +--rw location-group-id           uint16
|   +--rw name?                      string
|   +--rw date?                      yang:date-and-time
|   +--rw group-type?                enumeration
|   +--rw meta-data-server?          inet:ipv4-address
|   +--rw group-member?              string
|   +--rw risk-level?                uint16
+--rw threat-feed
|   +--rw threat-feed* [threat-feed-id]
|   |   +--rw threat-feed-id          uint16
|   |   +--rw name?                  string
|   |   +--rw date?                  yang:date-and-time
|   |   +--rw feed-type               enumeration
|   |   +--rw feed-server?            inet:ipv4-address
|   |   +--rw feed-priority?          uint16
|   +--rw custom-list* [custom-list-id]
|   |   +--rw custom-list-id          uint16
|   |   +--rw name?                  string
|   |   +--rw date?                  yang:date-and-time
|   |   +--rw list-type               enumeration
|   |   +--rw list-property           enumeration
|   |   +--rw list-content?           string

```

```

|--rw malware-scan-group* [malware-scan-group-id]
|   |--rw malware-scan-group-id      uint16
|   |--rw name?                      string
|   |--rw date?                      yang:date-and-time
|   |--rw signature-server?          inet:ipv4-address
|   |--rw file-types?               string
|   |--rw malware-signatures?       string
|--rw event-map-group* [event-map-group-id]
|   |--rw event-map-group-id        uint16
|   |--rw name?                     string
|   |--rw date?                     yang:date-and-time
|   |--rw security-events?          string
|   |--rw threat-map?               string
|--rw telemetry-data
|   |--rw telemetry-data* [telemetry-data-id]
|   |   |--rw telemetry-data-id      uint16
|   |   |--rw name?                 string
|   |   |--rw date?                 yang:date-and-time
|   |   |--rw logs?                 boolean
|   |   |--rw syslog?               boolean
|   |   |--rw snmp?                 boolean
|   |   |--rw sflow?                boolean
|   |   |--rw netflow?              boolean
|   |   |--rw interface-stats?      boolean
|--rw telemetry-source* [telemetry-source-id]
|   |--rw telemetry-source-id        uint16
|   |--rw name?                      string
|   |--rw date?                      yang:date-and-time
|   |--rw source-type?               enumeration
|   |--rw nsf-source?                inet:ipv4-address
|   |--rw nsf-credentials?           string
|   |--rw collection-interval?       uint16
|   |--rw collection-method?         enumeration
|   |--rw heartbeat-interval?        uint16
|   |--rw qos-marking?               uint16
|--rw telemetry-destination* [telemetry-destination-id]
|   |--rw telemetry-destination-id   uint16
|   |--rw name?                      string
|   |--rw date?                      yang:date-and-time
|   |--rw collector-source?          inet:ipv4-address
|   |--rw collector-credentials?     string
|   |--rw data-encoding?             string
|   |--rw data-transport?            enumeration

```

Figure 1: Generic Data Model for Security Policies for cf Interface

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

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

```
<CODE BEGINS> file "policy-general.yang"
module ietf-policy-general {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-policy-general";
  prefix
    cf-interface;

  import ietf-yang-types{
    prefix yang;
  }

  import ietf-inet-types{
    prefix inet;
  }

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

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

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

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

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

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

  revision "2018-11-04"{
    description "fourth revision";
    reference
      "draft-kumar-i2nsf-client-facing-interface-im-04";
  }
}
```

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

  list rule {
    key "rule-id";
    leaf rule-id {
      type uint16;
      description
      "This is ID for rules.";
    }
    description
    "This is a container for rules.";
    leaf name {
      type string;
      description
      "This field identifies the name of this object.";
    }
    leaf date {
      type yang:date-and-time;
      description
      "Date this object was created or last
      modified";
    }
    leaf case {
      type string;
      description
      "to identify whether the rule belongs to
      web filter or enterprise mode.";
    }
    list event {
      key "event-id";
      description
      "This represents the security event of a
      policy-rule.";

      leaf event-id {
        type string;
        mandatory true;
        description
        "This represents the event-id.";
      }
      leaf name {
        type string;
        description

```

```
        "This field identifies the name of this object.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "Date this object was created or last
            modified";
    }
    leaf event-type {
        type string;
        description
            "This field identifies the event of
            policy enforcement trigger type.";
    }
    leaf time-information {
        type string;
        description
            "This field contains time calendar such as
            BEGIN-TIME and END-TIME for one time
            enforcement or recurring time calendar for
            periodic enforcement.";
    }
    leaf event-map-group {
        type leafref {
            path "/threat-feed/event-map-group/event-map-group-id";
        }
        description
            "This field contains security events or threat
            map in order to determine when a policy need
            to be activated. This is a reference to
            Evnet-Map-Group.";
    }
    leaf enable {
        type boolean;
        description
            "This determines whether the condition
            matches the security event or not.";
    }
}
list condition {
    key "condition-id";
    description
        "This represents the condition of a
        policy-rule.";

    leaf condition-id {
        type string;
        description
```

```

    "This represents the condition-id.";
}
leaf source {
  type string;
  description
    "This field identifies the source of
    the traffic. This could be reference to
    either 'Policy Endpoint Group' or
    'Threat-Feed' or 'Custom-List' if Security
    Admin wants to specify the source; otherwise,
    the default is to match all traffic.";
}
leaf destination {
  type string;
  description
    "This field identifies the source of
    the traffic. This could be reference to
    either 'Policy Endpoint Group' or
    'Threat-Feed' or 'Custom-List' if Security
    Admin wants to specify the source; otherwise,
    the default is to match all traffic.";
}
leaf match {
  type boolean;
  description
    "This field identifies the match criteria used to
    evaluate whether the specified action need to be
    taken or not. This could be either a Policy-
    Endpoint-Group identifying a Application set or a
    set of traffic rules.";
}
leaf match-direction {
  type string;
  description
    "This field identifies if the match criteria is
    to evaluated for both direction of the traffic or
    only in one direction with default of allowing in
    the other direction for stateful match conditions.
    This is optional and by default rule should apply
    in both directions.";
}
leaf exception {
  type string;
  description
    "This field identifies the exception
    consideration when a rule is evaluated for a
    given communication. This could be reference to
    Policy-Endpoint-Group object or set of traffic

```

```

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

  leaf name {
    type string;
    description
      "The name of the policy-action object.";
  }
  leaf date {
    type yang:date-and-time;
    description
      "When the object was created or last
      modified.";
  }
  leaf primary-action {
    type string;
    description
      "This field identifies the action when a rule
      is matched by NSF. The action could be one of
      'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
      'AUTHENTICATE-SESSION', 'IPS', 'APP-FIREWALL', etc.";
  }
  leaf secondary-action {
    type string;
    description
      "This field identifies additional actions if
      a rule is matched. This could be one of 'LOG',
      'SYSLOG', 'SESSION-LOG', etc.";
  }
  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 descriptions of multi-tenancy.";

  list policy-domain {
    key "policy-domain-id";

    leaf policy-domain-id {
      type uint16;
      description
        "This represents the list of domains.";
    }
    description
      "this represent the list of policy domains";
    leaf name {
      type string;
      mandatory true;
      description
        "Name of the organization or customer representing
        this domain.";
    }
    leaf address {
      type string;
      description
        "address of an organization or customer.";
    }
    leaf contact {
      type string;
      mandatory true;
      description
        "contact information of the organization
        or customer.";
    }
    leaf date {
      type yang:date-and-time;
      mandatory true;
      description
        "The date when this account was created
        or last modified.";
    }
    list policy-tenant {
      key "policy-tenant-id";
      leaf policy-tenant-id {
        type uint16;

```

```

        description
            "The policy tenant id.";
    }
    description
    "This represents the list of tenants";

    leaf name {
        type string;
        mandatory true;
        description
            "Name of the Department or Division within
            an organization.";
    }
    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
            "Date this account was created or last modified.";
    }
    leaf domain {
        type leafref {
            path "/multi-tenancy/policy-domain/policy-domain-id";
        }
        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 "/multi-tenancy/policy-mgmt-auth-method/policy-mgmt-auth-meth
od-id";
    }
    description
        "Authentication method to be used for this domain.
        It should be a reference to a 'policy-mgmt-auth-method'
        object.";
}
}
list policy-role {
    key "policy-role-id";

    leaf policy-role-id {
        type uint16;
        mandatory true;
        description
            "This defines a set of permissions assigned
            to a user in an organization that want to manage

```

```

        its own Security Policies.";
    }
    description
    "This represents the list of policy roles.";

    leaf name {
        type string;
        mandatory true;
        description
        "This field identifies name of the role.";
    }
    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
        "Date this role was created or last modified.";
    }
    leaf access-profile {
        type string;
        mandatory true;
        description
        "This field identifies the access profile for the
        role. The profile grants or denies access to policy
        objects. Multiple access profiles can be
        concatenated together.";
    }
}
list policy-user {
    key "policy-user-id";

    leaf policy-user-id {
        type uint16;
        description
        "This represents the policy-user-id.";
    }
    description
    "This represents the list of policy users.";
    leaf name {
        type string;
        mandatory true;
        description
        "The name of a user.";
    }
    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
        "Date this user was created or last modified";
    }
}

```

```

    }
    leaf password {
        type string;
        mandatory true;
        description
            "User password for basic authentication";
    }
    leaf email {
        type string;
        mandatory true;
        description
            "The email account of a user";
    }
    leaf scope-type {
        type string;
        description
            "identifies whether a user has domain-wide
            or tenant-wide privileges";
    }
    leaf scope-reference {
        type string;
        description
            "This references policy-domain or policy-tenant
            to identify the scope.";
    }
    leaf role {
        type string;
        mandatory true;
        description
            "This references policy-role to define specific
            permissions";
    }
}
list policy-mgmt-auth-method {
    key "policy-mgmt-auth-method-id";

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

```

```

        "name of the authentication method";
    }
    leaf date {
        type yang:date-and-time;
        mandatory true;
        description
            "date when the authentication method
            was created";
    }
    leaf authentication-method {
        type enumeration{
            enum password{
                description
                    "password-based authentication.";
            }
            enum token{
                description
                    "token-based authentication.";
            }
            enum certificate{
                description
                    "certificate-based authentication.";
            }
        }
        mandatory true;
        description
            "The description of authentication method;
            token-based, password, certificate,
            single-sign-on";
    }
    leaf mutual-authentication {
        type boolean;
        mandatory true;
        description
            "To identify whether the authentication
            is mutual";
    }
    leaf token-server {
        type inet:ipv4-address;
        mandatory true;
        description
            "The token-server information if the
            authentication method is token-based";
    }
    leaf certificate-server {
        type inet:ipv4-address;
        mandatory true;
        description

```

```
        "The certificate-server information if
        the authentication method is certificate-based";
    }
    leaf single-sing-on-server {
        type inet:ipv4-address;
        mandatory true;
        description
            "The single-sign-on-server information
            if the authentication method is
            single-sign-on-based";
    }
}
}
container endpoint-group {
    description
        "A logical entity in their business
        environment, where a security policy
        is to be applied.";

    list meta-data-source {
        key "meta-data-source-id";
        leaf meta-data-source-id {
            type uint16;
            mandatory true;
            description
                "This represents the meta-data source id.";
        }
        description
            "This represents the meta-data source.";

        leaf name {
            type string;
            mandatory true;
            description
                "This identifies the name of the
                meta-datas-ource.";
        }
        leaf date {
            type yang:date-and-time;
            mandatory true;
            description
                "This identifies the date this object was
                created or last modified.";
        }
        leaf tag-type {
            type boolean;
            description
                "This identifies the group type; user group,
```

```

        app group or device group.";
    }
    leaf tag-server-information {
        type inet:ipv4-address;
        description
            "The description of suthentication method;
            token-based, password, certificate,
            single-sign-on";
    }
    leaf tag-application-protocol {
        type string;
        description
            "This filed identifies the protocol e.g. LDAP,
            Active Directory, or CMDB";
    }
    leaf tag-server-credential {
        type string;
        description
            "This field identifies the credential
            information needed to access the tag server";
    }
}
list user-group{
    key "user-group-id";

    leaf user-group-id {
        type uint16;
        mandatory true;
        description
            "This represents the the user group id.";
    }
    description
        "This represents the user group.";

    leaf name {
        type string;
        description
            "This field identifies the name of user-group.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "when this user-group was created or last modified.";
    }
    leaf group-type {
        type enumeration{
            enum user-tag{
                description

```

```

        "The user group is based on user-tag.";
    }
    enum user-name{
        description
            "The user group is based on user-name.";
    }
    enum ip-address{
        description
            "The user group is based on ip-address.";
    }
    }
    description
        "This describes the group type; User-tag,
        User-name or IP-address.";
    }
    leaf meta-data-server {
        type inet:ipv4-address;
        description
            "This references metadata source";
    }

    leaf group-member {
        type string;
        description
            "This describes the user-tag information";
    }

    leaf risk-level {
        type uint16;
        description
            "This represents the threat level; valid range
            may be 0 to 9.";
    }
    }
    list device-group {
        key "device-group-id";
        leaf device-group-id {
            type uint16;
            description
                "This represents a device group id.";
        }
        description
            "This represents a device group.";
        leaf name {
            type string;
            description
                "This field identifies the name of
                a device-group.";
        }
    }

```

```

    }
    leaf date {
        type yang:date-and-time;
        description
            "The date when this group was create or
            last modified.";
    }
    leaf group-type {
        type enumeration{
            enum device-tag{
                description
                    "The device group is based on device-tag.";
            }
            enum device-name{
                description
                    "The device group is based on device-name.";
            }
            enum ip-address{
                description
                    "The device group is based on ip-address.";
            }
        }
        description
            "This describes the group type; device-tag,
            device-name or IP-address.";
    }
    leaf meta-data-server {
        type inet:ipv4-address;
        description
            "This references meta-data-source
            object.";
    }
    leaf group-member {
        type string;
        description
            "This describes the device-tag, device-name or
            IP-address information";
    }
    leaf risk-level {
        type uint16;
        description
            "This represents the threat level; valid range
            may be 0 to 9.";
    }
}
list application-group{
    key "application-group-id";
    leaf application-group-id {

```

```

type uint16;
description
  "This represents an application group id.";
}
description
  "This represents an application group.";
leaf name {
  type string;
  description
    "This field identifies the name of
    an application group";
}
leaf date {
  type yang:date-and-time;
  description
    "The date when this group was created or
    last modified.";
}
leaf group-type {
  type enumeration{
    enum application-tag{
      description
        "The application group is based on application-tag.";
    }
    enum device-name{
      description
        "The application group is based on application-name.";
    }
    enum ip-address{
      description
        "The application group is based on ip-address.";
    }
  }
  description
    "This identifies the group type;
    application-tag, application-name or
    IP-address.";
}
leaf meta-data-server {
  type inet:ipv4-address;
  description
    "This references meta-data-source
    object.";
}
leaf group-member {
  type string;
  description
    "This describes the application-tag,

```

```

        application-name or IP-address information";
    }
    leaf risk-level {
        type uint16;
        description
            "This represents the threat level; valid range
            may be 0 to 9.";
    }
}
list location-group{
    key "location-group-id";
    leaf location-group-id {
        type uint16;
        description
            "This represents a location group id.";
    }
    description
        "This represents a location group.";

    leaf name {
        type string;
        description
            "This field identifies the name of
            a location group";
    }
    leaf date {
        type yang:date-and-time;
        description
            "The date when this group was created or
            last modified.";
    }
}
leaf group-type {
    type enumeration{
        enum location-tag{
            description
                "The location group is based on location-tag.";
        }
        enum location-name{
            description
                "The location group is based on location-name.";
        }
        enum ip-address{
            description
                "The location group is based on ip-address.";
        }
    }
    description
        "This identifies the group type;

```

```

        location-tag, location-name or
        IP-address.";
    }
    leaf meta-data-server {
        type inet:ipv4-address;
        description
            "This references meta-data-source
            object.";
    }
    leaf group-member {
        type string;
        description
            "This describes the location-tag,
            location-name or IP-address information";
    }
    leaf risk-level {
        type uint16;
        description
            "This represents the threat level; valid range
            may be 0 to 9.";
    }
}
}
container threat-feed {
    description
        "this describes the list of threat-feed.";

    list threat-feed {
        key "threat-feed-id";
        leaf threat-feed-id {
            type uint16;
            mandatory true;
            description
                "This represents the threat-feed-id.";
        }
        description
            "This represents the threat feed within the
            threat-prevention-list.";
        leaf name {
            type string;
            description
                "Name of the theat feed.";
        }
        leaf date {
            type yang:date-and-time;
            description
                "when the threat-feed was created.";
        }
    }
}

```

```

leaf feed-type {
  type enumeration {
    enum unknown {
      description
        "feed-type is unknown.";
    }
    enum ip-address {
      description
        "feed-type is IP address.";
    }
    enum url {
      description
        "feed-type is URL.";
    }
  }
  mandatory true;
  description
    "This determined whether the feed-type is IP address
    based or URL based.";
}
leaf feed-server {
  type inet:ipv4-address;
  description
    "this contains threat feed server information.";
}
leaf feed-priority {
  type uint16;
  description
    "this describes the priority of the threat from
    0 to 5, where 0 means the threat is minimum and
    5 meaning the maximum.";
}
}
list custom-list {
  key "custom-list-id";
  leaf custom-list-id {
    type uint16;
    description
      "this describes the custom-list-id.";
  }
  description
    "this describes the threat-prevention custom list.";
  leaf name {
    type string;
    description
      "Name of the custom-list.";
  }
  leaf date {

```

```

    type yang:date-and-time;
    description
      "when the custom list was created.";
  }
  leaf list-type {
    type enumeration {
      enum unknown {
        description
          "list-type is unknown.";
      }
      enum ip-address {
        description
          "list-type is IP address.";
      }
      enum mac-address {
        description
          "list-type is MAC address.";
      }
      enum url {
        description
          "list-type is URL.";
      }
    }
    mandatory true;
    description
      "This determined whether the feed-type is IP address
      based or URL based.";
  }
  leaf list-property {
    type enumeration {
      enum unknown {
        description
          "list-property is unknown.";
      }
      enum blacklist {
        description
          "list-property is blacklist.";
      }
      enum whitelist {
        description
          "list-property is whitelist.";
      }
    }
    mandatory true;
    description
      "This determined whether the list-type is blacklist
      or whitelist.";
  }
}

```

```

    leaf list-content {
        type string;
        description
            "This describes the contents of the custom-list.";
    }
}
list malware-scan-group {
    key "malware-scan-group-id";
    leaf malware-scan-group-id {
        type uint16;
        mandatory true;
        description
            "This is the malware-scan-group-id.";
    }
    description
        "This represents the malware-scan-group.";
    leaf name {
        type string;
        description
            "Name of the malware-scan-group.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "when the malware-scan-group was created.";
    }
    leaf signature-server {
        type inet:ipv4-address;
        description
            "This describes the signature server of the
            malware-scan-group.";
    }
    leaf file-types {
        type string;
        description
            "This contains a list of file types needed to
            be scanned for the virus.";
    }
    leaf malware-signatures {
        type string;
        description
            "This contains a list of malware signatures or hash.";
    }
}
list event-map-group {
    key "event-map-group-id";
    leaf event-map-group-id {
        type uint16;

```

```

    mandatory true;
    description
    "This is the event-map-group-id.";
  }
  description
  "This represents the event map group.";

  leaf name {
    type string;
    description
    "Name of the event-map.";
  }
  leaf date {
    type yang:date-and-time;
    description
    "when the event-map was created.";
  }
  leaf security-events {
    type string;
    description
    "This contains a list of security events.";
  }
  leaf threat-map {
    type string;
    description
    "This contains a list of threat levels.";
  }
}
}
container telemetry-data {
  description
  "Telemetry provides visibility into the network
  activities which can be tapped for further
  security analytics, e.g., detecting potential
  vulnerabilities, malicious activities, etc.";

  list telemetry-data {
    key "telemetry-data-id";

    leaf telemetry-data-id {
      type uint16;
      mandatory true;
      description
      "This is ID for telemetry-data-id.";
    }
    description
    "This is ID for telemetry-data.";
    leaf name {

```

```

    type string;
    description
        "Name of the telemetry-data object.";
}
leaf date {
    type yang:date-and-time;
    description
        "This field states when the telemery-data
        object was created.";
}
leaf logs {
    type boolean;
    description
        "This field identifies whether logs
        need to be collected.";
}
leaf syslogs {
    type boolean;
    description
        "This field identifies whether System logs
        need to be collected.";
}
leaf snmp {
    type boolean;
    description
        "This field identifies whether 'SNMP traps' and
        'SNMP alarms' need to be collected.";
}
leaf sflow {
    type boolean;
    description
        "This field identifies whether 'sFlow' data
        need to be collected.";
}
leaf netflow {
    type boolean;
    description
        "This field identifies whether 'NetFlow' data
        need to be collected.";
}
leaf interface-stats {
    type boolean;
    description
        "This field identifies whether 'Interface' data
        such as packet bytes and counts need to be
        collected.";
}
}

```

```
list telemetry-source {
  key "telemetry-source-id";
  leaf telemetry-source-id {
    type uint16;
    mandatory true;
    description
      "This is ID for telemetry-source-id.";
  }
  description
    "This is ID for telemetry-source.";
  leaf name {
    type string;
    description
      "This identifies the name of this object.";
  }
  leaf date {
    type yang:date-and-time;
    description
      "Date this object was created or last modified";
  }
  leaf source-type {
    type enumeration {
      enum network-nsf {
        description
          "NSF telemetry source type is network-nsf.";
      }

      enum firewall-nsf {
        description
          "NSF telemetry source type is firewall-nsf.";
      }

      enum ids-nsf {
        description
          "NSF telemetry source type is ids-nsf.";
      }

      enum ips-nsf {
        description
          "NSF telemetry source type is ips-nsf.";
      }

      enum proxy-nsf {
        description
          "NSF telemetry source type is proxy-nsf.";
      }

      enum other-nsf {
        description
          "NSF telemetry source type is other-nsf.";
      }
    }
  }
}
```

```

description
    "This should have one of the following type of
    the NSF telemetry source: NETWORK-NSF,
    FIREWALL-NSF, IDS-NSF, IPS-NSF,
    PROXY-NSF, VPN-NSF, DNS, ACTIVE-DIRECTORY,
    IP Reputation Authority, Web Reputation
    Authority, Anti-Malware Sandbox, Honey Pot,
    DHCP, Other Third Party, ENDPOINT";
}
leaf nsf-source {
    type inet:ipv4-address;
    description
        "This field contains information such as
        IP address and protocol (UDP or TCP) port
        number of the NSF providing telemetry data.";
}
leaf nsf-credentials {
    type string;
    description
        "This field contains username and password
        to authenticate with the NSF.";
}
leaf collection-interval {
    type uint16;
    units seconds;
    default 5000;
    description
        "This field contains time in milliseconds
        between each data collection. For example,
        a value of 5000 means data is streamed to
        collector every 5 seconds. Value of 0 means
        data streaming is event-based";
}
leaf collection-method {
    type enumeration {
        enum unknown {
            description
                "collection-method is unknown.";
        }
        enum push-based {
            description
                "collection-method is PUSH-based.";
        }
        enum pull-based {
            description
                "collection-method is PULL-based.";
        }
    }
}

```

```

        description
            "This field contains a method of collection,
            i.e., whether it is PUSH-based or PULL-based.";
    }
    leaf heartbeat-interval {
        type uint16;
        units seconds;
        description
            "time in seconds the source sends telemetry
            heartbeat.";
    }
    leaf qos-marking {
        type uint16;
        description
            "DSCP value must be contained in this field.";
    }
}
list telemetry-destination {
    key "telemetry-destination-id";

    leaf telemetry-destination-id {
        type uint16;
        description
            "this represents the telemetry-destination-id";
    }
    description
        "This object contains information related to
        telemetry destination. The destination is
        usually a collector which is either a part of
        Security Controller or external system
        such as Security Information and Event
        Management (SIEM).";
    leaf name {
        type string;
        description
            "This identifies the name of this object.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "Date this object was created or last
            modified";
    }
    leaf collector-source {
        type inet:ipv4-address;
        description
            "This field contains information such as
            IP address and protocol (UDP or TCP) port

```

```

        number for the collector's destination.";
    }
    leaf collector-credentials {
        type string;
        description
            "This field contains the username and
            password for the collector.";
    }
    leaf data-encoding {
        type string;
        description
            "This field contains the telemetry data encoding
            in the form of schema.";
    }
    leaf data-transport {
        type enumeration{
            enum grpc {
                description
                    "telemetry data protocol is grpc.";
            }
            enum buffer-over-udp{
                description
                    "telemetry data protocol is buffer over UDP.";
            }
        }
        description
            "This field contains streaming telemetry data
            protocols. This could be gRPC, protocol
            buffer over UDP, etc.";
    }
}
}
}
}
<CODE ENDS>

```

Figure 2: YANG for policy-general

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

A common scenario for VoIP/VoLTE policy enforcement could be that a malicious call is made to a benign user of any telecommunication company. For example, imagine a case where a company "A" employs a hacker with a malicious attempt to hack a user's phone with malware. The company "A" is located in a country, such as Africa, and uses the user's hacked phone to call the company. The hacked user is unaware of the company "A" so complains about the international call that was made to the company "B", which is the user's telecommunications company. The company "A" charges the company "B" for the

international call. The company "B" cannot charge the user for the call, and has no choice but to pay the company "A". The following shows the example data tree model for the VoIP/VoLTE services. Multi-tenancy, endpoint groups, threat prevention, and telemetry data components are general part of the tree model, so we can just modify the policy instance in order to generate and enforce high-level policies. The policy-calendar can act as a scheduler to set the start and end time to block calls which uses suspicious ids, or calls from other countries.

```

module: policy-voip
  +--rw policy-voip
    +--rw rule-voip* [rule-voip-id]
      +--rw rule-voip-id      uint16
      +--rw name?             string
      +--rw date?             yang:date-and-time
      +--rw event* [event-id]
        +--rw event-id        string
        +--rw name?           string
        +--rw date?           yang:date-and-time
        +--rw event-type?     string
        +--rw Time-Information? string
        +--rw event-map-group? -> /threat-feed/event-map-group
                                   /event-map-group-id
        +--rw enable?         boolean
      +--rw condition* [condition-id]
        +--rw condition-id    string
        +--rw source-caller?  -> /threat-feed/threat-feed
                                   /threat-feed-id
        +--rw destination-callee? -> /threat-feed/custom-list
                                   /custom-list-id
        +--rw match?          boolean
        +--rw match-direction? string
        +--rw exception?      string
      +--rw action* [action-id]
        +--rw action-id       string
        +--rw name?           string
        +--rw date?           yang:date-and-time
        +--rw primary-action? string
        +--rw secondary-action? string
        +--rw precedence?     uint16
      +--rw owner* [owner-id]
        +--rw owner-id        string
        +--rw name?           string
        +--rw date?           yang:date-and-time
    +--rw threat-feed
      +--rw threat-feed* [threat-feed-id]
        +--rw threat-feed-id  uint16

```

```

    +--rw name?                string
    +--rw date?                yang:date-and-time
    +--rw feed-type            enumeration
    +--rw feed-server?        inet:ipv4-address
    +--rw feed-priority?      uint16
+--rw custom-list* [custom-list-id]
    +--rw custom-list-id      uint16
    +--rw name?                string
    +--rw date?                yang:date-and-time
    +--rw list-type            enumeration
    +--rw list-property        enumeration
    +--rw list-content?       string
+--rw malware-scan-group* [malware-scan-group-id]
    +--rw malware-scan-group-id uint16
    +--rw name?                string
    +--rw date?                yang:date-and-time
    +--rw signature-server?    inet:ipv4-address
    +--rw file-types?          string
    +--rw malware-signatures?  string
+--rw event-map-group* [event-map-group-id]
    +--rw event-map-group-id   uint16
    +--rw name?                string
    +--rw date?                yang:date-and-time
    +--rw security-events?     string
    +--rw threat-map?          string

```

Figure 3: Policy Instance Example for VoIP/VoLTE Security Services

5.1. Policy Instance YANG Example for VoIP/VoLTE Security Services

The following YANG data model is a policy instance for VoIP/VoLTE security services. The policy-calendar can act as a scheduler to set the start time and end time to block malicious calls which use suspicious IDs, or calls from other countries.

```

<CODE BEGINS> file "ietf-i2nsf-cf-interface-voip.yang"

module ietf-policy-voip {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-policy-voip";
  prefix
    "cf-interface";

  import ietf-yang-types{
    prefix yang;
  }

```

```
import ietf-inet-types{
  prefix inet;
}
organization
  "IETF I2NSF (Interface to Network Security Functions)
  Working Group";

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

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

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

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

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

revision "2018-11-04"{
  description "sixth revision";
  reference
    "draft-kumar-i2nsf-client-facing-interface-im-07";
}
container policy-voip {
  description
    "This object is a policy instance to have
    complete information such as where and when
    a policy need to be applied.";
  list rule-voip {
    key "rule-voip-id";
    leaf rule-voip-id {
      type uint16;
      mandatory true;
      description
        "This is ID for rules.";
    }
    description
      "This is a container for rules.";
    leaf name {
      type string;
      description
        "This field identifies the name of this object.";
    }
  }
}
```

```

    }
    leaf date {
        type yang:date-and-time;
        description
            "Date this object was created or last
            modified";
    }
}
list event {
    key "event-id";
    description
        "This represents the security event of a
        policy-rule.";
    leaf event-id {
        type string;
        mandatory true;
        description
            "This represents the event-id.";
    }
    leaf name {
        type string;
        description
            "This field identifies the name of this object.";
    }
    leaf date {
        type yang:date-and-time;
        description
            "Date this object was created or last
            modified";
    }
    leaf event-type {
        type string;
        description
            "This field identifies the event event type
            .";
    }
    leaf Time-Information {
        type string;
        description
            "This field contains time calendar such as
            BEGIN-TIME and END-TIME for one time
            enforcement or recurring time calendar for
            periodic enforcement.";
    }
    leaf event-map-group {
        type leafref{
            path "/threat-feed/event-map-group/event-map-group-id";
        }
        description

```

```

        "This field contains security events or threat
        map in order to determine when a policy need
        to be activated. This is a reference to
        Evnet-Map-Group.";
    }
    leaf enable {
        type boolean;
        description
            "This determines whether the condition
            matches the security event or not.";
    }
}
list condition {
    key "condition-id";
    description
        "This represents the condition of a
        policy-rule.";
    leaf condition-id {
        type string;
        description
            "This represents the condition-id.";
    }
    leaf source-caller {
        type leafref {
            path "/threat-feed/threat-feed/threat-feed-id";
        }
        description
            "This field identifies the source of
            the traffic. This could be reference to
            either 'Policy Endpoint Group' or
            'Threat-Feed' or 'Custom-List' if Security
            Admin wants to specify the source; otherwise,
            the default is to match all traffic.";
    }
    leaf destination-callee {
        type leafref {
            path "/threat-feed/custom-list/custom-list-id";
        }
        description
            "This field identifies the source of
            the traffic. This could be reference to
            either 'Policy Endpoint Group' or
            'Threat-Feed' or 'Custom-List' if Security
            Admin wants to specify the source; otherwise,
            the default is to match all traffic.";
    }
    leaf match {
        type boolean;

```

```
    description
      "This field identifies the match criteria used to
      evaluate whether the specified action need to be
      taken or not. This could be either a Policy-
      Endpoint-Group identifying a Application set or a
      set of traffic rules.";
  }
  leaf match-direction {
    type string;
    description
      "This field identifies if the match criteria is
      to evaluated for both direction of the traffic or
      only in one direction with default of allowing in
      the other direction for stateful match conditions.
      This is optional and by default rule should apply
      in both directions.";
  }
  leaf exception {
    type string;
    description
      "This field identifies the exception
      consideration when a rule is evaluated for a
      given communication. This could be reference to
      Policy-Endpoint-Group object or set of traffic
      matching criteria.";
  }
}
list action {
  key "action-id";
  leaf action-id {
    type string;
    mandatory true;
    description
      "this represents the policy-action-id.";
  }
  description
    "This object represents actions that a
    Security Admin wants to perform based on
    a certain traffic class.";
  leaf name {
    type string;
    description
      "The name of the policy-action object.";
  }
}

leaf date {
  type yang:date-and-time;
  description
```

```

        "When the object was created or last
        modified.";
    }
    leaf primary-action {
        type string;
        description
            "This field identifies the action when a rule
            is matched by NSF. The action could be one of
            'PERMIT', 'DENY', 'RATE-LIMIT', 'TRAFFIC-CLASS',
            'AUTHENTICATE-SESSION', 'IPS', 'APP-FIREWALL', etc.";
    }
    leaf secondary-action {
        type string;
        description
            "This field identifies additional actions if
            a rule is matched. This could be one of 'LOG',
            'SYSLOG', 'SESSION-LOG', etc.";
    }
}
leaf precedence {
    type uint16;
    description
        "This field identifies the precedence
        assigned to this rule by Security Admin.
        This is helpful in conflict resolution
        when two or more rules match a given
        traffic class.";
}
}
list owner {
    key "owner-id";
    leaf owner-id {
        type string;
        mandatory true;
        description
            "this represents the owner-id.";
    }
    description
        "This field defines the owner of this policy.
        Only the owner is authorized to modify the
        contents of the policy.";
    leaf name {
        type string;
        description
            "The name of the owner.";
    }
    leaf date {
        type yang:date-and-time;
    }
}

```

```

        description
            "When the object was created or last
            modified.";
    }
}
}
container threat-feed {
    description
        "this describes the list of threat-feed.";

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

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

        leaf feed-type {
            type enumeration {
                enum unknown {
                    description
                        "feed-type is unknown.";
                }
                enum ip-address {
                    description
                        "feed-type is IP address.";
                }
                enum url {
                    description
                        "feed-type is URL.";
                }
            }
        }
    }
}

```

```
        mandatory true;
        description
            "This determined whether the feed-type is IP address
            based or URL based.";
    }

    leaf feed-server {
        type inet:ipv4-address;
        description
            "this contains threat feed server information.";
    }

    leaf feed-priority {
        type uint16;
        description
            "this describes the priority of the threat from
            0 to 5, where 0 means the threat is minimum and
            5 meaning the maximum.";
    }
}
list custom-list {
    key "custom-list-id";
    leaf custom-list-id {
        type uint16;
        description
            "this describes the custom-list-id.";
    }
    description
        "this describes the threat-prevention custom list.";
    leaf name {
        type string;
        description
            "Name of the custom-list.";
    }

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

    leaf list-type {
        type enumeration {
            enum unknown {
                description
                    "list-type is unknown.";
            }
            enum ip-address {
```

```

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

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

```

```

    mandatory true;
    description
    "This is the malware-scan-group-id.";
  }
  description
  "This represents the malware-scan-group.";
  leaf name {
    type string;
    description
    "Name of the malware-scan-group.";
  }

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

  leaf signature-server {
    type inet:ipv4-address;
    description
    "This describes the signature server of the
    malware-scan-group.";
  }

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

  leaf malware-signatures {
    type string;
    description
    "This contains a list of malware signatures or hash.";
  }
}
list event-map-group {
  key "event-map-group-id";
  leaf event-map-group-id {
    type uint16;
    mandatory true;
    description
    "This is the event-map-group-id.";
  }
  description
  "This represents the event map group.";
}

```

```

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

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

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

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

```

Figure 4: Policy Instance YANG Example for VoIP Security Services

6. Example XML Output for Various Use Cases

In this section, we present an XML example for various use cases. Here, we show the policy examples that can be delivered through consumer-facing interface. For now, the considered use cases are: VoIP security service, DDoS-attack mitigation, time-based firewall, and web-filter.

6.1. Case 1: VoIP Security Service

The first example is a VoIP policy. Here, we are going to drop calls commin from a country with an Ip from South Africa that is classified as malicious. The below figure shows the XML document generated by using the YANG data tree as shown in the previous section.

```

<?xml version="1.1" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:restconf:base:1.0">

```

```

<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <i2nsf-cf-interface-voip-req nc:operation="create">
      <policy-voip>
        <rule-voip>
          <rule-voip-id>01</rule-voip-id>
          <rule-voip-name>voip-policy-example</rule-voip-name>
          <rule-voip-date>2017.10.25/20:30:32</rule-voip-date>
          <event>
            <event-id>01</event-id>
            <event-name>voip_call</event-name>
            <event-date>2017.10.25/20:30:32</event-date>
            <event-type>malicious</event-type>
            <event-map-group>19</event-map-group>
            <enable>True</enable>
          </event>
          <condition>
            <condition-id>01</condition-id>
            <source-caller>105.176.0.0</source-caller>
            <destination-callee>192.168.171.35</destination-callee>
            <time-information>
              <begin-time>22:00</begin-time>
              <end-time>08:00</end-time>
            </time-information>
            <match-direction>default</match-direction>
            <exeption>00</exeption>
          </condition>
          <action>
            <action-id>01</action-id>
            <action-name>action-voip</action-name>
            <action-date>2017.10.25/20:30:32</action-date>
            <primary-action>DENY</primary-action>
            <secondary-action>LOG</secondary-action>
          </action>
          <precedence>none</precedence>
        <owner>
          <owner-id>01</owner-id>
          <name>i2nsf-admin</name>
        </owner>
      </rule-voip>
    </policy-voip>
  </i2nsf-cf-interface-voip-req>
</config>
</edit-config>
</rpc>

```

Figure 5: An XML Example for VoIP Security Service

6.2. Case 2: DDoS-Attack Mitigation

The second example is 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.

```
<?xml version="1.1" encoding="UTF-8"?>
<rpc message-id="2" xmlns="urn:ietf:params:xml:ns:restconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <i2nsf-cf-interface-ddos-req nc:operation="create">
        <policy-ddos>
          <rule-ddos>
            <rule-ddos-id>03</rule-ddos-id>
            <rule-ddos-name>ddos-policy-example</rule-ddos-name>
            <rule-ddos-date>2018.10.25/11:25:32</rule-ddos-date>
            <event>
              <event-id>03</event-id>
              <event-name>ddos</event-name>
              <event-date>2018.10.25/11:25:32</event-date>
              <event-type>ddos</event-type>
              <event-map-group>03</event-map-group>
              <enable>True</enable>
            </event>
            <condition>
              <condition-id>03</condition-id>
              <source-ip>Any</source-ip>
              <destination-ip>192.168.173.37</destination-ip>
              <threshold>30</threshold>
              <time-information>
                <begin-time>--:--</begin-time>
                <end-time>--:--</end-time>
              </time-information>
              <match-direction>default</match-direction>
              <exeption>00</exeption>
            </condition>
            <action>
              <action-id>03</action-id>
              <action-name>action-ddos</action-name>
              <action-date>2018.10.25/11:25:32</action-date>
            </action>
          </rule-ddos>
        </policy-ddos>
      </i2nsf-cf-interface-ddos-req>
    </config>
  </edit-config>
</rpc>
```

```

        <primary-action>REJECT</primary-action>
        <secondary-action>LOG</secondary-action>
    </action>
    <precedence>none</precedence>
  <owner>
    <owner-id>03</owner-id>
    <name>i2nsf-admin</name>
  </owner>
</rule-ddos>
</policy-ddos>
</i2nsf-cf-interface-ddos-req>
</config>
</edit-config>
</rpc>

```

Figure 6: An XML Example for DDoS-attack Mitigation

6.3. Case 3: Time-Based Firewall

The third example is a time-based firewall policy. Consider a Smart Factory which operates from 9 am to 7 pm during the working days. During these hours, only the admin responsible for operating the factory is allow to access a control system. The below figure show that any access during outside the operating hours is rejected.

```

<?xml version="1.1" encoding="UTF-8"?>
<rpc message-id="3" xmlns="urn:ietf:params:xml:ns:restconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <i2nsf-cf-interface-fw-req nc:operation="create">
        <policy-fw>
          <rule-fw>
            <rule-fw-id>01</rule-fw-id>
            <rule-fw-name>fw-policy-example</rule-fw-name>
            <rule-fw-date>2018.10.25/11:19:05</rule-fw-date>
            <event>
              <event-id>01</event-id>
              <event-name>invalid_access</event-name>
              <event-date>2018.10.25/11:19:05</event-date>
              <event-type>invalid</event-type>
              <event-map-group>02</event-map-group>
              <enable>True</enable>
            </event>
          <condition>
            <condition-id>02</condition-id>

```

```

        <source-ip>115.176.0.1</source-ip>
        <destination-ip>192.168.173.41</destination-ip>
        <time-information>
            <begin-time>09:00</begin-time>
            <end-time>17:00</end-time>
        </time-information>
        <match-direction>default</match-direction>
        <exeption>00</exeption>
    </condition>
    <action>
        <action-id>02</action-id>
        <action-name>action-fw</action-name>
        <action-date>2018.10.25/11:19:05</action-date>
        <primary-action>PASS</primary-action>
        <secondary-action>LOG</secondary-action>
    </action>
    <precedence>none</precedence>
    <owner>
        <owner-id>02</owner-id>
        <name>i2nsf-admin</name>
    </owner>
    </rule-fw>
</policy-fw>
</i2nsf-cf-interface-fw-req>
</config>
</edit-config>
</rpc>

```

Figure 7: An XML Example for Time-based Firewall

6.4. Case 4: Time-Based Web-Filter

The last example is a time-based web-filter policy. Let us suppose that a owner of an enterprise wants to forbid access to a specific set of websites, such as Facebook, Youtube, Instagram, etc. The below figure shows an example policy an admin of a sector or department can deploy.

```

<?xml version="1.1" encoding="UTF-8"?>
<rpc message-id="4" xmlns="urn:ietf:params:xml:ns:restconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <i2nsf-cf-interface-wf-req nc:operation="create">
        <policy-wf>
          <rule-wf>

```

```

<rule-wf-id>03</rule-wf-id>
<rule-wf-name>wf-policy-example</rule-wf-name>
<rule-wf-date>2018.10.26/14:03:17</rule-wf-date>
<event>
  <event-id>04</event-id>
  <event-name>wf</event-name>
  <event-date>2018.10.26/14:03:17</event-date>
  <event-type>wf</event-type>
  <event-map-group>04</event-map-group>
  <enable>True</enable>
</event>
<condition>
  <condition-id>04</condition-id>
  <source-ip>192.168.1.3</source-ip>
  <destination-url>www.facebook.com</destination-url>
  <time-information>
    <begin-time>09:00</begin-time>
    <end-time>18:00</end-time>
  </time-information>
  <match-direction>default</match-direction>
  <exeption>00</exeption>
</condition>
<action>
  <action-id>04</action-id>
  <action-name>action-wf</action-name>
  <action-date>2018.10.26/14:03:17</action-date>
  <primary-action>REJECT</primary-action>
  <secondary-action>LOG</secondary-action>
</action>
<precedence>none</precedence>
<owner>
  <owner-id>03</owner-id>
  <name>i2nsf-admin</name>
</owner>
</rule-wf>
</policy-wf>
</i2nsf-cf-interface-wf-req>
</config>
</edit-config>
</rpc>

```

Figure 8: An XML Example for Time-based Web-filter

6.5. Case 5: Threat-Feed Configuration

The threat-feed container described above can convey various sources containing information concerning security threats. One good example can be STIX. STIX (Structured Threat Information Expression) is a

language and serialization format used to exchange cyber threat intelligence (CTI). It is a language to describe threat information in a standardized format to enable exchanging and sharing them. The below figure shows the necessary configuration, which can be generated and delivered by consumer-facing interface.

```

...
...
  <configuration-tf>
    <threat-feed>
      <threat-feed-id>02</threat-feed-id>
      <threat-feed-name>stix</threat-feed-name>
      <threat-feed-date>2018.10.25/11:25:32</threat-feed-date>
      <threat-feed-type>ip-address</threat-feed-type>
      <feed-server>105.134.171.24</feed-server>
      <feed-priority>ip-address</feed-priority>
    </threat-feed>
  </configuration-tf>
...
...

```

Figure 9: An XML Example for Threat-feed Configuration

Usually, STIX can be obtained from a TAXII server which contains a collection of cyber threat information formatted in STIX. Here, the "feed-server" leaf contains the ip-address of the TAXII server, so that recent threat related information can be collected when the configuration is set.

7. 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.

8. References

8.1. Normative References

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

8.2. Informative References

[client-facing-inf-im]

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

[client-facing-inf-req]

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

[i2nsf-framework]

Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 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-06 (work in progress), July 2018.

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

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

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

- o In Section 6, four additional XML output examples (VoIP, DDoS-attack, Time-based Firewall and Web-filter) for security policies are added. Also, an example XML output for Threat-feed configuration is added using STIX and TAXII as a threat-feed example.
- o The overall organization of the YANG data model and its data types have also been reviewed and corrected, and produced the corresponding data tree as shown in the Sections 4 and 5.
- o Overall editorial errors have been corrected.

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 following are considered co-authors:

- o Hyoungshick Kim (Sungkyunkwan University)
- o Seungjin Lee (Sungkyunkwan University)
- o Jinyong Tim Kim (Sungkyunkwan University)

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@endzh.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

J. Kim
J. Jeong
Sungkyunkwan University
J. Park
ETRI
S. Hares
Q. Lin
Huawei
July 02, 2018

I2NSF Network Security Function-Facing Interface YANG Data Model
draft-ietf-i2nsf-nsf-facing-interface-dm-01

Abstract

This document defines a YANG data model corresponding to the information model for Network Security Functions (NSF) facing interface in Interface to Network Security Functions (I2NSF). It describes a data model for the features provided by generic security functions. This data model provides generic components whose vendors is well understood, so that the generic component can be used even if it has some vendor specific functions. These generic functions represent a point of interoperability, and can be provided by any product that offers the required Capabilities. Also, if vendors need additional features for its network security function, they can add the features by extending the YANG data model.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 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. The Structure and Objective of I2NSF Security Policy	4
4.1. I2NSF Security Policy Rule	4
4.2. Event Clause	4
4.3. Condition Clause	4
4.4. Action Clause	5
5. Data Model Structure	5
5.1. I2NSF Security Policy Rule	5
5.2. Event Clause	7
5.3. Condition Clause	8
5.4. Action Clause	10
6. YANG Module	12
6.1. IETF NSF-Facing Interface YANG Data Module	12
7. Security Considerations	46
8. Acknowledgments	46
9. Contributors	47
10. References	47
10.1. Normative References	47
10.2. Informative References	47
Appendix A. Changes from draft-ietf-i2nsf-nsf-facing-interface-dm-01	49
Authors' Addresses	49

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of security services with the information model for Network Security Functions (NSF) facing interface in Interface to

Network Security Functions (I2NSF). It provides a specific information model and the corresponding data models for generic network security functions (i.e., network security functions), as defined in [i2nsf-nsf-cap-im]. With these data model, I2NSF controller can control the capabilities of NSFs.

The "Event-Condition-Action" (ECA) policy model is used as the basis for the design of I2NSF Policy Rules.

The "ietf-i2nsf-nsf-facing-interface" YANG module defined in this document provides the following features:

- o configuration of I2NSF security policy rule for generic network security function policy
- o configuration of event clause for generic network security function policy
- o configuration of condition clause for generic network security function policy
- o configuration of action clause for generic network security function policy

2. Requirements Language

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

3. Terminology

This document uses the terminology described in [i2nsf-nsf-cap-im][i2rs-rib-data-model][supa-policy-info-model]. Especially, the following terms are from [supa-policy-info-model]:

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

3.1. Tree Diagrams

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

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

4. The Structure and Objective of I2NSF Security Policy

4.1. I2NSF Security Policy Rule

This shows a policy rule for generic network security functions. The object of a policy rule is defined as policy information and rule information. This includes ECA Policy Rule such as Event Clause Objects, Condition Clause Objects, Action Clause Objects, Resolution Strategy, and Default Action.

4.2. Event Clause

This shows an event clause for generic network security functions. An Event is any important occurrence in time of a change in the system being managed, and/or in the environment of the system being managed. When used in the context of I2NSF Policy Rules, it is used to determine whether the Condition clause of the I2NSF Policy Rule can be evaluated or not. The object of an event clauses is defined as user security event, device security event, system security event, and time security event. The objects of event clauses can be extended according to specific vendor event features.

4.3. Condition Clause

This shows a condition clause for generic network security functions. 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. These objects are defined as packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. The objects of action clauses can be extended according to specific vendor condition features.

4.4. Action Clause

This shows an action clause 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 functions by executing various Actions. The object of an action clause is defined as ingress action, egress action, and apply profile action. The objects of action clauses can be extended according to specific vendor action features.

5. Data Model Structure

This section shows a data model structure tree of generic network security functions that are defined in the [i2nsf-nsf-cap-im].

- o Consideration of ECA Policy Model by Aggregating the Event, Condition, and Action Clauses Objects.
- o Consideration of Capability Algebra.
- o Consideration of NSFs Capability Categories (i.e., Network Security, Content Security, and Attack Mitigation Capabilities).
- o Definitions for Network Security Event Class, Network Security Condition Class, and Network Security Action Class.

5.1. I2NSF Security Policy Rule

The data model for the identification of network security policy has the following structure:

```

module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy
|   +--rw policy-name?          string
|   +--rw rules* [rule-name]
|   |   +--rw rule-name          string
|   |   +--rw rule-description?  string
|   |   +--rw rule-priority?     uint8
|   |   +--rw enable?           boolean
|   |   +--rw session-aging-time? uint16
|   |   +--rw long-connection

```

```

|   |--rw enable?    boolean
|   |--rw during?   uint16
+--rw policy-event-clause-aggr-ptr*    instance-identifier
+--rw policy-condition-clause-aggr-ptr* instance-identifier
+--rw policy-action-clause-aggr-ptr*   instance-identifier
+--rw time-zone
  |--rw absolute-time-zone
  |   |--rw time
  |   |   |--rw start-time? yang:date-and-time
  |   |   |--rw end-time?   yang:date-and-time
  |   |--rw date
  |   |   |--rw absolute-date? yang:date-and-time
  |--rw periodic-time-zone
  |   |--rw day
  |   |   |--rw sunday?      boolean
  |   |   |--rw monday?     boolean
  |   |   |--rw tuesday?    boolean
  |   |   |--rw wednesday?  boolean
  |   |   |--rw thursday?   boolean
  |   |   |--rw friday?     boolean
  |   |   |--rw saturday?   boolean
  |   |--rw month
  |   |   |--rw january?    boolean
  |   |   |--rw february?   boolean
  |   |   |--rw march?     boolean
  |   |   |--rw april?     boolean
  |   |   |--rw may?       boolean
  |   |   |--rw june?      boolean
  |   |   |--rw july?      boolean
  |   |   |--rw august?    boolean
  |   |   |--rw september? boolean
  |   |   |--rw october?   boolean
  |   |   |--rw november?  boolean
  |   |   |--rw december?  boolean
+--rw resolution-strategy
  |--rw (resolution-strategy-type)?
  |   |--:(fmr)
  |   |   |--rw first-matching-rule? boolean
  |   |--:(lmr)
  |   |   |--rw last-matching-rule?  boolean
+--rw default-action
  |--rw default-action-type? boolean
+--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
|           +--rw description?      string
+--rw event-clause-container
|   ...
+--rw condition-clause-container
|   ...
+--rw action-clause-container
|   ...

```

Figure 1: Data Model Structure for Network Security Policy Identification

5.2. Event Clause

The data model for event rule has the following structure:

```

module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|   ...
|   +--rw eca-policy-rules* [rule-id]
|   |   ...
|   |   +--rw resolution-strategy
|   |   |   ...
|   |   +--rw default-action
|   |   |   ...
+--rw event-clause-container
|   +--rw event-clause-list* [eca-object-id]
|   |   +--rw entity-class?           identityref
|   |   +--rw eca-object-id           string
|   |   +--rw description?            string
|   |   +--rw sec-event-content       string
|   |   +--rw sec-event-format        sec-event-format
|   |   +--rw sec-event-type          string
+--rw condition-clause-container
|   ...
+--rw action-clause-container
|   ...

```

Figure 2: Data Model Structure for Event Rule

These objects are defined as user security event, device security event, system security event, and time security event. These objects can be extended according to specific vendor event features. We will add additional event objects for more generic network security functions.

5.3. Condition Clause

The data model for condition rule has the following structure:

```

module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|
|   ...
|   +--rw eca-policy-rules* [rule-id]
|   |
|   |   ...
|   |   +--rw resolution-strategy
|   |   |
|   |   |   ...
|   |   +--rw default-action
|   |   |
|   |   |   ...
+--rw event-clause-container
|
|   ...
+--rw condition-clause-container
|
|   +--rw condition-clause-list* [eca-object-id]
|   |
|   |   +--rw entity-class?                identityref
|   |   +--rw eca-object-id                string
|   |   +--rw packet-security-condition
|   |   |
|   |   |   +--rw packet-description?      string
|   |   |   +--rw packet-security-mac-condition
|   |   |   |
|   |   |   |   +--rw pkt-sec-cond-mac-dest*   yang:phys-address
|   |   |   |   +--rw pkt-sec-cond-mac-src*    yang:phys-address
|   |   |   |   +--rw pkt-sec-cond-mac-8021q*  string
|   |   |   |   +--rw pkt-sec-cond-mac-ether-type* string
|   |   |   |   +--rw pkt-sec-cond-mac-tci*   string
|   |   |   +--rw packet-security-ipv4-condition
|   |   |   |
|   |   |   |   +--rw pkt-sec-cond-ipv4-header-length*  uint8
|   |   |   |   +--rw pkt-sec-cond-ipv4-tos*            uint8
|   |   |   |   +--rw pkt-sec-cond-ipv4-total-length*   uint16
|   |   |   |   +--rw pkt-sec-cond-ipv4-id*             uint8
|   |   |   |   +--rw pkt-sec-cond-ipv4-fragment*      uint8
|   |   |   |   +--rw pkt-sec-cond-ipv4-fragment-offset* uint16
|   |   |   |   +--rw pkt-sec-cond-ipv4-ttl*           uint8
|   |   |   |   +--rw pkt-sec-cond-ipv4-protocol*      uint8
|   |   |   |   +--rw pkt-sec-cond-ipv4-src*           inet:ipv4-address
|   |   |   |   +--rw pkt-sec-cond-ipv4-dest*         inet:ipv4-address
|   |   |   |   +--rw pkt-sec-cond-ipv4-ipopts?       string
|   |   |   |   +--rw pkt-sec-cond-ipv4-sameip?       boolean
|   |   |   |   +--rw pkt-sec-cond-ipv4-geoip*        string
|   |   |   +--rw packet-security-ipv6-condition
|   |   |   |
|   |   |   |   +--rw pkt-sec-cond-ipv6-dscp*         string
|   |   |   |   +--rw pkt-sec-cond-ipv6-ecn*          string
|   |   |   |   +--rw pkt-sec-cond-ipv6-traffic-class* uint8
|   |   |   |   +--rw pkt-sec-cond-ipv6-flow-label*   uint32
|   |   |   |   +--rw pkt-sec-cond-ipv6-payload-length* uint16
|   |   |   |   +--rw pkt-sec-cond-ipv6-next-header*  uint8

```



```

|         |   +--rw group
|         |   |   +--rw (group-name)?
|         |   |   |   +--:(tenant)
|         |   |   |   |   +--rw tenant      uint8
|         |   |   |   |   +--:(vn-id)
|         |   |   |   |   |   +--rw vn-id    uint8
|         |   |   |   |   |   +--rw security-grup      string
|         |   |   |   |   |   +--rw url-category-condition
|         |   |   |   |   |   |   +--rw pre-defined-category*   string
|         |   |   |   |   |   |   +--rw user-defined-category*   string
|         |   |   |   |   |   |   +--rw context-condition
|         |   |   |   |   |   |   |   +--rw context-description?  string
|         |   |   |   |   |   |   |   +--rw gen-context-condition
|         |   |   |   |   |   |   |   |   +--rw gen-context-description?  string
|         |   |   |   |   |   |   |   |   +--rw geographic-location
|         |   |   |   |   |   |   |   |   |   +--rw src-geographic-location*   uint32
|         |   |   |   |   |   |   |   |   |   +--rw dest-geographic-location*   uint32
|         |   |   |   |   |   |   |   |   |   +--rw action-clause-container
|         |   |   |   |   |   |   |   |   |   ...

```

Figure 3: Data Model Structure for Condition Rule

These objects are defined as packet security condition, packet payload security condition, target security condition, user security condition, context condition, and generic context condition. These objects can be extended according to specific vendor condition features. We will add additional condition objects for more generic network security functions.

5.4. Action Clause

The data model for action rule has the following structure:

```

module: ietf-i2nsf-policy-rule-for-nsf
+--rw i2nsf-security-policy* [policy-name]
|   ...
|   +--rw eca-policy-rules* [rule-id]
|   ...
|   +--rw resolution-strategy
|   ...
|   +--rw default-action
|   ...
+--rw event-clause-container
|   ...
+--rw condition-clause-container
|   ...
+--rw action-clause-container
|   +--rw action-clause-list* [eca-object-id]

```

```

+--rw entity-class?      identityref
+--rw eca-object-id     string
+--rw rule-log?         boolean
+--rw session-log?      boolean
+--rw ingress-action
|   +--rw ingress-description?  string
|   +--rw ingress-action-type?  ingress-action
+--rw egress-action
|   +--rw egress-description?   string
|   +--rw egress-action-type?   egress-action
+--rw apply-profile
+--rw profile-description?      string
+--rw content-security-control
|   +--rw content-security-control-types
|   |   +--rw antivirus?        string
|   |   +--rw ips?              string
|   |   +--rw ids?              string
|   |   +--rw url-filtering?    string
|   |   +--rw data-filtering?   string
|   |   +--rw mail-filtering?   string
|   |   +--rw file-blocking?    string
|   |   +--rw file-isolate?     string
|   |   +--rw pkt-capture?      string
|   |   +--rw application-control? string
|   |   +--rw voip-volte?       string
+--rw attack-mitigation-control
+--rw ddos-attack
|   +--rw ddos-attack-type
|   |   +--rw network-layer-ddos-attack
|   |   |   +--rw network-layer-ddos-attack-type
|   |   |   |   +--rw syn-flood?      string
|   |   |   |   +--rw udp-flood?     string
|   |   |   |   +--rw icmp-flood?    string
|   |   |   |   +--rw ip-frag-flood?  string
|   |   |   |   +--rw ipv6-related?   string
|   |   +--rw app-layer-ddos-attack
|   |   |   +--rw app-ddos-attack-types
|   |   |   |   +--rw http-flood?    string
|   |   |   |   +--rw https-flood?   string
|   |   |   |   +--rw dns-flood?     string
|   |   |   |   +--rw dns-amp-flood?  string
|   |   |   |   +--rw ssl-ddos?      string
+--rw single-packet-attack
+--rw single-packet-attack-type
+--rw scan-and-sniff-attack
|   +--rw scan-and-sniff-attack-types
|   |   +--rw ip-sweep?            string
|   |   +--rw port-scanning?      string

```

```

+--rw malformed-packet-attack
|   +--rw malformed-packet-attack-types
|       +--rw ping-of-death?   string
|       +--rw teardrop?        string
+--rw special-packet-attack
    +--rw special-packet-attack-types
        +--rw oversized-icmp?  string
        +--rw tracert?         string

```

Figure 4: Data Model Structure for Action Rule

These objects are defined as ingress action, egress action, and apply profile action. These objects can be extended according to specific vendor action feature. We will add additional action objects for more generic network security functions.

6. YANG Module

6.1. IETF NSF-Facing Interface YANG Data Module

This section introduces a YANG module for the information model of network security functions, as defined in the [i2nsf-nsf-cap-im].

<CODE BEGINS> file "ietf-i2nsf-policy-rule-for-nsf@2018-07-02.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
    policy-rule-for-nsf;

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

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

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

     WG Chair: Adrian Farrel

```

<mailto:Adrain@olddog.co.uk>

WG Chair: Linda Dunbar
<mailto:Linda.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.";
revision "2018-07-02" {
  description "The fourth revision";
  reference
    "draft-ietf-i2nsf-capability-00";
}

typedef sec-event-format {
  type enumeration {
    enum unknown {
      description
        "If SecEventFormat is unknown";
    }
    enum guid {
      description
        "If SecEventFormat is GUID
        (Generic Unique Identifier)";
    }
    enum uuid {
      description
        "If SecEventFormat is UUID
        (Universal Unique Identifier)";
    }
    enum uri {
      description
        "If SecEventFormat is URI
        (Uniform Resource Identifier)";
    }
    enum fqdn {
      description
        "If SecEventFormat is FQDN
        (Fully Qualified Domain Name)";
    }
  }
}
```

```
    }
    enum fqpn {
        description
            "If SecEventFormat is FQPN
            (Fully Qualified Path Name)";
    }
}
description
    "This is used for SecEventFormat.";
}

typedef ingress-action {
    type enumeration {
        enum pass {
            description
                "If ingress action is pass";
        }
        enum drop {
            description
                "If ingress action is drop";
        }
        enum reject {
            description
                "If ingress action is reject";
        }
        enum alert {
            description
                "If ingress action is alert";
        }
        enum mirror {
            description
                "If ingress action is mirror";
        }
    }
    description
        "This is used for ingress action.";
}

typedef egress-action {
    type enumeration {
        enum invoke-signaling {
            description
                "If egress action is invoke signaling";
        }
        enum tunnel-encapsulation {
            description
                "If egress action is tunnel encapsulation";
        }
    }
}
```

```
        enum forwarding {
            description
                "If egress action is forwarding";
        }
        enum redirection {
            description
                "If egress action is redirection";
        }
    }
    description
        "This is used for egress action.";
}

identity ECA-OBJECT-TYPE {
    description "TBD";
}

identity ECA-EVENT-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
}

identity ECA-CONDITION-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
}

identity ECA-ACTION-TYPE {
    base ECA-OBJECT-TYPE;
    description "TBD";
}

identity EVENT-USER-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
}

identity EVENT-DEV-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
}

identity EVENT-SYS-TYPE {
    base ECA-EVENT-TYPE;
    description "TBD";
}

identity EVENT-TIME-TYPE {
```

```
    base ECA-EVENT-TYPE;
    description "TBD";
}

grouping i2nsf-eca-object-type {
  leaf entity-class {
    type identityref {
      base ECA-OBJECT-TYPE;
    }
    description "TBD";
  }
  leaf eca-object-id {
    type string;
    description "TBD";
  }
  description "TBD";
}

grouping i2nsf-event-type {
  description "TBD";
  leaf description {
    type string;
    description
      "This is description for event.
      Vendors can write instructions for event
      that vendor made";
  }

  leaf sec-event-content {
    type string;
    mandatory true;
    description
      "This is a mandatory string that contains the content
      of the SecurityEvent. The format of the content
      is specified in the SecEventFormat class
      attribute, and the type of event is defined in the
      SecEventType class attribute. An example of the
      SecEventContent attribute is a string hrAdmin,
      with the SecEventFormat set to 1 (GUID) and the
      SecEventType attribute set to 5 (new logon).";
  }

  leaf sec-event-format {
    type sec-event-format;
    mandatory true;
    description
      "This is a mandatory uint 8 enumerated integer, which
```

```
        is used to specify the data type of the
        SecEventContent attribute. The content is
        specified in the SecEventContent class attribute,
        and the type of event is defined in the
        SecEventType class attribute. An example of the
        SecEventContent attribute is string hrAdmin,
        with the SecEventFormat attribute set to 1 (GUID)
        and the SecEventType attribute set to 5
        (new logon).";
    }

    leaf sec-event-type {
        type string;
        mandatory true;
        description
            "This is a mandatory uint 8 enumerated integer, which
            is used to specify the type of event that involves
            this user. The content and format are specified in
            the SecEventContent and SecEventFormat class
            attributes, respectively. An example of the
            SecEventContent attribute is string hrAdmin,
            with the SecEventFormat attribute set to 1 (GUID)
            and the SecEventType attribute set to 5
            (new logon).";
    }
}

container i2nsf-security-policy {
    description
        "policy is a container
        including a set of security rules according to certain logic,
        i.e., their similarity or mutual relations, etc. The network
        security policy is able to apply over both the unidirectional
        and bidirectional traffic across the NSF.";

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

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

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

leaf 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.";
    }
}

leaf-list policy-event-clause-agg-ptr {
    type instance-identifier;
    must 'derived-from-or-self (/event-clause-container/
event-clause-list/entity-class, "ECA-EVENT-TYPE")';
    description
        "TBD";
}

leaf-list policy-condition-clause-agg-ptr {
    type instance-identifier;
    must 'derived-from-or-self (/condition-clause-container/
condition-clause-list/entity-class, "ECA-CONDITION-TYPE")';
    description
        "TBD";
}

leaf-list policy-action-clause-agg-ptr {
    type instance-identifier;
    must 'derived-from-or-self (/action-clause-container/
action-clause-list/entity-class, "ECA-ACTION-TYPE")';
    description
        "TBD";
}

container time-zone {
    description
        "This can be used to apply rules according to time-zone";
    container absolute-time-zone {
        description
            "This can be used to apply rules according to
absolute-time";
        container time {
            description
                "This can be used to apply rules according to time";
            leaf start-time {
                type yang:date-and-time;
                description
                    "This is start time for time zone";
            }
            leaf end-time {
                type yang:date-and-time;
                description
                    "This is end time for time zone";
            }
        }
    }
}
```

```
    }
    container date {
      description
        "This can be used to apply rules according to date";
      leaf absolute-date {
        type yang:date-and-time;
        description
          "This is absolute date for time zone";
      }
    }
  }
}
container periodic-time-zone {
  description
    "This can be used to apply rules according to
    periodic-time-zone";
  container day {
    description
      "This can be used to apply rules according
      to periodic day";
    leaf sunday {
      type boolean;
      description
        "This is sunday for periodic day";
    }
    leaf monday {
      type boolean;
      description
        "This is monday for periodic day";
    }
    leaf tuesday {
      type boolean;
      description
        "This is tuesday for periodic day";
    }
    leaf wednesday {
      type boolean;
      description
        "This is wednesday for periodic day";
    }
    leaf thursday {
      type boolean;
      description
        "This is thursday for periodic day";
    }
    leaf friday {
      type boolean;
      description
        "This is friday for periodic day";
    }
  }
}
```

```
    }
    leaf saturday {
        type boolean;
        description
            "This is saturday for periodic day";
    }
}
container month {
    description
        "This can be used to apply rules according
        to periodic month";
    leaf january {
        type boolean;
        description
            "This is january for periodic month";
    }
    leaf february {
        type boolean;
        description
            "This is february for periodic month";
    }
    leaf march {
        type boolean;
        description
            "This is march for periodic month";
    }
    leaf april {
        type boolean;
        description
            "This is april for periodic month";
    }
    leaf may {
        type boolean;
        description
            "This is may for periodic month";
    }
    leaf june {
        type boolean;
        description
            "This is june for periodic month";
    }
    leaf july {
        type boolean;
        description
            "This is july for periodic month";
    }
    leaf august {
        type boolean;
```

```
        description
            "This is august for periodic month";
    }
    leaf september {
        type boolean;
        description
            "This is september for periodic month";
    }
    leaf october {
        type boolean;
        description
            "This is october for periodic month";
    }
    leaf november {
        type boolean;
        description
            "This is november for periodic month";
    }
    leaf december {
        type boolean;
        description
            "This is december for periodic month";
    }
    }
}

container resolution-strategy {
    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";

    choice resolution-strategy-type {
        description
            "Vendors can use YANG data model to configure rules";

        case fmr {
            leaf first-matching-rule {
                type boolean;
                description
                    "If the resolution strategy is first matching rule";
            }
        }
        case lmr {
```

```
        leaf last-matching-rule {
            type boolean;
            description
                "If the resolution strategy is last matching rule";
        }
    }
}

container default-action {
    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.";

    leaf default-action-type {
        type boolean;
        description
            "True is permit
            False is deny.";
    }
}

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";
}
}
}

container event-clause-container {
    description "TBD";
    list event-clause-list {
        key eca-object-id;
        uses i2nsf-eca-object-type {
            refine entity-class {
                default ECA-EVENT-TYPE;
            }
        }
    }
}

description
    " This is abstract. 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.).";

    uses i2nsf-event-type;
}
}
container condition-clause-container {
```

```
description "TBD";
list condition-clause-list {
  key eca-object-id;
  uses i2nsf-eca-object-type {
    refine entity-class {
      default ECA-CONDITION-TYPE;
    }
  }
}
description
  " This is abstract. 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.";

container packet-security-condition {
  description
    "TBD";
  leaf packet-description {
    type string;
    description
      "This is description for packet condition.
      Vendors can write instructions for packet condition
      that vendor made";
  }
}

container packet-security-mac-condition {
  description
    "The purpose of this Class is to represent packet MAC
    packet header information that can be used as part of
    a test to determine if the set of Policy Actions in
    this ECA Policy Rule should be execute or not.";

  leaf-list pkt-sec-cond-mac-dest {
    type yang:phys-address;
    description
      "The MAC destination address (6 octets long).";
  }

  leaf-list pkt-sec-cond-mac-src {
    type yang:phys-address;
    description
      "The MAC source address (6 octets long).";
  }
}
```

```
leaf-list pkt-sec-cond-mac-8021q {
  type string;
  description
    "This is an optional string attribute, and defines
    The 802.1Q tag value (2 octets long).";
}

leaf-list pkt-sec-cond-mac-ether-type {
  type string;
  description
    "The EtherType field (2 octets long). Values up to
    and including 1500 indicate the size of the
    payload in octets; values of 1536 and above
    define which protocol is encapsulated in the
    payload of the frame.";
}

leaf-list pkt-sec-cond-mac-tci {
  type string;
  description
    "This is an optional string attribute, and defines
    the Tag Control Information. This consists of a 3
    bit user priority field, a drop eligible indicator
    (1 bit), and a VLAN identifier (12 bits).";
}
}

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

  leaf-list pkt-sec-cond-ipv4-header-length {
    type uint8;
    description
      "The IPv4 packet header consists of 14 fields,
      of which 13 are required.";
  }

  leaf-list pkt-sec-cond-ipv4-tos {
    type uint8;
    description
      "The ToS field could specify a datagram's priority
      and request a route for low-delay,
      high-throughput, or highly-reliable service..";
  }
}
```

```
    }  
  
    leaf-list pkt-sec-cond-ipv4-total-length {  
        type uint16;  
        description  
            "This 16-bit field defines the entire packet size,  
            including header and data, in bytes.";  
    }  
  
    leaf-list pkt-sec-cond-ipv4-id {  
        type uint8;  
        description  
            "This field is an identification field and is  
            primarily used for uniquely identifying  
            the group of fragments of a single IP datagram.";  
    }  
  
    leaf-list pkt-sec-cond-ipv4-fragment {  
        type uint8;  
        description  
            "IP fragmentation is an Internet Protocol (IP)  
            process that breaks datagrams into smaller pieces  
            (fragments), so that packets may be formed that  
            can pass through a link with a smaller maximum  
            transmission unit (MTU) than the original  
            datagram size.";  
    }  
  
    leaf-list pkt-sec-cond-ipv4-fragment-offset {  
        type uint16;  
        description  
            "Fragment offset field along with Don't Fragment  
            and More Fragment flags in the IP protocol  
            header are used for fragmentation and reassembly  
            of IP datagrams.";  
    }  
  
    leaf-list pkt-sec-cond-ipv4-ttl {  
        type uint8;  
        description  
            "The ttl keyword is used to check for a specific  
            IP time-to-live value in the header of  
            a packet.";  
    }  
  
    leaf-list pkt-sec-cond-ipv4-protocol {  
        type uint8;  
        description
```

```
        "Internet Protocol version 4(IPv4) is the fourth
        version of the Internet Protocol (IP).";
    }

    leaf-list pkt-sec-cond-ipv4-src {
        type inet:ipv4-address;
        description
            "Defines the IPv4 Source Address.";
    }

    leaf-list pkt-sec-cond-ipv4-dest {
        type inet:ipv4-address;
        description
            "Defines the IPv4 Destination Address.";
    }

    leaf pkt-sec-cond-ipv4-ipopts {
        type string;
        description
            "With the ipopts keyword you can check if
            a specific ip option is set. Ipopts has
            to be used at the beginning of a rule.";
    }

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

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

container packet-security-ipv6-condition {
    description
        "The purpose of this Class is to represent packet
        IPv6 packet header information that can be used as
```

```
part of a test to determine if the set of Policy
Actions in this ECA Policy Rule should be executed
or not.";

leaf-list pkt-sec-cond-ipv6-dscp {
  type string;
  description
    "Differentiated Services Code Point (DSCP)
    of ipv6.";
}

leaf-list pkt-sec-cond-ipv6-ecn {
  type string;
  description
    "ECN allows end-to-end notification of network
    congestion without dropping packets.";
}

leaf-list pkt-sec-cond-ipv6-traffic-class {
  type uint8;
  description
    "The bits of this field hold two values. The 6
    most-significant bits are used for
    differentiated services, which is used to
    classify packets.";
}

leaf-list pkt-sec-cond-ipv6-flow-label {
  type uint32;
  description
    "The flow label when set to a non-zero value
    serves as a hint to routers and switches
    with multiple outbound paths that these
    packets should stay on the same path so that
    they will not be reordered.";
}

leaf-list pkt-sec-cond-ipv6-payload-length {
  type uint16;
  description
    "The size of the payload in octets,
    including any extension headers.";
}

leaf-list pkt-sec-cond-ipv6-next-header {
  type uint8;
  description
    "Specifies the type of the next header.
```

```
        This field usually specifies the transport
        layer protocol used by a packet's payload.";
    }

    leaf-list pkt-sec-cond-ipv6-hop-limit {
        type uint8;
        description
            "Replaces the time to live field of IPv4.";
    }

    leaf-list pkt-sec-cond-ipv6-src {
        type inet:ipv6-address;
        description
            "The IPv6 address of the sending node.";
    }

    leaf-list pkt-sec-cond-ipv6-dest {
        type inet:ipv6-address;
        description
            "The IPv6 address of the destination node(s).";
    }
}

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

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

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

    leaf-list pkt-sec-cond-tcp-seq-num {
        type uint32;
        description
```

```
        "If the SYN flag is set (1), then this is the
        initial sequence number.";
    }

    leaf-list pkt-sec-cond-tcp-ack-num {
        type uint32;
        description
            "If the ACK flag is set then the value of this
            field is the next sequence number that the sender
            is expecting.";
    }

    leaf-list pkt-sec-cond-tcp-window-size {
        type uint16;
        description
            "The size of the receive window, which specifies
            the number of windows size units
            (by default,bytes) (beyond the segment
            identified by the sequence number in the
            acknowledgment field) that the sender of this
            segment is currently willing to receive.";
    }

    leaf-list pkt-sec-cond-tcp-flags {
        type uint8;
        description
            "This is a mandatory string attribute, and defines
            the nine Control bit flags (9 bits).";
    }
}

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

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

    leaf-list pkt-sec-cond-udp-dest-port {
        type inet:port-number;
        description
```

```
        "This is a mandatory string attribute, and
        defines the UDP Destination Port number (16 bits).";
    }

    leaf-list pkt-sec-cond-udp-length {
        type string;
        description
            "This is a mandatory string attribute, and defines
            the length in bytes of the UDP header and data
            (16 bits).";
    }
}

container packet-security-icmp-condition {
    description
        "The internet control message protocol condition.";

    leaf-list pkt-sec-cond-icmp-type {
        type uint8;
        description
            "ICMP type, see Control messages.";
    }

    leaf-list pkt-sec-cond-icmp-code {
        type uint8;
        description
            "ICMP subtype, see Control messages.";
    }

    leaf-list pkt-sec-cond-icmp-seg-num {
        type uint32;
        description
            "The icmp Sequence Number.";
    }
}

container packet-payload-condition {
    description
        "TBD";
    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 acl-number {
    type uint32;
    description
        "This is acl-number.";
}

container application-condition {
    description
        "TBD";
    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
        "TBD";
    list application-category {
        key "name application-subcategory";
        description
            "TBD";
        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
        "TBD";
    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 pc {
        type boolean;
        description
            "If type of a device is PC.";
    }

    leaf mobile-phone {
        type boolean;
        description
            "If type of a device is mobile-phone.";
    }

    leaf voip-volte-phone {
        type boolean;
        description
            "If type of a device is voip-volte-phone.";
    }

    leaf tablet {
```

```
        type boolean;
        description
            "If type of a device is tablet.";
    }

    leaf iot {
        type boolean;
        description
            "If type of a device is Internet of Things.";
    }

    leaf vehicle {
        type boolean;
        description
            "If type of a device is vehicle.";
    }
}

container users-condition {
    description
        "TBD";
    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
        "TBD";
    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
      "TBD";
    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
      "TBD";
    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 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
        the database.";
    }
  }
}
```

```
    }
  }
}
}
container action-clause-container {
  description "TBD";
  list action-clause-list {
    key eca-object-id;
    uses i2nsf-eca-object-type {
      refine entity-class {
        default ECA-ACTION-TYPE;
      }
    }
  }
  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.";

  leaf rule-log {
    type boolean;
    description
      "True is enable
      False is not enable.";
  }
  leaf session-log {
    type boolean;
    description
      "True is enable
      False is not enable.";
  }
  container ingress-action {
    description
      "TBD";
    leaf ingress-description {
      type string;
      description
        "This is description for ingress action.
        Vendors can write instructions for ingress action
        that vendor made";
    }
    leaf ingress-action-type {
      type ingress-action;
      description
        "Ingress action type: permit, deny, and mirror.";
    }
  }
}
```

```
    }
  }
  container egress-action {
    description
      "TBD";
    leaf egress-description {
      type string;
      description
        "This is description for egress action.
        Vendors can write instructions for egress action
        that vendor made";
    }
    leaf egress-action-type {
      type egress-action;
      description
        "Egress-action-type: invoke-signaling,
        tunnel-encapsulation, and forwarding.";
    }
  }
}

container apply-profile {
  description
    "TBD";
  leaf profile-description {
    type string;
    description
      "This is description for apply profile action.
      Vendors can write instructions for apply
      profile action that vendor made";
  }
}

container content-security-control {
  description
    "Content security control is another category of
    security capabilities applied to application layer.
    Through detecting the contents carried over the
    traffic in application layer, these capabilities
    can realize various security purposes, such as
    defending against intrusion, inspecting virus,
    filtering malicious URL or junk email, and blocking
    illegal web access or data retrieval.";

  container content-security-control-types {
    description
      "Content Security types: Antivirus, IPS, IDS,
      url-filtering, data-filtering, mail-filtering,
      file-blocking, file-isolate, pkt-capture,
      application-control, and voip-volte.";
  }
}
```

```
leaf antivirus {
    type string;
    description
        "Additional inspection of antivirus.";
}

leaf ips {
    type string;
    description
        "Additional inspection of IPS.";
}

leaf ids {
    type string;
    description
        "Additional inspection of IDS.";
}

leaf url-filtering {
    type string;
    description
        "Additional inspection of URL filtering.";
}

leaf data-filtering {
    type string;
    description
        "Additional inspection of data filtering.";
}

leaf mail-filtering {
    type string;
    description
        "Additional inspection of mail filtering.";
}

leaf file-blocking {
    type string;
    description
        "Additional inspection of file blocking.";
}

leaf file-isolate {
    type string;
    description
        "Additional inspection of file isolate.";
}
```

```
leaf pkt-capture {
  type string;
  description
    "Additional inspection of packet capture.";
}

leaf application-control {
  type string;
  description
    "Additional inspection of app control.";
}

leaf voip-volte {
  type string;
  description
    "Additional inspection of VoIP/VoLTE.";
}
}

container attack-mitigation-control {
  description
    "This category of security capabilities is
    specially used to detect and mitigate various
    types of network attacks.";

  container ddos-attack {
    description
      "A distributed-denial-of-service (DDoS) is
      where the attack source is more than one,
      often thousands of unique IP addresses.";

    container ddos-attack-type {
      description
        "DDoS-attack types: Network Layer
        DDoS Attacks and Application Layer
        DDoS Attacks.";

      container network-layer-ddos-attack {
        description
          "Network layer DDoS-attack.";
        container network-layer-ddos-attack-type {
          description
            "Network layer DDoS attack types:
            Syn Flood Attack, UDP Flood Attack,
            ICMP Flood Attack, IP Fragment Flood,
            IPv6 Related Attacks, and etc";
        }
      }
    }
  }
}
```

```
leaf syn-flood {
  type string;
  description
    "Additional Inspection of
    Syn Flood Attack.";
}

leaf udp-flood {
  type string;
  description
    "Additional Inspection of
    UDP Flood Attack.";
}

leaf icmp-flood {
  type string;
  description
    "Additional Inspection of
    ICMP Flood Attack.";
}

leaf ip-frag-flood {
  type string;
  description
    "Additional Inspection of
    IP Fragment Flood.";
}

leaf ipv6-related {
  type string;
  description
    "Additional Inspection of
    IPv6 Related Attacks.";
}
}

container app-layer-ddos-attack {
  description
    "Application layer DDoS-attack.";

  container app-ddos-attack-types {
    description
      "Application layer DDoS-attack types:
      Http Flood Attack, Https Flood Attack,
      DNS Flood Attack, and
      DNS Amplification Flood Attack,
      SSL DDoS Attack, and etc.";
  }
}
```

```
    leaf http-flood {
      type string;
      description
        "Additional Inspection of
        Http Flood Attack.";
    }

    leaf https-flood {
      type string;
      description
        "Additional Inspection of
        Https Flood Attack.";
    }

    leaf dns-flood {
      type string;
      description
        "Additional Inspection of
        DNS Flood Attack.";
    }

    leaf dns-amp-flood {
      type string;
      description
        "Additional Inspection of
        DNS Amplification Flood Attack.";
    }

    leaf ssl-ddos {
      type string;
      description
        "Additional Inspection of
        SSL Flood Attack.";
    }
  }
}

container single-packet-attack {
  description
    "Single Packet Attacks.";
  container single-packet-attack-type {
    description
      "DDoS-attack types: Scanning Attack,
      Sniffing Attack, Malformed Packet Attack,
      Special Packet Attack, and etc.";
  }
}
```

```
container scan-and-sniff-attack {
  description
    "Scanning and Sniffing Attack.";
  container scan-and-sniff-attack-types {
    description
      "Scanning and sniffing attack types:
      IP Sweep attack, Port Scanning,
      and etc.";

    leaf ip-sweep {
      type string;
      description
        "Additional Inspection of
        IP Sweep Attack.";
    }

    leaf port-scanning {
      type string;
      description
        "Additional Inspection of
        Port Scanning Attack.";
    }
  }
}

container malformed-packet-attack {
  description
    "Malformed Packet Attack.";
  container malformed-packet-attack-types {
    description
      "Malformed packet attack types:
      Ping of Death Attack, Teardrop Attack,
      and etc.";

    leaf ping-of-death {
      type string;
      description
        "Additional Inspection of
        Ping of Death Attack.";
    }

    leaf teardrop {
      type string;
      description
        "Additional Inspection of
        Teardrop Attack.";
    }
  }
}
```


Technology Development for the Customized Security Service Provisioning).

9. 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 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)

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

10.2. Informative References

- [i2nsf-nsf-cap-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-ietf-i2nsf-capability-00 (work in progress), September 2017.
- [i2rs-rib-data-model] Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-10 (work in progress), February 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. Changes from draft-ietf-i2nsf-nsf-facing-interface-dm-01

The following changes are made from draft-ietf-i2nsf-nsf-facing-interface-dm-00:

1. We added rule enable, session aging time, and long connection attributes.
2. We added a rule group attribute.
3. We added additional conditions such as application and url.
4. We replaced manual to description to clarify the meaning.

Authors' Addresses

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

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

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

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

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 34129
Republic of Korea

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

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

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

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

EMail: linqiushi@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2019

S. Hyun
Chosun University
J. Jeong
T. Roh
S. Wi
Sungkyunkwan University
J. Park
ETRI
October 20, 2018

I2NSF Registration Interface Data Model
draft-ietf-i2nsf-registration-interface-dm-00

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 search, instantiation and registration according to required security capabilities via I2NSF Registration Interface.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2019.

Copyright Notice

Copyright (c) 2018 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

NSFs of some required security capabilities on demand. As an example, if additional security capabilities are required to meet the new security requirements that an I2NSF user requests, 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 [RFC6020] 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 search, instantiation and registration according to required security capabilities. It also describes the procedure 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 Advanced Inspection/Action: As like the I2NSF information model for NSF facing interface [capability-im], Advanced Inspection/Action means that a security function calls another security function for further inspection based on its own inspection result. [nsf-triggered-steering]
- o NSF Profile (NSF Capability Information): NSF Capability Information specifies the inspection capabilities of the associated NSF instance. Each NSF instance has its own NSF

Capability Information to specify the type of security service it provides and its resource capacity etc. [nsf-triggered-steering]

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

4. Objectives

- o Registering NSF instances from Developer's Management System: Depending on system's security requirements, it may require some NSFs by default. In this case, DMS creates these default NSF instances and notifies Security Controller of those NSF instances via Registration Interface.
- o Requesting NSF instances with required security capabilities: I2NSF users request security policies to Security Controller, and enforcing the security policies requires a set of security capabilities. In addition, when an NSF triggers another type of NSF(s) for more advanced security inspection of a given traffic, some security capabilities are also required to perform the advanced security inspection. If Security Controller has no NSF instance registered with the required capabilities, Security Controller requests DMS for new NSF instances that can provide the required capabilities. Once receiving this request, DMS could first search its inventory for NSF instances with the required capabilities. If DMS fails to find any NSF instance, it creates new NSF instances with the required security capabilities and registers the NSF instances to Security Controller.
- o Deleting unnecessary NSF instances: In I2NSF framework, users decide which security service is unnecessary in the system. If there exist any unused NSF instances, then we should delete the existing instances by requesting DMS via registration interface.
- o Updating NSF instances: After an NSF instance is registered into I2NSF framework, some changes may happen on the capability of the NSF instance. These changes should be informed to Security Controller. For this, after updating some NSF instances, DMS notifies Security Controller of the update via registration interface.

5. Information Model

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

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

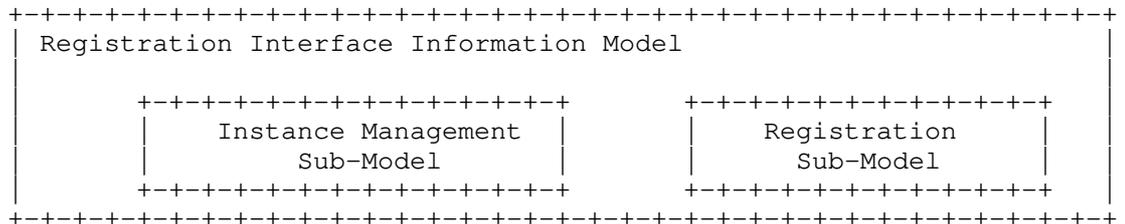


Figure 1: Registration Interface Information Model

As illustrated in Figure 1, the information model for Registration Interface consists of two sub-models: instance management, registration sub-models. The instance management functionality and the registration functionality use NSF Profile to achieve their goals. In this context, NSF Profile is the capability objects that describe and/or prescribe inspection capability an NSF instance can provide.

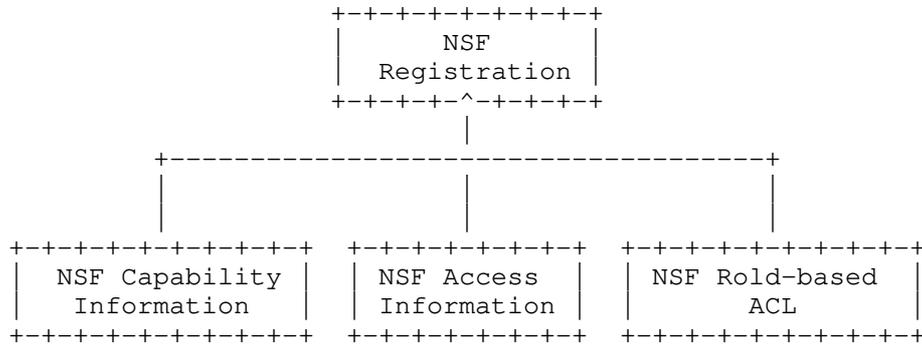


Figure 3: Registration Mechanism Sub-Model Overview

5.3. NSF Access Information

NSF Access Information contains the followings that are required to communicate with an NSF: IPv4 address, IPv6 address, port number, and supported transport protocol(s) (e.g., Virtual Extensible LAN (VXLAN) [RFC 7348], Generic Protocol Extension for VXLAN (VXLAN-GPE) [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).

5.4. NSF Capability Information (Capabilities of an NSF instance)

NSF Profile basically describes the inspection capabilities of an NSF instance. In Figure 4, we show capability objects of an NSF instance. 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 Profile additionally contains the performance capabilities and role-Based access control list (ACL) as shown in Figure 4.

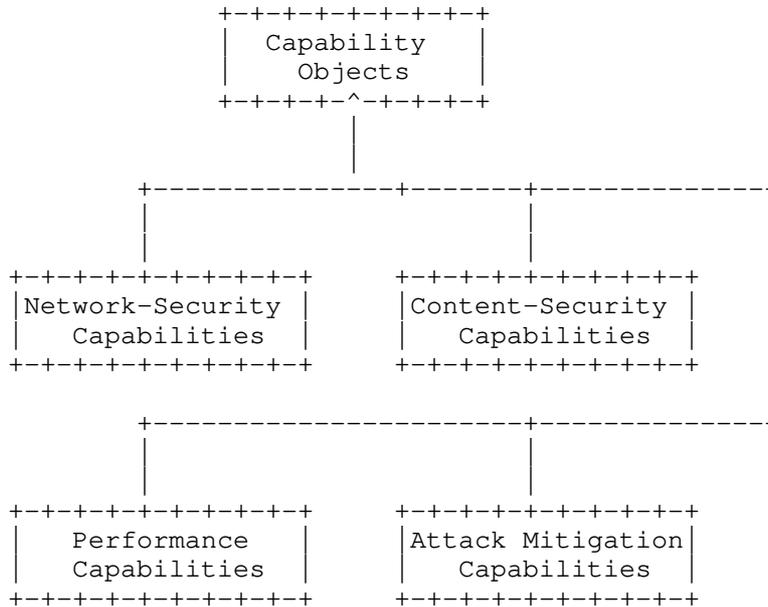


Figure 4: NSF Profile Overview

5.4.1. 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 5, 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.

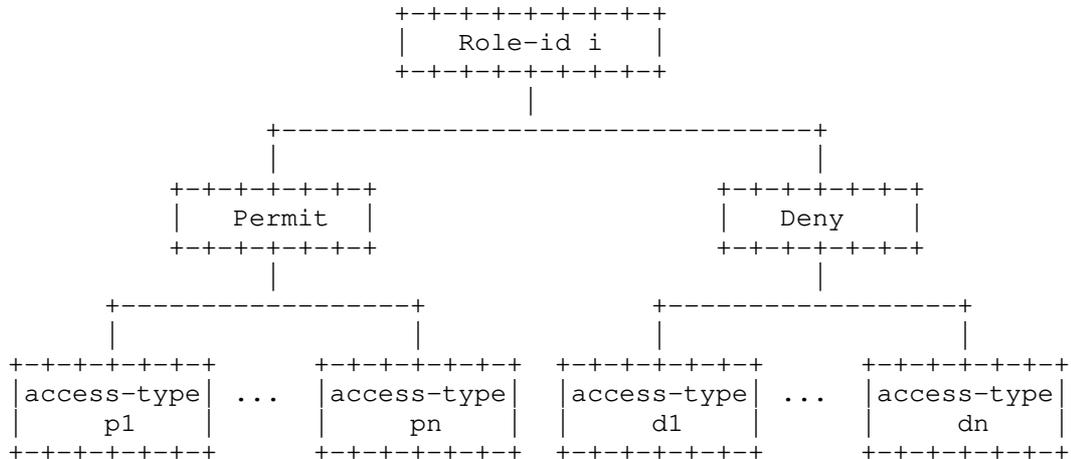


Figure 7: Role-id Subtree

6. Data Model

6.1. High-Level YANG

This section provides an overview of the high level YANG.

6.1.1. 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 [i2rs-rib-data-model] is as follows:

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

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

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

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

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

6.1.2. Registration Interface

```
module : ietf-i2nsf-regs-interface-model
  +--rw regs-req
  |   uses i2nsf-regs-req
  +--rw instance-mgmt-req
  |   uses i2nsf-instance-mgmt-req
```

Figure 8: High-Level YANG of I2NSF Registration Interface

Each of these sections mirror sections of Section 5.

6.1.3. Registration Request

This section expands the i2nsf-regs-req in Figure 8.

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

Figure 9: High-Level YANG of I2NSF Registration Request

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

6.1.4. Instance Management Request

This section expands the i2nsf-instance-mgmt-req in Figure 8.

```

Instance Management Request
+--rw i2nsf-instance-mgmt-req
  +--rw req-level uint16
  +--rw req-id uint64
  +--rw (req-type)?
    +--rw (instanciation-request)
      +--rw in-nsf-capability-information
        |   uses i2nsf-nsf-capability-information
    +--rw (deinstanciation-request)
      +--rw de-nsf-access-info
        |   uses i2nsf-nsf-access-info
    +--rw (updating-request)
      +--rw update-nsf-capability-information
        |   uses i2nsf-nsf-capability-information

```

Figure 10: High-Level YANG of I2NSF Instance Mgmt Request

Instance management request consists of two types: `instanciation-request`, `deinstanciation-request`, and `updating-request`. The `instanciation-request` is used to request generation of a new NSF instance with NSF Capability Information which specifies required NSF capability information. The `deinstanciation-request` is used to remove an existing NSF with NSF Access Information. The `updating nsf` request is used to updating a existing NSF information with NSF capabilities.

6.1.5. NSF Capability Information

This section expands the `i2nsf-nsf-capability-information` in Figure 9 and Figure 10.

```

NSF Capability Information
+--rw i2nsf-nsf-capability-information
  +--rw i2nsf-capability
    |   uses ietf-i2nsf-capability
  +--rw performance-capability
    |   uses i2nsf-nsf-performance-caps

```

Figure 11: High-Level YANG of I2NSF NSF Capability Information

In Figure 11, `ietf-i2nsf-capability` refers module `ietf-i2nsf-capability` in `[i2nsf-capability-dm]`. We add the `performance-capability` because it is absent in `[i2nsf-capability-dm]` and `[netmod-acl-model]`

6.1.6. NSF Access Information

This section expands the `i2nsf-nsf-access-info` in Figure 9 and Figure 10.

```

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

Figure 12: High-Level YANG of I2NSF NSF Access Information

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

6.1.7. NSF Performance Capability

This section expands the `i2nsf-nsf-performance-caps` in Figure 11.

```

NSF Performance Capability
  +--rw i2nsf-nsf-performance-caps
    +--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 13: High-Level YANG of I2NSF NSF Performance Capability

When the Security Controller requests the Developer Management System to create a new NSF instance, the performance capability is used to specify the performance requirements of the new instance.

6.1.8. Role-Based ACL (Access Control List)

This section expands the `ietf-netmod-acl-model` in [netmod-acl-model].

```

Role-Based ACL
  +--rw role-based-acl
    uses ietf-netmod-acl-model
    
```

Figure 14: Role-Based ACL

In [netmod-acl-model], ietf-netmod-acl-model refers module ietf-netmod-acl-model in [netmod-acl-model]. We add the role-based ACL because it is absent in [i2nsf-capability-dm].

6.2. YANG Modules

This section introduces a YANG module for the information model of the required data for the registration interface between Security Controller and Developer's Management System, as defined in Section 5.

```
<CODE BEGINS> file "ietf-i2nsf-regs-interface@2018-07-26.yang"
  module ietf-i2nsf-regs-interface {
    namespace
      "urn:ietf:params:xml:ns:yang:ietf-i2nsf-regs-interface";
    prefix
      regs-interface;
    import ietf-inet-types{
      prefix inet;
    }

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

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

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

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

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

      Editor: 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 module for Registration Interface.";
revision "2018-07-26"{
  description "The second revision";
  reference
    "draft-ietf-i2nsf-capability-data-model-01";
}
list interface-container{
  key "interface-name";
  description
    "i2nsf-reg-interface-container";
  leaf interface-name{
    type string;
    description
      "interface name";
  }
  container i2nsf-regs-req {
    description
      "The capability information of newly
      created NSF to notify its
      capability to Security Controller";
    container nsf-capability-information {
      description
        "nsf-capability-information";
      uses i2nsf-nsf-capability-information;
    }
    container nsf-access-info {
      description
        "nsf-access-info";
      uses i2nsf-nsf-access-info;
    }
    container ietf-netmod-acl-model{
      description
        "netmod-acl-model";
      uses ietf-netmod-acl-model;
    }
  }
  container i2nsf-instance-mgmt-req {
    description
      "Required information for instantiation-request,
      deinstantiation-request and updating-request";
    leaf req-level {
      type uint16;
      description
        "req-level";
    }
  }
}
```



```
        "processing-average";
    }
    leaf processing-peak{
        type uint16;
        description
            "processing peak";
    }
}
container bandwidth{
    description
        "bandwidth info";
    container inbound{
        description
            "inbound";
        leaf inbound-average{
            type uint16;
            description
                "inbound-average";
        }
        leaf inbound-peak{
            type uint16;
            description
                "inbound-peak";
        }
    }
    container outbound{
        description
            "outbound";
        leaf outbound-average{
            type uint16;
            description
                "outbound-average";
        }
        leaf outbound-peak{
            type uint16;
            description
                "outbound-peak";
        }
    }
}
}
grouping i2nsf-nsf-capability-information {
    description
        "Detail information of an NSF";
    container performance-capability {
        uses i2nsf-nsf-performance-caps;
        description
            "performance-capability";
    }
}
```

```
    }
    container i2nsf-capability {
      description
        "It refers draft-ietf-i2nsf-capability-data-model-01.txt
        later";
    }
  }
  grouping ietf-netmod-acl-model {
    description
      "Detail information";
    container role-based-acl {
      description
        "It refers draft-ietf-netmod-acl-model-19.txt
        later";
    }
  }
  grouping i2nsf-nsf-access-info {
    description
      "NSF access information";
    leaf nsf-address {
      type inet:ipv4-address;
      mandatory true;
      description
        "nsf-address";
    }
    leaf nsf-port-address {
      type inet:port-number;
      description
        "nsf-port-address";
    }
  }
}

<CODE ENDS>
```

Figure 15: Data Model of I2NSF Registration Interface

6.2.1. XML Example of Registration Interface Data Model

Requirement: Registering the IDS NSF with VoIP/VoLTE security capability using Registration interface.

Here is the configuration xml for this Registration Interface:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
```

```
<running/>
</target>
<config>
<i2nsf-regs-req>
  <i2nsf-nsf-capability-information>
    <ietf-i2nsf-capability>
      <nsf-capabilities>
        <nsf-capabilities-id>1</nsf-capabilities-id>
        <con-sec-control-capabilities>
          <content-security-control>
            <ids>
              <ids-support>true</ids-support>
              <ids-fcn nc:operation="create">
                <ids-fcn-name>ids-service</ids-fcn-name>
              </ids-fcn>
            </ids>
            <voip-volte>
              <voip-volte-support>true</voip-volte-support>
              <voip-volte-fcn nc:operation="create">
                <voip-volte-fcn-name>
                  ips-service
                </voip-volte-fcn-name>
              </voip-volte-fcn>
            </voip-volte>
          </content-security-control>
        </con-sec-control-capabilities>
      </nsf-capabilities>
    </ietf-i2nsf-capability>
    <i2nsf-nsf-performance-caps>
      <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>
    </i2nsf-nsf-performance-caps>
  </i2nsf-nsf-capability-information>
  <nsf-access-info>
    <nsf-address>10.0.0.1</nsf-address>
    <nsf-port-address>145</nsf-port-address>
```

```
        </nsf-access-info>
      </i2nsf-regs-req>
    </config>
  </edit-config>
</rpc>
```

Figure 16: Registration Interface example

7. Security Considerations

This document introduces no additional security threats and SHOULD follow the security requirements as stated in [RFC8329].

8. Acknowledgments

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

9. References

9.1. Normative References

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

9.2. Informative References

- [capability-im] Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", draft-i2nsf-capability-02 (work in progress), July 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-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-06 (work in progress), July 2018.

[i2rs-rib-data-model]
Wang, L., Chen, M., Dass, A., Ananthakrishnan, H., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-15 (work in progress), May 2018.

[netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-19 (work in progress), April 2018.

[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.

[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.

Authors' Addresses

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

E-Mail: 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
E-Mail: 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
E-Mail: tkroh0198@skku.edu
URI: http://imtl.skku.ac.kr/xs/index.php?mid=board_YoKq57

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
E-Mail: dn19795@skku.edu
URI: http://imtl.skku.ac.kr/xs/index.php?mid=board_YoKq57

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

Interface to Network Security Functions (I2NSF)
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2019

L. Xia
Q. Lin
Huawei
October 21, 2018

I2NSF Security Policy Object YANG Data Model
draft-xia-i2nsf-sec-object-dm-01

Abstract

This document describes a set of policy objects which are reusable and can be referenced by variable I2NSF policy rules. And the YANG data models of these policy objects are provided.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2019.

Copyright Notice

Copyright (c) 2018 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
4.	Tree Diagrams	3
5.	Policy Object	4
5.1.	Address Object and Address Group	4
5.2.	Service Object and Service Group	5
5.3.	Application Object and Application Group	7
5.4.	User Object, User Group and Security Group	9
5.5.	Time Range Object	11
5.6.	Region Object and Region Group	11
5.7.	Domain Object	12
6.	I2NSF Security Policy Object YANG Module	13
7.	Acknowledgements	46
8.	IANA Considerations	46
9.	Security Considerations	46
10.	References	46
10.1.	Normative References	46
10.2.	Informative References	46
	Authors' Addresses	47

1. Introduction

As described in [RFC8329], provisioning to NSFs can be standardized by using policy rules, and I2NSF uses Event-Condition-Action (ECA) model to represent policy rules. According to [I-D.ietf-i2nsf-terminology], an I2NSF 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 the set of actions in that I2NSF policy rules can be executed or not. Information Model of NSFs Capabilities [I-D.ietf-i2nsf-capability] describes attributes of different condition subclasses. When configuring I2NSF condition clause by attributes or features, it is common to see that the same value of an attribute or the same value set of several attributes are configured for many times. And modifications of the policy rules are also very tedious and time-consuming.

To facilitate the provisioning of NSF instances, this document describes a set of policy objects which are reusable. These policy objects can then be referenced in the condition clause of variable I2NSF policy rules. A policy object consists of a name attribute that identifies itself and one or several attributes that are typically used together to represent a certain condition. For example, protocol type and port number are usually used together to represent a certain service. Each policy object should be predefined

and named in order to be used in I2NSF policy rules. By defining policy objects, the creation and maintenance of policy rules are greatly simplified.

- o A policy object can be referenced in different policy rules as required to provide re-usability. And a policy rule can reference several policy objects.
- o The modification of a policy object will be propagated to the I2NSF policy rules that reference this object. No modification should be made to the related policy rules.

According to [I-D.ietf-i2nsf-terminology], there are two kinds of I2NSF policy rules, I2NSF Directly Consumable Policy Rule and I2NSF Indirectly Consumable Policy Rule. The former one can be executed by a network device without translating its content or structure, while the latter one can not be executed by a network device without first translating its content or structure. In this document, policy objects are defined for I2NSF directly consumable policy rules.

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 terms defined in [I-D.ietf-i2nsf-terminology] and [RFC7950].

4. Tree Diagrams

Tree diagram defined in [RFC8340] is used to represent the policy objects defined in this document. The meaning of the symbols used in the tree diagrams of following sections and the syntax are as follows:

- o Groupings, offset by 2 spaces, and identified by the keyword "grouping" followed by the name of the grouping and a colon (":") character.
- o Each node in the tree is prefaced with "+--". Schema nodes that are children of another node are offset from the parent by 3 spaces.
- o Brackets "[" and "]" enclose list keys.

- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" means state data (read-only), and "-u" indicates the use of a predefined grouping.
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Curly brackets and a question mark "{...}?" are combined to represent the features that node depends on.

5. Policy Object

These document defines policy objects that are commonly used. Figure 1 shows all the defined policy objects and their relationships.

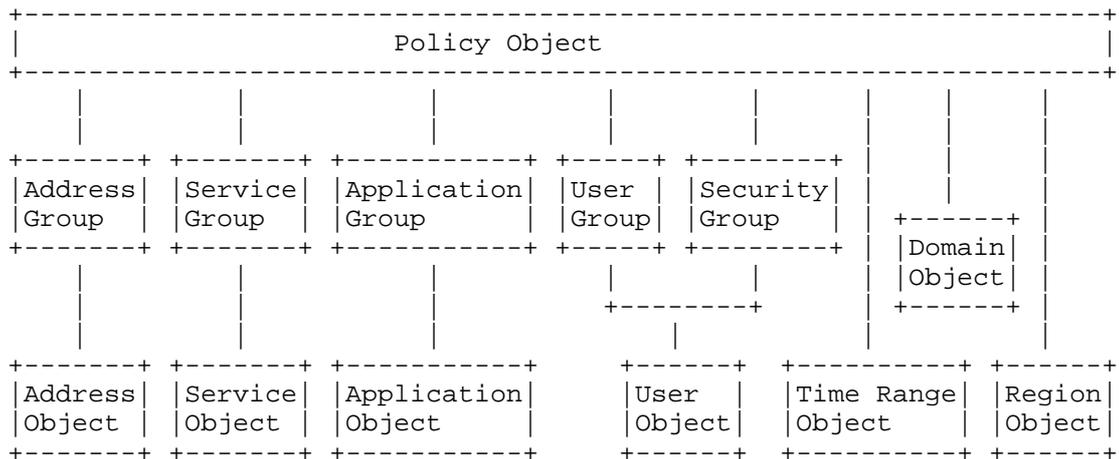


Figure 1: The Policy Objects Overview

5.1. Address Object and Address Group

An address object is identified by a unique name, which contains a set of IPv4/IPv6 addresses or MAC addresses. Several address objects can be organized into an address group object.

This document defines groupings for address objects and address groups.

The tree diagram of address object is:

grouping address-objects:

```

+--rw address-object* [name]
  +--rw name                address-set-name
  +--rw desc?               string
  +--rw vpn-instance?       string
  +--rw elements* [elem-id]
    +--rw elem-id           uint16
    +--rw (object-items)
      +--:(ipv4)
        | +--rw address-ipv4  inet:ipv4-prefix
      +--:(ipv6)
        | +--rw address-ipv6  inet:ipv6-prefix
      +--:(mac)
        | +--rw mac-address   yang:mac-address
        | +--rw mac-address-mask yang:mac-address
      +--:(ipv4-range)
        | +--rw start-ipv4    inet:ipv4-address
        | +--rw end-ipv4      inet:ipv4-address
      +--:(ipv6-range)
        | +--rw start-ipv6    inet:ipv6-address
        | +--rw end-ipv6      inet:ipv6-address

```

The tree diagram of address group is:

grouping address-groups:

```

+--rw address-group* [name]
  +--rw name                address-set-name
  +--rw desc?               string
  +--rw vpn-instance        string
  +--rw elements* [elem-id]
    +--rw elem-id           uint16
    +--rw addr-object-name   address-set-name

```

5.2. Service Object and Service Group

A service object is a kind of service based on IP, or ICMP, or UDP, or TCP, or SCTP. Several related objects consist a service group. To identify different kinds of services, different kinds of attributes should be specified.

- o UDP, TCP, or SCTP based service is recognized by port number. The source port number and destination port number are used to identify the sending and receiving service respectively.
- o ICMP or ICMPv6 based service is recognized by two header fields in the ICMP/ICMPv6 packets: type field and code field.

- o IP based service is recognized by the value of the protocol field in IP packet header.

Besides, a set of well-known services should be predefined by NSFs as service objects to support direct usage.

The tree diagram of service object is:

grouping service-objects:

```

+--ro pre-defined-service* [name]
|   +--ro name                string
|   +--ro session-aging-time  uint16
+--rw service-object* [name]
    +--rw name                 service-set-name
    +--rw session-aging-time   uint16
    +--rw desc?                string
    +--rw items* [id]
        +--rw id                uint16
        +--rw (item)
            +--:(tcp-item)
                |   +--rw tcp
                |   +---u port-items
            +--:(udp-item)
                |   +--rw udp
                |   +---u port-items
            +--:(sctp-item)
                |   +--rw sctp
                |   +---u port-items
            +--:(icmp-item)
                |   +--rw (icmp-type)
                |   +--:(name-type)
                |   |   +--rw icmp-name                icmp-name-type
                |   +--:(type-code)
                |   |   +--rw icmp-type-code
                |   |   +--rw icmp-type-number          uint8
                |   |   +--rw icmp-code-number         string
            +--:(icmp6-item)
                |   +--rw (icmp6)
                |   +--:(name-type)
                |   |   +--rw icmp6-name                icmp6-name-type
                |   +--:(type-code)
                |   |   +--rw icmp6-type-code
                |   |   +--rw icmp-type-number          uint8
                |   |   +--rw icmp-code-number         string
            +--:(protocol-id)
                |   +--rw proto-id                proto-id-range

```

The "port-items" grouping reuses "port-range-or-operator" grouping defined in [I-D.ietf-netmod-acl-model].

```
grouping port-items:
  +--rw source-port
  |   +---u pf:port-range-or-operator
  +--rw dest-port
      +---u pf:port-range-or-operator
```

The tree diagram of service group is:

```
grouping service-groups:
  +--rw service-group* [name]
      +--rw name                service-set-name
      +--rw desc?               string
      +--rw items* [id]
          +--rw id                uint16
          +--rw service-set-name  service-set-name
```

5.3. Application Object and Application Group

Due to the diversity and large amount of applications, it is not able to identify a certain application based on protocol type and port number. For example, there are many web applications with different risk levels run on ports 80 and 443 using HTTP and HTTPS, such as web gaming application and web chat application. Protocol type and port number could not distinguish applications using the same application protocol. In this document, category, subcategory, data transmission model, and risk level are used to describe an application. A set of well-known application objects should be predefined in NSFs to support direct reference.

The tree diagram of application object is:

```

grouping application-objects:
  +--rw user-defined-application {user-defined-application}?
  |   +--rw application* [name]
  |   |   +--rw name                string
  |   |   +--rw label*              string
  |   |   +--rw data-model?         string
  |   |   +--rw category?           string
  |   |   +--rw subcategory?        string
  |   |   +--ro risk-value?         uint32
  |   |   +--rw desc? string
  |   |   +--rw rule* [name]
  |   |   |   +--rw name                string
  |   |   |   +--rw protocol?          protocol
  |   |   |   +--rw signature
  |   |   |   |   +--rw mode?           mode
  |   |   |   |   +--rw direction?     direction
  |   |   |   |   +--rw pattern-type?  pattern-type
  |   |   |   |   +--rw pattern?      string
  |   |   |   |   +--rw field?        identityref
  |   |   +--rw ip-address*         inet:ip-prefix
  |   |   +--rw port*               inet:port-number
  |   |   +--rw desc? string
  +--ro predefined-application
  |   +--ro application* [name]
  |   |   +--ro name                string
  |   |   +--ro protocol*           string
  |   |   +--ro risk-value?         uint32
  |   |   +--ro label*              string
  |   |   +--ro abandon?            boolean
  |   |   +--ro multichannel?       boolean
  |   |   +--ro data-model?         string
  |   |   +--ro category?           string
  |   |   +--ro subcategory?        string
  |   |   +--ro desc?               string

```

The tree diagram of application group is:

```

grouping application-groups:
  +--rw application-group* [name]
  |   +--rw name                string
  |   +--rw desc?               string
  |   +--rw items* [id]
  |   |   +--rw id                uint16
  |   |   +--rw application-object-name string

```

5.4. User Object, User Group and Security Group

A user object identifies a person who may access network resources. It is the basis of implementing user-based policy control. The user objects may be created locally on the NSFs, or be imported from third parties, such as authentication servers. User objects that require the same policy enforcement are grouped as user group objects or security group objects. The user group objects are organized as a hierarchical structure. A security group object consists of user objects from different user group objects that require the same policy enforcement.

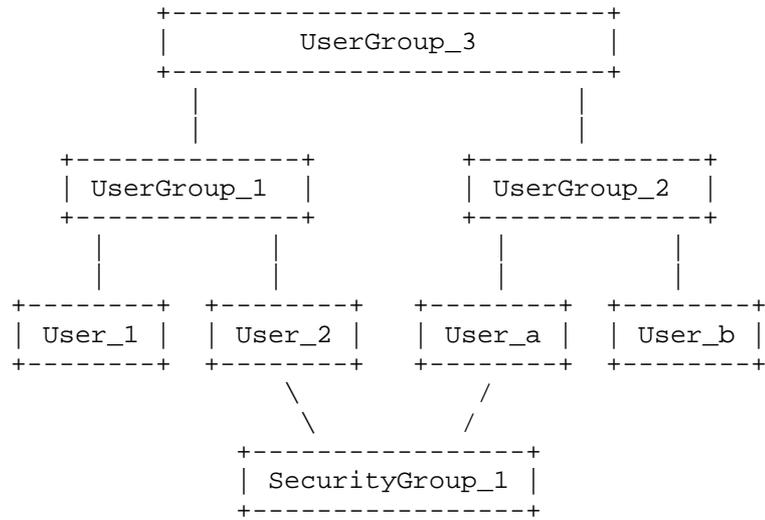


Figure 2: Example of User, User Group and Security Group Structure

The tree diagram of user object is:

```

grouping user-objects:
  +--rw user-object* [name aaa-domain]
    +--rw name          user-name
    +--rw aaa-domain   string
    +--rw desc?        string
    +--rw password?    ianach:crypt-hash
    +--rw parent-user-group      user-group-name
    +--rw parent-security-group  user-security-group-name
    +--rw expiration-time
      +--:(expiration-type)
      |   +--rw (never-expire)
      |   |   +--rw never-expire
      |   +--rw (expire-after-this-time)
      |   |   +--rw expiration-time yang:date-and-time
    +--rw ip-mac-binding
      +--:(bind-state)
      |   +--rw (no-binding)
      |   |   +--rw no-binding
      |   +--rw (binding)
      |   |   +--rw bind-mode          ip-mac-binding-type
      |   |   +--rw ip-binding*       inet:ipv4-address
      |   |   +--rw mac-binding*      yang:mac-address
      |   |   +--rw ip-mac-bindings [ip-binding]
      |   |   |   +--rw ip-binding    inet:ipv4-address
      |   |   |   +--rw mac-binding   yang:mac-address;

```

The tree diagram of user group is:

```

grouping user-groups:
  +--rw user-group* [name]
    +--rw name          user-group-name
    +--rw desc?        string
    +--rw parent-user-group  user-group-name

```

The tree diagram of security group is:

```

grouping security-groups:
  +--rw security-group* [name]
    +--rw name          user-security-group-name
    +--rw desc?        string
    +--rw parent-security-group*? user-security-group-name
    +--rw filter-action
      +--:(filter-type)
      |   +--rw (static)
      |   |   +--rw static
      |   +--rw (dynamic)
      |   |   +--rw dynamic
      |   |   +--rw filter-rule* string

```

5.5. Time Range Object

There are two kinds of time ranges: periodic time range and absolute time range. A periodic time range occurs every week. An absolute time range occurs only once.

The tree diagram of time range object is:

```
grouping time-range-objects:
  +--rw time-range-object* [name]
    +--rw name                time-range-name
    +--rw period-time* [start end]
      | +--rw start          hour-minute-second
      | +--rw end            hour-minute-second
      | +--rw weekday        weekday
    +--rw absolute-time* [start end]
      +--rw start            yang:date-and-time
      +--rw end              yang:date-and-time
```

5.6. Region Object and Region Group

A region object is a set of public IP addresses that are assigned to a certain geographic location. A region group consists of a set of region objects.

The tree diagram of region object is:

```

grouping region-objects:
  +--ro pre-defined-region* [name]
  |   +--ro name          region-name
  |   +--ro desc?        string
  |   +--ro region-ipv4-address
  |   |   +--ro address-ipv4*  inet:ipv4-prefix
  |   |   +--ro address-ipv4-range* [start-ipv4 end-ipv4]
  |   |   |   +--ro start-ipv4  inet:ipv4-address
  |   |   |   +--ro end-ipv4    inet:ipv4-address
  |   +--ro region-ipv6-address {support-ipv6-address}?
  |   |   +--ro address-ipv6*  inet:ipv6-prefix
  |   |   +--ro address-ipv6-range* [start-ipv6 end-ipv6]
  |   |   |   +--ro start-ipv6  inet:ipv6-address
  |   |   |   +--ro end-ipv6    inet:ipv6-address
  +--rw user-defined-region* [name]
  |   +--rw name          region-name
  |   +--rw desc?        string
  |   +--rw coordinate
  |   |   +--rw longitude    region-longitude
  |   |   +--rw latitude    region-latitude
  +--rw region-ipv4-address
  |   +--rw address-ipv4*  inet:ipv4-prefix
  |   +--rw address-ipv4-range* [start-ipv4 end-ipv4]
  |   |   +--rw start-ipv4  inet:ipv4-address
  |   |   +--rw end-ipv4    inet:ipv4-address
  +--rw region-ipv6-address {support-ipv6-address}?
  |   +--rw address-ipv6*  inet:ipv6-prefix
  |   +--rw address-ipv6-range* [start-ipv6 end-ipv6]
  |   |   +--rw start-ipv6  inet:ipv6-address
  |   |   +--rw end-ipv6    inet:ipv6-address

```

The tree diagram of region group is:

```

grouping region-groups:
  +--rw region-group* [name]
  |   +--rw name          region-name
  |   +--rw desc?        string
  |   +--rw region-name*  region-name
  |   +--rw region-group-name*  region-name

```

5.7. Domain Object

The tree diagram of domain object is:

```
grouping domain-objects:
  +--rw domain-object* [name]
    +--rw name          domain-name
    +--rw desc?         string
    +--rw domain*       string
```

6. I2NSF Security Policy Object YANG Module

```
<CODE BEGINS> file "ietf-policy-object@2018-10-12.yang"
module ietf-policy-object {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-policy-object";
  prefix policy-object;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC7317 - A YANG Data Model for System Management.";
  }

  import ietf-packet-fields {
    prefix pf;
    reference
      "draft-ietf-netmod-acl-model - Network Access Control List (ACL) YANG Data
Model.";
  }

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

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

    Editor: Liang Xia
           frank.xialiang@huawei.com
    Editor: Qiushi Lin
```

```
        linqiushi@huawei.com";

description
  "This YANG module defines groupings that are used by ietf-policy-object YANG
  module. Their usage is not limited to ietf-policy-object and can be used anywhe
  re as applicable.";

revision 2018-10-12 {
  description "Initial version.";
  reference
    "draft-xia-i2nsf-sec-object-dm-01";
}

/*
 * Typedefs for address object and address group
 */
typedef address-set-name {
  type string {
    length "1..63";
  }
  description
    "This type represents an address object or an address group name.";
}

/*
 * Typedefs for service object and service group
 */
typedef service-set-name {
  type string {
    length "1..63";
  }
  description
    "This type represents a service object or a service group name.";
}

typedef port-range {
  type uint16;
  description
    "This type represents a port number, which may be a start port of a port r
    ange or an end port of a port range.";
}

typedef proto-id-range {
  type uint8 {
    range "0..255";
  }
  description
    "This type represents the range of protocol id.";
}

typedef icmp-name-type {
```

```
type enumeration {
  enum echo {
    description
    "ICMP type number 8, ICMP code number 0";
  }
  enum echo-reply {
    description
    "ICMP type number 0, ICMP code number 0";
  }
  enum fragmentneed-DFset {
    description
    "ICMP type number 3, ICMP code number 4";
  }
  enum host-redirect {
    description
    "ICMP type number 5, ICMP code number 1";
  }
  enum host-tos-redirect {
    description
    "ICMP type number 5, ICMP code number 3";
  }
  enum host-unreachable {
    description
    "ICMP type number 3, ICMP code number 1";
  }
  enum information-reply {
    description
    "ICMP type number 16, ICMP code number 0";
  }
  enum information-request {
    description
    "ICMP type number 15, ICMP code number 0";
  }
  enum net-redirect {
    description
    "ICMP type number 5, ICMP code number 0";
  }
  enum net-tos-redirect {
    description
    "ICMP type number 5, ICMP code number 2";
  }
  enum net-unreachable {
    description
    "ICMP type number 3, ICMP code number 0";
  }
  enum parameter-problem {
    description
    "ICMP type number 12, ICMP code number 0";
  }
}
```

```
    }
    enum port-unreachable {
        description
        "ICMP type number 3, ICMP code number 3";
    }
    enum protocol-unreachable {
        description
        "ICMP type number 3, ICMP code number 2";
    }
    enum reassembly-timeout {
        description
        "ICMP type number 11, ICMP code number 1";
    }
    enum source-quench {
        description
        "ICMP type number 4, ICMP code number 0";
    }
    enum source-soute-failed {
        description
        "ICMP type number 3, ICMP code number 5";
    }
    enum timestamp-reply {
        description
        "ICMP type number 14, ICMP code number 0";
    }
    enum timestamp-request {
        description
        "ICMP type number 13, ICMP code number 0";
    }
    enum ttl-exceeded {
        description
        "ICMP type number 11, ICMP code number 0";
    }
}
description
    "This type is an enumeration of ICMP type names.";
}

typedef icmp6-name-type {
    type enumeration {
        enum redirect {
            description
            "ICMPv6 type number 137, ICMPv6 code number 0";
        }
        enum echo {
            description
            "ICMPv6 type number 128, ICMPv6 code number 0";
        }
    }
}
```

```
enum echo-reply {
    description
    "ICMPv6 type number 129, ICMPv6 code number 0";
}
enum err-Header-field {
    description
    "ICMPv6 type number 4, ICMPv6 code number 0";
}
enum frag-time-exceeded {
    description
    "ICMPv6 type number 3, ICMPv6 code number 1";
}
enum hop-limit-exceeded {
    description
    "ICMPv6 type number 3, ICMPv6 code number 0";
}
enum host-admin-prohib {
    description
    "ICMPv6 type number 1, ICMPv6 code number 1";
}
enum host-unreachable {
    description
    "ICMPv6 type number 1, ICMPv6 code number 3";
}
enum neighbor-advertisement {
    description
    "ICMPv6 type number 136, ICMPv6 code number 0";
}
enum neighbor-solicitation {
    description
    "ICMPv6 type number 135, ICMPv6 code number 0";
}
enum network-unreachable {
    description
    "ICMPv6 type number 1, ICMPv6 code number 0";
}
enum packet-too-big {
    description
    "ICMPv6 type number 2, ICMPv6 code number 0";
}
enum port-unreachable {
    description
    "ICMPv6 type number 1, ICMPv6 code number 4";
}
enum router-advertisement {
    description
    "ICMPv6 type number 134, ICMPv6 code number 0";
}
```

```
enum router-solicitation {
    description
    "ICMPv6 type number 133, ICMPv6 code number 0";
}
enum unknown-ipv6-opt {
    description
    "ICMPv6 type number 4, ICMPv6 code number 2";
}
enum unknown-next-hdr {
    description
    "ICMPv6 type number 4, ICMPv6 code number 1";
}
}
description
    "This type is an enumeration of ICMPv6 type names.";
}

/*
 * Typedefs for application object and application group
 */
typedef protocol {
    type enumeration {
        enum tcp {
            description
            "tcp protocol";
        }
        enum udp {
            description
            "udp protocol";
        }
        enum any {
            description
            "any protocol";
        }
    }
}
description
    "The protocol of user-defined application rule:tcp/udp/any.";
}

typedef mode {
    type enumeration {
        enum flow {
            description
            "Keyword exists in multiple packets";
        }
        enum packet{
            description
            "Keyword exists in one packet";
        }
    }
}
```

```
    }
  }
  description
    "The mode of keyword identification to identify user-defined applications.
    If the keyword exists in one packet, the mode is Packet. If the keyword exists
    in multiple packets, the mode is Flow.";
}

typedef direction {
  type enumeration {
    enum request {
      description
        "Request indicates that data to the server is monitored to detect appl
        ications.";
    }
    enum response {
      description
        "Response indicates that data from the server is monitored to detect a
        pplications.";
    }
    enum both {
      description
        "Both indicates that data from and to the server is monitored to detec
        t applications.";
    }
  }
  description
    "The data flow direction that is monitored to identify user-defined applic
    ations:request/response/both. Request indicates that data to the server is monit
    ored to detect applications, Response indicates that data from the server is mon
    itored to detect applications, and Both indicates that data from and to the serv
    er is monitored to detect applications.";
}

typedef pattern-type {
  type enumeration {
    enum regular {
      description
        "Regular indicates that the keyword of the match pattern is not a fixe
        d string, it is represented by regular expression.";
    }
    enum plain {
      description
        "Plain indicates that the keyword of the match pattern is a fixed stri
        ng.";
    }
  }
  description
    "The match pattern of the user-defined application rule. If the keyword is
    a fixed string, the pattern type is Plain. If the keyword is not a fixed string
    , the pattern type is Regular Expression.";
}

/*
 * Typedefs for user object, user group, and security group
 */

typedef user-name {
  type string {
    length "1..63";
```



```
    }
    description
      "This type represents a user name.";
  }

  typedef user-group-name {
    type string {
      length "1..63";
    }
    description
      "This type represents a user group name.";
  }

  typedef user-security-group-name {
    type string {
      length "1..63";
    }
    description
      "This type represents a security group name.";
  }

  typedef ip-mac-binding-type {
    type enumeration {
      enum bidirectional {
        description
          "Bidirectional binding indicates that a user must use the specified IP
          and MAC addresses to log in. The same IP and MAC addresses cannot be used by ot
          her bidirectional binding users.";
      }
      enum unidirectional {
        description
          "Unidirectional binding indicates that a user must use the specified I
          P and MAC addresses to log in. The same IP and MAC addresses can also be used by
          other users.";
      }
    }
    description
      "The user and IP/MAC address binding mode: bidirectional, or unidirectiona
      l. In unidirectional binding, a user must use the specified IP and MAC addresses
      to log in. The same IP and MAC addresses can also be used by other users. In b
      idirectional binding, a user must use the specified IP and MAC addresses to log
      in. The same IP and MAC addresses cannot be used by other bidirectional binding
      users.";
  }

  /*
  * Typedefs for time range object
  */
  typedef time-range-name {
    type string {
      length "1..32";
    }
    description
      "This type represents a time-range name.";
  }
}
```

```
typedef hour-minute-second {
  type string {
    pattern '\d{1,2}:\d{1,2}:\d{1,2}';
  }
  description
    "The representation of Hour, Minute, Sencond - hh:mm:ss";
}

typedef weekday {
  type enumeration {
    enum sunday {
      description
        "Sunday of the week";
    }
    enum monday {
      description
        "Monday of the week";
    }
    enum tuesday {
      description
        "Tuesday of the week";
    }
    enum wednesday {
      description
        "Wednesday of the week";
    }
    enum thursday {
      description
        "Thursday of the week";
    }
    enum friday {
      description
        "Friday of the week";
    }
    enum saturday {
      description
        "Saturday of the week";
    }
  }
  description
    "A type modeling the weekdays in the Greco-Roman tradition.";
}

/*
 * Typedefs for region object and region group
 */
typedef region-name {
```

```
    type string;
    description
        "This type represents a location or location set name.";
}

typedef region-longitude {
    type string;
    description
        "This type represents a region longitude number(-180.00 - 180.00).";
}

typedef region-latitude {
    type string;
    description
        "This type represents a region latitude number(-90.00 - 90.00).";
}

typedef domain-name {
    type string {
        length "1..63";
    }
    description
        "This type represents a domain object name.";
}

/*
 * Identities for application object and application group
 */
identity protocol-field {
    description
        "Base type of protocol field.";
}

identity general-payload {
    base protocol-field;
    description
        "The field of signature is general-payload.";
}

identity http-method {
    base protocol-field;
    description
        "The field of signature is http.method.";
}

identity http-uri {
    base protocol-field;
```

```
    description
      "The field of signature is http.uri.";
  }

  identity http-user-agent {
    base protocol-field;
    description
      "The field of signature is http.user-agent.";
  }

  identity http-host {
    base protocol-field;
    description
      "The field of signature is http.host.";
  }

  identity http-content-type {
    base protocol-field;
    description
      "The field of signature is http.content-type.";
  }

  identity http-cookie {
    base protocol-field;
    description
      "The field of signature is http.cookie.";
  }

  identity http-body {
    base protocol-field;
    description
      "The field of signature is http.body.";
  }

  /*
  * Features for application object
  */
  feature user-defined-application {
    description
      "This feature means the NSF supports user-defined application function tha
t can be used to define application rule.";
  }

  /*
  * Features for region object
  */
  feature support-ipv6-address {
    description
```

```
    "This feature means the NSF support configuring IPv6 addresses for Region
Object.";
  }

  /*
  * Groupings for address object and address group
  */
  grouping address-objects {
    list address-object {
      key "name";
      leaf name {
        type address-set-name;
        description
          "The name of the address object.";
      }
      leaf desc {
        type string{
          length "1..127";
        }
        description
          "The description of the address object.";
      }
      leaf vpn-instance {
        type string;
        description
          "The name of the vpn-instrance.";
      }
      list elements {
        key "elem-id";
        leaf elem-id {
          type uint16;
          description
            "The id of the element in address object.";
        }
      }
      choice object-items {
        case ipv4 {
          leaf address-ipv4 {
            type inet:ipv4-prefix;
            description
              "A set of IPv4 addresses that are represented by an IPv4 address
prefix.";
          }
        }
        case ipv6 {
          leaf address-ipv6 {
            type inet:ipv6-prefix;
            description
              "A set of IPv6 addresses that are represented by an IPv6 address
prefix.";
          }
        }
      }
    }
  }
}
```

```

    case mac {
      leaf mac-address {
        type yang:mac-address;
        description
          "MAC address. This leaf is combined with the mac-address-mask leaf to represent a single MAC address or a set of MAC addresses. If the mac-address-mask leaf is not presented, this leaf represents a single MAC address. If the mac-address-mask leaf is setted, this leaf represents a range of contiguous MAC addresses.";
      }
      leaf mac-address-mask {
        type yang:mac-address;
        description
          "If this leaf is not presented, the mac-address leaf represents a single MAC address. If this leaf is setted, the mac-address leaf represents a range of contiguous MAC addresses.";
      }
    }
    case ipv4-range {
      leaf start-ipv4 {
        type inet:ipv4-address;
        description
          "The start IPv4 address of an IPv4 address range.";
      }
      leaf end-ipv4 {
        type inet:ipv4-address;
        description
          "The end IPv4 address of an IPv4 address range.";
      }
    }
    case ipv6-range {
      leaf start-ipv6 {
        type inet:ipv6-address;
        description
          "The start IPv6 address of an IPv6 address range.";
      }
      leaf end-ipv6 {
        type inet:ipv6-address;
        description
          "The end IPv6 address of an IPv6 address range.";
      }
    }
  }
  description
    "Diffrent types of addresses: IPv4, IPv6, MAC.";
}
description
  "A list of addresses that belong to a specific address object.";
}
description
  "A list of address objects.";
}
description
  "This grouping represents a list of address objects. An address object is identified by a unique name and contains a set of IPv4/IPv6 addresses or MAC addresses. This grouping reuse the predefined address-object-item grouping.";
}

```

```

grouping address-groups {
  list address-group {
    key "name";
    leaf name {
      type address-set-name;
      description
        "The name of the address group.";
    }
    leaf desc {
      type string{
        length "1..127";
      }
      description
        "The description of the address group.";
    }
    leaf vpn-instance {
      type string;
      description
        "The name of the vpn-instrance.";
    }
    list elements {
      key "elem-id";
      leaf elem-id {
        type uint16;
        description
          "The id of the element in address group.";
      }
      leaf addr-object-name {
        type address-set-name;
        mandatory true;
        description
          "The name of the address object that consists the address group.";
      }
      description
        "A list of address objects that consists the address group object.";
    }
    description
      "A list of address group objects.";
  }
  description
    "An address group object is comprised of several address objects that require the same policy enforcement. This grouping represents a list of address groups.";
}

/*
 * Groupings for service object and service group
 */
grouping port-items {

```

```
    container source-port {
      uses pf:port-range-or-operator;
      description
        "Source port definition from range or operator.";
    }
    container dest-port {
      uses pf:port-range-or-operator;
      description
        "Destination port definition from range or operator.";
    }
  description
    "This grouping consists of the source port numbers and destination port nu
mbers that represent UDP, TCP or SCTP based services.";
}

grouping service-objects {
  list pre-defined-service {
    key "name";
    config false;
    leaf name {
      type service-set-name;
      config false;
      description
        "The name of the predefined service object.";
    }
    leaf session-aging-time {
      type uint16;
      units second;
      config false;
      description
        "The aging time of the predefined service object.";
    }
  }
  description
    "A list of the predefined service objects.";
}
list service-object {
  key "name";
  leaf name {
    type service-set-name;
    description
      "The name of the service object.";
  }
  leaf session-aging-time {
    type uint16;
    units second;
    description
      "The aging time of the service object.";
  }
}
```

```

leaf desc {
  type string{
    length "1..127";
  }
  description
    "The description of the service object.";
}
list items {
  key "id";
  leaf id {
    type uint16;
    description
      "The id of the element in service object.";
  }
  choice item {
    case tcp-item {
      container tcp {
        uses port-items;
        description
          "TCP based service is recognized by source port number and destination port number. This container reuse the port-items grouping.";
      }
    }
    case udp-item {
      container udp {
        uses port-items;
        description
          "UDP based service is recognized by source port number and destination port number. This container reuse the port-items grouping.";
      }
    }
    case sctp-item {
      container sctp {
        uses port-items;
        description
          "SCTP based service is recognized by source port number and destination port number. This container reuse the port-items grouping.";
      }
    }
    case icmp-item {
      choice icmp-type {
        case name-type {
          leaf icmp-name {
            type icmp-name-type;
            mandatory true;
            description
              "The ICMP based service is identified by the predefined ICMP name type.";
          }
        }
        case type-code {
          container icmp-type-code {

```

```

        leaf icmp-type-number {
            type uint8;
            mandatory true;
            description
                "The ICMP type number.";
        }
        leaf icmp-code-number {
            type string;
            mandatory true;
            description
                "The ICMP code number.";
        }
        description
            "The ICMP based service is recognized by two header fields i
n the ICMP packets: type field and code field.";
    }
    }
    description
        "The ICMP based service object and its attributes.";
}
}
case icmp6-item {
    choice icmp6-type {
        case name-type {
            leaf icmp6-name {
                type icmp6-name-type;
                mandatory true;
                description
                    "The ICMPv6 based service is identified by the predefined IC
MPv6 name type.";
            }
        }
        case type-code {
            container icmp6-type-code {
                leaf icmp6-type-number {
                    type uint8;
                    mandatory true;
                    description
                        "The ICMPv6 type number.";
                }
                leaf icmp6-code-number {
                    type string;
                    mandatory true;
                    description
                        "The ICMP code number.";
                }
            }
            description
                "The ICMPv6 based service is recognized by two header fields
in the ICMPv6 packets: type field and code field.";
        }
    }
}

```

```

        description
        "The ICMPv6 based service object and its attributes.";
    }
    description
    "The ICMPv6 based service object and its attributes.";
}
case protocol-id {
  leaf proto-id {
    type proto-id-range;
    mandatory true;
    description
    "IP based service is identified by the value of the protocol fie
ld in IP packet header.";
  }
}
  description
  "Diffrent types of protocols for service definition.";
}
  description
  "A list of service items that consist an service object.";
}
description
  "A list of user defined service objects.";
}
description
  "A list of the predefined service objects and user defined service objects
.";
}

grouping service-groups {
  list service-group {
    key "name";
    leaf name {
      type service-set-name;
      description
      "The name of the service group.";
    }
    leaf desc {
      type string{
        length "1..127";
      }
      description
      "The description of the service group.";
    }
  }
  list items {
    key "id";
    leaf id {
      type uint16;
      description
      "The id of the element in service group.";
    }
  }
}

```

```

    }
    leaf service-object-name {
        type service-set-name;
        mandatory true;
        description
            "The name of the service object that consists the service group.";
    }
    description
        "A list of service objects that consists the service group object.";
}
description
    "A list of service group objects.";
}
description
    "A service group object is comprised of several service objects that require the same policy enforcement. This grouping represents a list of service groups.";
}

/*
 * Groupings for application object and application group
 */
grouping application-objects {
    container user-defined-application {
        if-feature user-defined-application;
        container applications {
            list application {
                key "name";
                leaf name {
                    type string;
                    description
                        "The name of user-defined application object.";
                }
                leaf-list label {
                    type string;
                    description
                        "A list of labels for user-defined application.";
                }
                leaf data-model {
                    type string;
                    description
                        "The data transmission model of user-defined application. Examples are client/server, peer-to-peer. Data transmission models are predefined in the NSF.";
                }
                leaf category {
                    type string;
                    description
                        "The category of user-defined application. The value of this leaf is selected from a predefined set of categories, e.g., general category, network category.";
                }
                leaf subcategory {

```

```

        type string;
        description
            "The subcategory of user-defined application. ";
    }
    leaf risk-value {
        type uint32;
        config false;
        description
            "The risk value of predefined application.";
    }
    leaf desc {
        type string;
        description
            "The description information of user-defined application.";
    }
    list rule {
        key "name";
        leaf name {
            type string;
            description
                "The name of the user-defined application rule.";
        }
        leaf protocol {
            type protocol;
            description
                "The protocol that user-defined application is based on.";
        }
        container signature {
            leaf mode {
                type string;
                description
                    "The mode of keyword identification. If the keyword exists in
one packet, the mode is Packet. If the keyword exists in multiple packets, the m
ode is Flow.";
            }
            leaf direction {
                type direction;
                description
                    "The traffic direction for application identification. Request
indicates that data to the server is detected, Response indicates that data fro
m the server is detected, and Both indicates that data from and to the server is
detected.";
            }
            leaf pattern-type{
                type pattern-type;
                description
                    "The match pattern of the user-defined application rule. If th
e keyword is a fixed string, the pattern type is Plain. If the keyword is not a
fixed string, the pattern type is Regular Expression.";
            }
            leaf pattern {
                type string;
                description
                    "The keyword of user-defined application rule.";
            }
        }
    }

```

```

        leaf field {
            type identityref {
                base protocol-field;
            }
            default general-payload;
            description
                "The protocol field to search for a signature. The default pro
protocol field is General-payload.";
        }
        description
            "The signature/characteristics of user-defined application.";
    }
    description
        "The rule used to identify the user-defined application.";
}
leaf-list ip-address {
    type inet:ip-prefix;
    description
        "The destination IPv4/IPv6 address of user-defined application.";
}
leaf-list port {
    type inet:port-number;
    description
        "The destination port number of user-defined application.";
}
description
    "A list of user-defined application objects.";
}
    description
        "When the NSF supports user-defined application function, these are a
list of user-defined application objects.";
}
description
    "When the NSF supports user-defined application function, this container
is used to configure application objects.";
}
container predefined-application {
    config false;
    list application {
        key "name";
        leaf name {
            type string;
            config false;
            description
                "The name of the predefined application.";
        }
        leaf-list protocol {
            type string;
            config false;
            description
                "The protocol information of application.";
        }
    }
}

```

```

    }
    leaf risk-value {
        type uint32;
        config false;
        description
            "The risk value of predefined application.";
    }
    leaf-list label {
        type string;
        config false;
        description
            "The label of predefined application,an application may have multiple labels.";
    }
    leaf abandon {
        type boolean;
        config false;
        description
            "The abandon flag of predefined application.";
    }
    leaf multichannel {
        type boolean;
        config false;
        description
            "The multi channel flag of predefined application.";
    }
    leaf data-model {
        type string;
        description
            "The data transmission model of user-defined application. Examples are client/server, peer-to-peer. Data transmission models are predefined in the N SF.";
    }
    leaf category {
        type string;
        config false;
        description
            "The category of user-defined application. The value of this leaf is selected from a predefined set of categories, e.g., general category, network category.";
    }
    leaf subcategory {
        type string;
        config false;
        description
            "The name of application subcategory.";
    }
    leaf desc {
        type string;
        config false;
        description
            "The description information of application.";
    }
}

```

```

        description
            "The attributes of a predefined application.";
    }
    description
        "The information of all predefined applications.";
    }
    description
        "A list of predefined application objects.";
    }

grouping application-groups {
    list application-group {
        key "name";
        leaf name {
            type string;
            description
                "The name of the application group.";
        }
        leaf desc {
            type string{
                length "1..127";
            }
            description
                "The description of the application group.";
        }
        list items {
            key "id";
            leaf id {
                type uint16;
                description
                    "The id of the element in application group.";
            }
            leaf application-object-name {
                type string;
                mandatory true;
                description
                    "The name of the application object that consists the application gr
oup.";
            }
            description
                "A list of application objects that consist an application group objec
t.";
        }
        description
            "A list of application group objects.";
    }
    description
        "An application group object is comprised of several application objects t
hat require the same policy enforcement. This grouping represents a list of appl
ication groups.";
    }
}

```

```

/*
 * Groupings for user object, user group and security group
 */
grouping user-objects {
  list user-object {
    key "name aaa-domain";
    leaf name {
      type user-name;
      description
        "The name of the user.";
    }
    leaf aaa-domain {
      type string {
        length "1..64";
      }
      description
        "The name of the domain to which the user belong.";
    }
    leaf desc {
      type string {
        length "1..127";
      }
      description
        "The description of the user.";
    }
    leaf password {
      type ianach:crypt-hash;
      description
        "If user is authenticated locally on the NSF, this attribute is mandatory. It defines the password corresponding to the user name.";
    }
    leaf parent-user-group {
      type user-group-name;
      description
        "The name of the parent group. User objects and user groups are in a hierarchical structure. A user object can only belong to one user group.";
    }
    leaf-list parent-security-group {
      type user-security-group-name;
      max-elements 40;
      description
        "The name of the parent security group. A user object can belong to several security groups.";
    }
    container expiration-time {
      choice expiration-type {
        case never-expire {
          leaf never-expire {
            type empty;
            description
              "This case indicates that the user never expire.";
          }
        }
      }
    }
  }
}

```

```

    }
  }
  case expire-after-this-time {
    leaf expiration-time {
      type yang:date-and-time;
      description
        "User expired time.";
    }
  }
  description
    "Two types of user expiration configurations.";
}
description
  "User expiration time.";
}
container ip-mac-binding {
  choice bind-state {
    case no-binding {
      leaf no-binding{
        type empty;
        mandatory true;
        description
          "No binding: Indicates that a user is not bound to any IP or MAC
address.";
      }
    }
    case binding {
      leaf bind-mode{
        type ip-mac-binding-type;
        description
          "The user and IP/MAC address binding mode: bidirectional, or uni
directional. In unidirectional binding, a user must use the specified IP and MAC
addresses to log in. The same IP and MAC addresses can also be used by other us
ers. In bidirectional binding, a user must use the specified IP and MAC address
es to log in. The same IP and MAC addresses cannot be used by other bidirectiona
l binding users.";
      }
      leaf-list ip-binding {
        type inet:ipv4-address;
        description
          "The IP address bound to the user.";
      }
      leaf-list mac-binding {
        type yang:mac-address;
        description
          "The MAC address bound to the user.";
      }
    }
  }
  list ip-mac-bindings {
    key "ip-binding";
    unique "mac-binding";
    leaf ip-binding {
      type inet:ipv4-address;
      description
        "The bound IPv4 address";
    }
  }
}

```

```

    }
    leaf mac-binding {
      type yang:mac-address;
      description
        "The bound mac address";
    }
  }
  description
    "Configure the IP address and MAC address pairs bound to the use
r.";
}
}
}
description
  "The binding state: no-binding, binding.";
}
description
  "Whether there are IP/MAC addresses bound to the user.";
}
description
  "User Object and its attributes.";
}
description
  "A list of user objects.";
}

grouping security-groups {
  list security-group {
    key "name";
    leaf name {
      type user-security-group-name;
      description
        "The name of the security-group.";
    }
    leaf desc {
      type string {
        length "1..127";
      }
      description
        "The description of the security-group.";
    }
  }
  leaf-list parent-security-group {
    type user-security-group-name;
    max-elements 40;
    description
      "Configure the name of the parent-security-group.";
  }
  container filter-action {
    choice filter-type {
      case static {
        leaf static {

```

```

        type empty;
        mandatory true;
        description
            "Empty leaf indicates that this is a static security group.";
    }
}
case dynamic {
    leaf dynamic {
        type empty;
        mandatory true;
        description
            "Empty leaf indicates that this is a dynamic security group.";
    }
    leaf-list filter-rule {
        type string {
            length "1..256";
        }
        max-elements 5;
        description
            "Filter rules for dynamic security group.";
    }
}
description
    "The filter type: static, dynamic.";
}
description
    "The filter type of the security group, static and dynamic. For dynamic
c security group, an filter rule needs to be configured.";
}
description
    "Security group and its attributes.";
}
description
    "A list of security groups.";
}

grouping user-groups {
    list user-group {
        key "name";
        leaf name {
            type user-group-name;
            description
                "The name of the user group.";
        }
        leaf desc {
            type string {
                length "1..63";
            }
            description

```

```
        "The description of the user group.";
    }
    leaf parent-user-group {
        type user-group-name;
        description
            "The name of the user group. A user group can only belong to one paren
t user group.";
    }
    description
        "User group and its attributes.";
}
description
    "A list of user groups";
}

/*
 * Groupings for time range object
 */
grouping time-range-objects {
    list time-range-object {
        key "name";
        leaf name {
            type time-range-name;
            description
                "The name of the time range object.";
        }
        list period-time {
            key "start end";
            leaf start {
                type hour-minute-second;
                mandatory true;
                description
                    "Start time of the periodic time range.";
            }
            leaf end {
                type hour-minute-second;
                mandatory true;
                description
                    "End time of the periodic time range.";
            }
        }
        leaf-list weekday {
            type weekday;
            min-elements 1;
            max-elements 7;
            description
                "The weekday to which the periodic time range belongs.";
        }
    }
    description
```

```
        "Periodic time that the associated function starts going into effect."
;
    }
    list absolute-time {
        key "start end";
        leaf start {
            type yang:date-and-time;
            description
                "Absolute start time and date";
        }
        leaf end {
            type yang:date-and-time;
            description
                "Absolute end time and date";
        }
        description
            "Absolute time and date that the associated function starts going into
effect.";
    }
    description
        "The time range object and its attributes.";
}
description
    "A list of time range objects";
}

/*
 * Groupings for region object and region group
 */
grouping region-objects {
    list pre-defined-region {
        key "name";
        config false;
        leaf name {
            type region-name;
            config false;
            description
                "The name of the predefined region.";
        }
        leaf desc {
            type string;
            config false;
            description
                "The description of the predefined region.";
        }
    }
    container region-ipv4-address {
        leaf-list address-ipv4 {
            type inet:ipv4-prefix;
            config false;
        }
    }
}
```

```
        description
            "IPv4 address.";
    }
    list address-ipv4-range {
        key "start-ipv4 end-ipv4";
        leaf start-ipv4 {
            type inet:ipv4-address;
            config false;
            description
                "Start ipv4 address.";
        }
        leaf end-ipv4 {
            type inet:ipv4-address;
            config false;
            description
                "End ipv4 address.";
        }
        description
            "A list of ipv4 address ranges";
    }
    description
        "The IPv4 addresses of the predefined region.";
}
container region-ipv6-address {
    if-feature support-ipv6-address;
    leaf-list address-ipv6 {
        type inet:ipv6-prefix;
        config false;
        description
            "IPv6 address.";
    }
    list address-ipv6-range {
        key "start-ipv6 end-ipv6";
        leaf start-ipv6 {
            type inet:ipv6-address;
            config false;
            description
                "Start ipv6 address.";
        }
        leaf end-ipv6 {
            type inet:ipv6-address;
            config false;
            description
                "End ipv6 address.";
        }
        description
            "A list of ipv6 address ranges";
    }
}
```

```
        description
            "The IPv6 addresses of the predefined region.";
    }
    description
        "A list of predefined region objects.";
}
list user-defined-region {
    key "name";
    leaf name {
        type region-name;
        description
            "The name of the user-defined region.";
    }
    leaf desc {
        type string;
        description
            "The description of the user-defined region.";
    }
}
container coordinate {
    leaf longitude {
        type region-longitude;
        description
            "The latitude of the user-defined region.";
    }
    leaf latitude {
        type region-latitude;
        description
            "The longitude of the user-defined region.";
    }
}
description
    "The latitude and longitude of the user-defined region.";
}
container region-ipv4-address {
    leaf-list address-ipv4 {
        type inet:ipv4-prefix;
        description
            "IPv4 address.";
    }
}
list address-ipv4-range {
    key "start-ipv4 end-ipv4";
    leaf start-ipv4 {
        type inet:ipv4-address;
        description
            "Start ipv4 address.";
    }
    leaf end-ipv4 {
        type inet:ipv4-address;
        description
```

```
        "End ipv4 address.";
    }
    description
        "A list of ipv4 address ranges";
    }
    description
        "The IPv4 addresses of the predefined region.";
    }
    container region-ipv6-address {
        if-feature support-ipv6-address;
        leaf-list address-ipv6 {
            type inet:ipv6-prefix;
            description
                "IPv6 address.";
        }
        list address-ipv6-range {
            key "start-ipv6 end-ipv6";
            leaf start-ipv6 {
                type inet:ipv6-address;
                description
                    "Start ipv6 address.";
            }
            leaf end-ipv6 {
                type inet:ipv6-address;
                description
                    "End ipv6 address.";
            }
        }
        description
            "A list of ipv6 address ranges";
    }
    description
        "The IPv6 addresses of the user-defined region.";
    }
    description
        "A list of user-defined region objects.";
    }
    description
        "A list of predefined region objects and a list of user-defined region objects.";
    }

    grouping region-groups {
        list region-group {
            key "name";
            leaf name {
                type region-name;
                description
                    "The name of the region group.";
            }
        }
    }
}
```

```
    leaf desc {
      type string;
      description
        "The description of the region group.";
    }
    leaf-list region-name {
      type region-name;
      description
        "A list of region objects.";
    }
    leaf-list region-group-name {
      type region-name;
      description
        "A list of region groups.";
    }
  }
  description
    "Region group consists of a set of region objects or region groups.";
}
description
  "A list of region group objects.";
}

/*
 * Groupings for domain object
 */
grouping domain-objects {
  list domain-object {
    key "name";
    leaf name {
      type domain-name;
      description
        "The name of the domain object.";
    }
    leaf desc {
      type string;
      description
        "The description of the domain object.";
    }
  }
  leaf-list domain {
    type string;
    description
      "A list of domains that consists the domain objects.";
  }
  description
    "Domain object and its attributes.";
}
description
  "A list of domain objects.";
```

```
}  
}
```

7. Acknowledgements

8. IANA Considerations

This document requires no IANA actions.

9. Security Considerations

Secure transport should be used to retrieve the current status of management plane security baseline.

10. References

10.1. Normative 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-02 (work in progress), July 2018.

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H.
Birkholz, "Interface to Network Security Functions (I2NSF)
Terminology", draft-ietf-i2nsf-terminology-06 (work in
progress), July 2018.

[I-D.ietf-netmod-acl-model]

Jethanandani, M., Agarwal, S., Huang, L., and D. Blair,
"Network Access Control List (ACL) YANG Data Model",
draft-ietf-netmod-acl-model-20 (work in progress), October
2018.

[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>>.

10.2. Informative 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>>.

[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>>.

[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>>.

Authors' Addresses

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

Qiushi Lin
Huawei
Huawei Industrial Base
Shenzhen, Guangdong 518129
China

Email: linqiushi@huawei.com

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

J. Yang
J. Jeong
J. Kim
Sungkyunkwan University
October 22, 2018

Security Policy Translation in Interface to Network Security Functions
draft-yang-i2nsf-security-policy-translation-02

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 April 25, 2019.

Copyright Notice

Copyright (c) 2018 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. Data Conversion in Data Converter	10
4.3.3. Policy Provisioning	11
4.4. CFG-based Policy Generator	12
4.4.1. Content Production	12
4.4.2. Structure Production	13
4.4.3. Generator Construction	13
5. Features of Policy Translator Design	17
6. Security Considerations	17
7. Acknowledgments	18
8. References	18
8.1. Normative References	18
8.2. Informative References	18
Appendix A. Changes from draft-yang-i2nsf-security-policy-translation-01	20
Authors' Addresses	20

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]. 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].

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] 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

Common security policies are created by Extensible Markup Language (XML) [XML] files. A popular way to change the format of an XML file is to use an Extensible Stylesheet Language Transformation (XSLT) [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 [Automata].

4.1. Overall Structure of Policy Translator

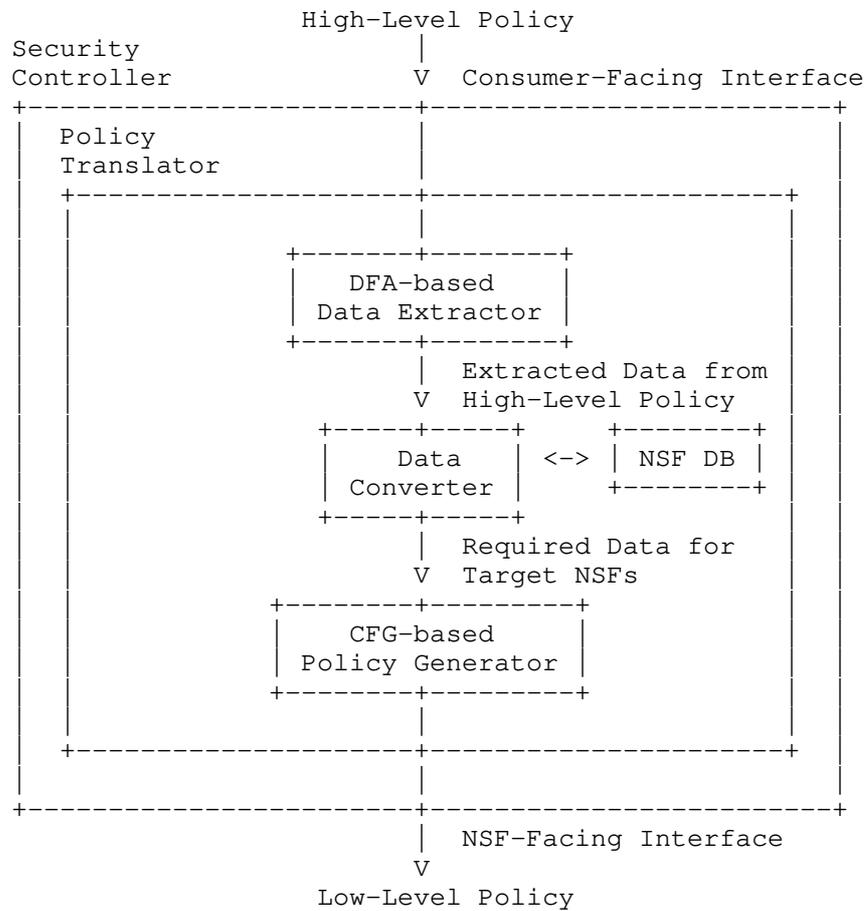


Figure 1: The Overall Design of Policy Translator

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 into the corresponding concrete 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

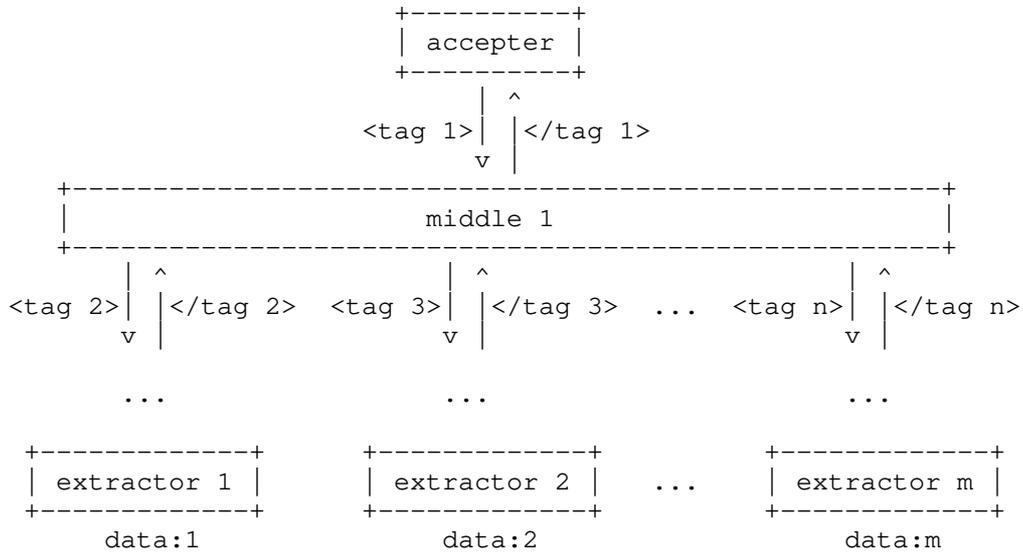


Figure 2: DFA Architecture of Data Extractor

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], 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</dest>
  </cond>
  <action>block</action>
</I2NSF>
    
```

Figure 3: The Example of High-level Policy

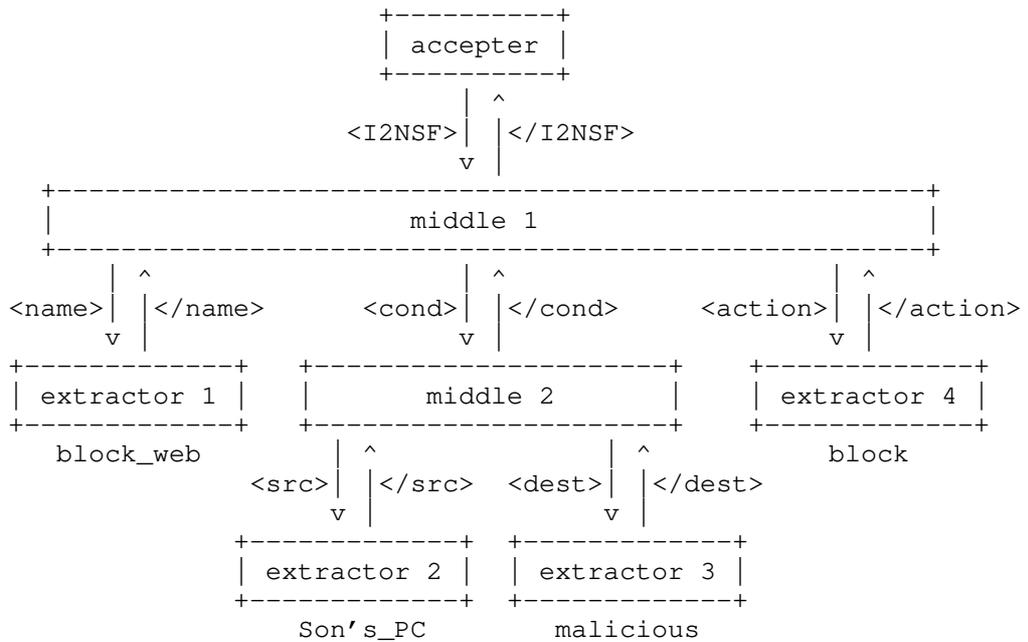


Figure 4: The Example of Data Extractor

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'.
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 the I2NSF User automatically. Thus, the translator design provides big benefits to the I2NSF Framework.

4.3.2. Data Conversion in Data Converter

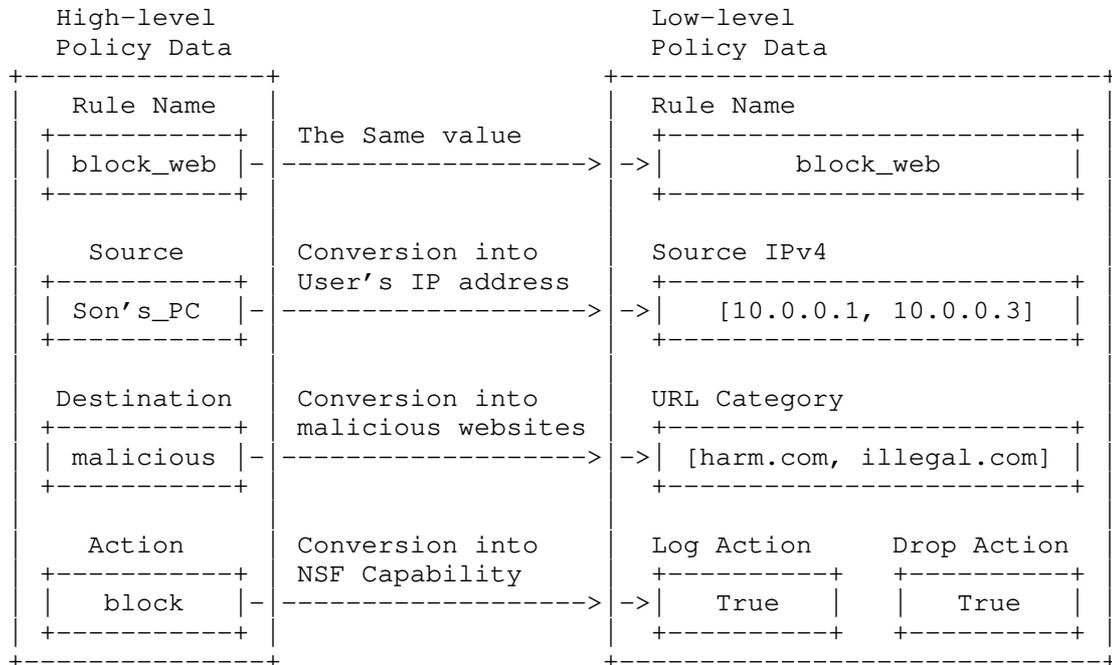


Figure 5: Example of Data Conversion

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.

4.3.3. Policy Provisioning

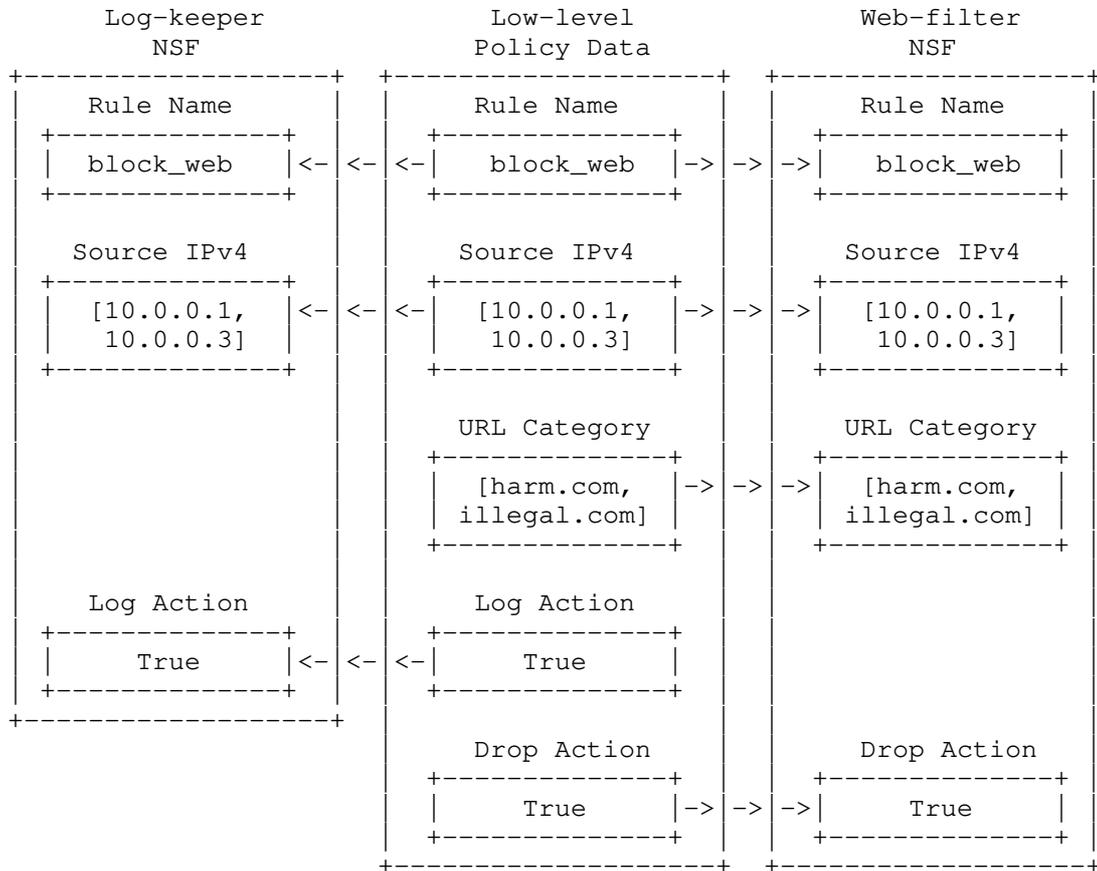


Figure 6: Example of Policy Provisioning

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 a Context-Free Grammar called 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]. 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_dat

- ```
a]</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><payload>[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.

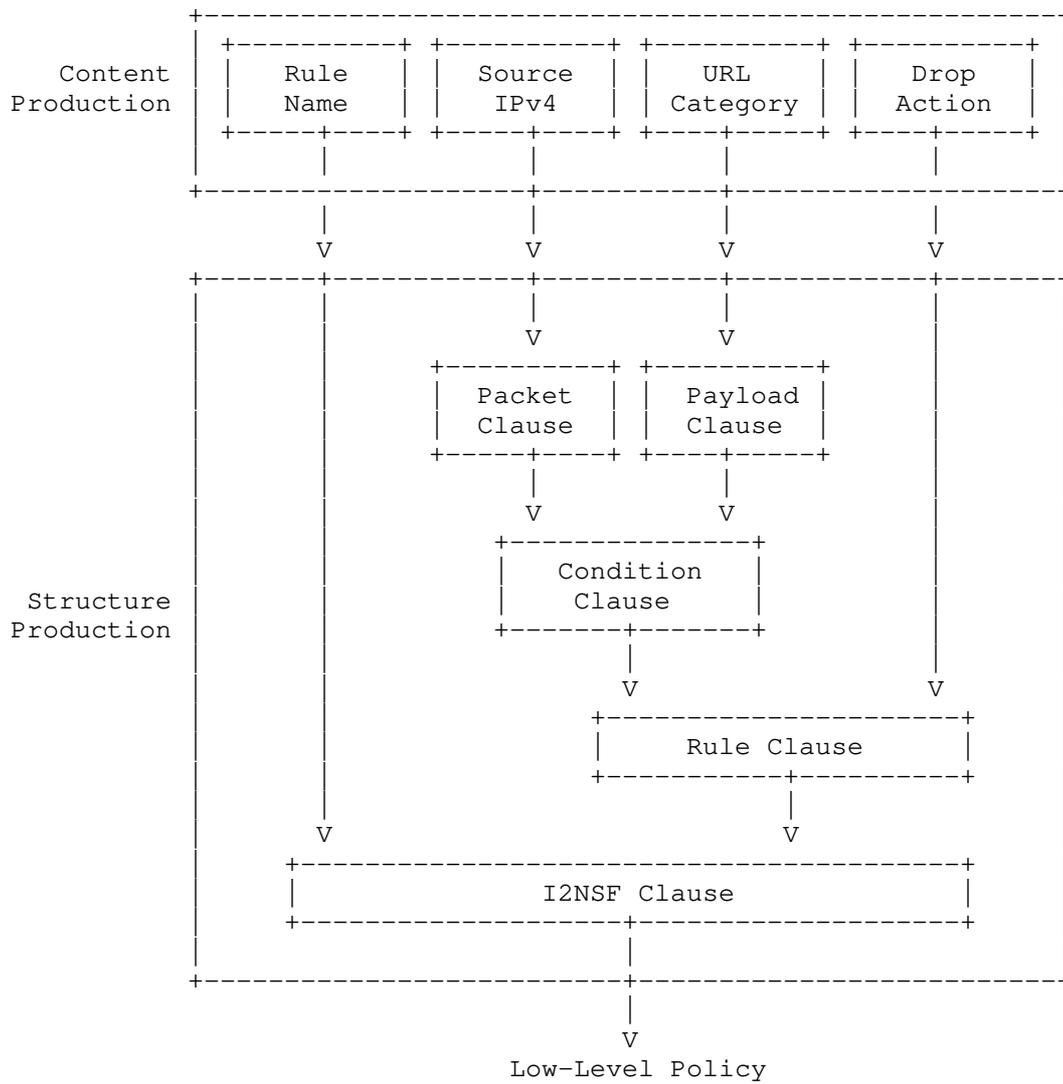


Figure 7: Generator Construction for Web-Filter NSF

```
<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. 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.

## 6. Security Considerations

There is no security concern in a security policy translator proposed in this document as long as the I2NSF interfaces (i.e., Consumer-Facing Interface, NSF-Facing Interface, and Registration Interface) are protected by secure communication channels.

## 7. 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.

## 8. References

### 8.1. 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.

### 8.2. Informative References

[consumer-facing-inf]

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-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-06 (work in progress), July 2018.

## [nsf-facing-inf]

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-data-model-01 (work in progress), July 2018.

## [registration-inf]

Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-ietf-i2nsf-registration-dm-00 (work in progress), October 2018.

## [XSLT]

W3C, "Extensible Stylesheet Language Transformations (XSLT) Version 1.0", November 1999.

Appendix A. Changes from draft-yang-i2nsf-security-policy-translation-01

The following changes are made from draft-yang-i2nsf-security-policy-translation-01:

- o In Section 3, an example is added to emphasize the necessity of a security policy translator. Also, the citation for Automata theory is added.
- o In Section 3 and Section 4, some grammatical errors are corrected.
- o In Figure 1, NSF DB component is added.
- o In Section 4.2, an extraction scenario is added for clearer explanation.
- o In Section 4.3, an example of data conversion is added for clearer explanation. Also, the detailed description and the schematic diagram of policy provisioning is added.
- o In Figure 5, the expression for each conversion is changed.
- o In Section 4.4, an example of CFG is added for clearer explanation. Also, the detailed description and the schematic diagram of CFG-based Generator is added.
- o Section 6 is added for describing "Security Considerations".
- o The References section is divided into two subsections, such as, Normative References and Informative References. Also, the references for Automata theory and XML (Extensible Markup Language) are added to Normative References. In addition, the reference of XSLT (Extensible Stylesheet Language Transformations) is added to Informative References.

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