IETF-115

I2NSF Consumer-Facing Interface and Registration Interface

draft-ietf-i2nsf-consumer-facing-interface-dm-24
draft-ietf-i2nsf-registration-interface-dm-22

November 8, 2022

Patrick Lingga [Presenter] and Jaehoon (Paul) Jeong [Editor]

Sungkyunkwan University

Update of Consumer-Facing Interface (CFI) & Registration Interface (RI)

- **Update**
  - CFI YANG Data Model and RI YANG Data Model are revised from the reviews of AD Roman Danyliw.

- **Revision of CFI YANG Data Model**
  - draft-ietf-i2nsf-consumer-facing-interface-dm-24

- **Revision of RI YANG Data Model**
  - draft-ietf-i2nsf-registration-interface-dm-22

I2NSF Consumer-Facing Interface

draft-ietf-i2nsf-consumer-facing-interface-dm-24

Major Updates of Consumer-Facing Interface (1/6)

OLD:

```
module: ietf-i2nsf-cons-facing-interface
  +--rw i2nsf-cfi-policy* [name]
     +--rw name                      string
     +--rw language?                 string
     +--rw resolution-strategy?      identityref
     +--rw rules* [name]
     |  ...
     +--rw endpoint-groups
     |  ...
     +--rw threat-prevention
        ...
```

Figure 2: Policy YANG Data Tree

NEW:

```
module: ietf-i2nsf-cons-facing-interface
  +--rw i2nsf-cfi-policy* [name]
  |  +--rw name                      string
  |  +--rw language?                 string
  |  +--rw resolution-strategy?      identityref
  |  +--rw rules* [name]
  |     ...
  +--rw endpoint-groups
  |  ...
  +--rw threat-prevention
     ...
```

Figure 2: Policy YANG Data Tree

The “endpoint-groups” and “threat-prevention” are excluded from the policy configuration

Major Updates of Consumer-Facing Interface (2/6)

Addition of Voice-Group for labeling SIP identities.

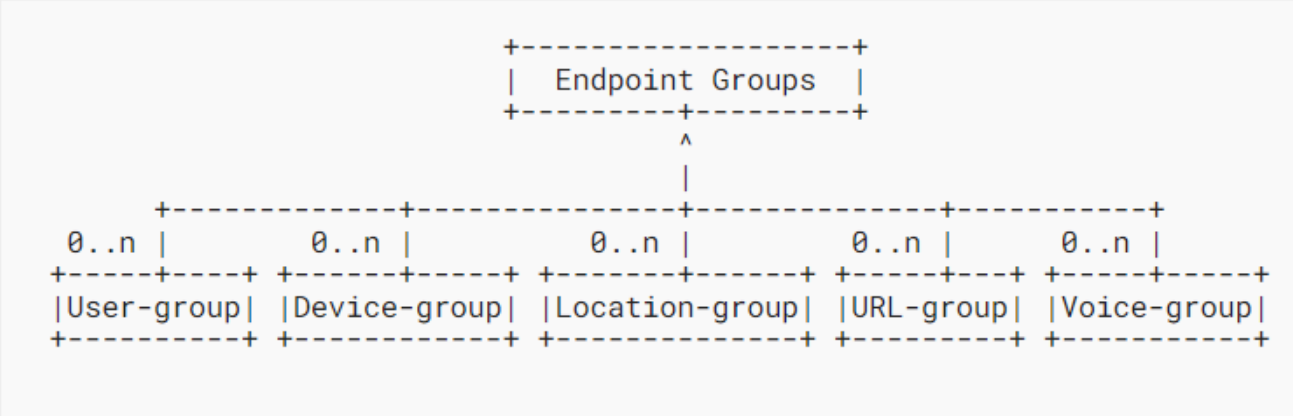

Figure 7: Endpoint Group Diagram

4.5. Voice-Group

The Voice-Group object represents the collection of Session Initiation Protocol (SIP) identities labeled into a group. Figure 13 shows the YANG tree of the Voice-Group object. The Voice-Group object SHALL have the following information:

Name: This field identifies the name of this object.

SIP-id: This field represents the SIP identities in SIP URI scheme (Section 19.1.1 of [RFC3261]).

```
+--rw voice-group* [name]
   +--rw name      string
   +--rw sip-id*   inet:uri
```

Figure 13: Voice-Group YANG Data Tree

Condition model for matching voice:

```
|  +--rw voice
|  |  +--rw source-id*        -> /endpoint-groups/voice-group/name
|  |  +--rw destination-id*   -> /endpoint-groups/voice-group/name
|  |  +--rw user-agent*       string
```

XML example of voice-group registration:

```
<?xml version="1.0" encoding="UTF-8" ?>
<endpoint-groups
 xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cons
        -facing-interface">
  <voice-group>
    <name>malicious-id</name>
    <sip-id>sip:alice@atlanta.com</sip-id>
    <sip-id>sip:bob@203.0.113.15</sip-id>
    <sip-id>sip:carol@chicago.com</sip-id>
  </voice-group>
</endpoint-groups>
```

Major Updates of Consumer-Facing Interface (3/6)

```
+--rw location-group* [country region city]
|  +--rw country                        string
|  +--rw region                         string
|  +--rw city                           string
|  +--rw (match-type)
|     +--:(range-match-ipv4)
|     |  +--rw range-ipv4-address* [start end]
|     |     +--rw start    inet:ipv4-address-no-zone
|     |     +--rw end      inet:ipv4-address-no-zone
|     +--:(range-match-ipv6)
|        +--rw range-ipv6-address* [start end]
|           +--rw start    inet:ipv6-address-no-zone
|           +--rw end      inet:ipv6-address-no-zone
```

Figure 11: Location-Group YANG Data Tree

Updates of Location-Group:

IPv4/IPv6 addresses can be labeled with **country, region, and city** based on RFC 8805 (A Format for Self-Published IP Geolocation Feeds).

Condition model for matching geographic location:

```
|  |  +--rw geographic-location
|  |     +--rw source
|  |     |  +--rw country?   -> /endpoint-groups/location-group/country
|  |     |  +--rw region?    -> /endpoint-groups/location-group/region
|  |     |  +--rw city?      -> /endpoint-groups/location-group/city
|  |     +--rw destination
|  |        +--rw country?   -> /endpoint-groups/location-group/country
|  |        +--rw region?    -> /endpoint-groups/location-group/region
|  |        +--rw city?      -> /endpoint-groups/location-group/city
```

Major Updates of Consumer-Facing Interface (4/6)

Updates of Threat Feed

OLD:

```
+--rw threat-prevention
   +--rw threat-feed-list* [name]
      +--rw name                string
      +--rw description?        string
      +--rw signatures*         identityref
```

NEW:

5.1. Threat Feed

This object represents a threat feed which provides the signatures of malicious activities. Figure 16 shows the YANG tree of a Threat-feed-list. The Threat-Feed object SHALL have the following information:

Name: This field identifies the name of this object.

IOC: This field represents the Indicators of Compromise (IOC), i.e., the critical information of patterns or characteristics in the threat feed that identifies malicious activities. The format of the information given in this field is based on the format field (e.g., STIX, MISP, and OpenIOC).

Format: This field represents the format or structure of the IOC field for the threat-feed such as Structured Threat Information Expression (STIX) [STIX], MISP Core [MISPCORE], and OpenIOC [OPENIOC]. This can be extended depending on the implementation of the existing threat-feed.

It is assumed that the I2NSF User obtains the threat signatures (i.e., threat content patterns) from a threat-feed server (i.e., feed provider), which is a server providing threat signatures. With the obtained threat signatures, the I2NSF User can deliver them to the Security Controller via the Consumer-Facing Interface. The retrieval of the threat signatures by the I2NSF User is out of the scope of this document.

```
      +--rw threat-feed-list* [name]
      |  +--rw name       string
      |  +--rw ioc*       string
      |  +--rw format     identityref
```

Figure 16: Threat Feed YANG Data Tree

Major Updates of Consumer-Facing Interface (5/6)

5.2. Payload Content

This object represents a list of raw binary patterns of a packet payload content (i.e., data after transport layer) to describe a threat. Figure 17 shows the YANG tree of a Payload-content list. The Payload-content object SHALL have the following information:

Name: This field identifies the name of this object. It is recommended to use short and simple words that describe the content. For example, the name "backdoor" indicates the payload content is related to a backdoor attack.

Description: This represents the description to further describe the content field in detail. This field is not mandatory but recommended to be used as it is helpful for future usage.

Content: This represents the payload content patterns(i.e., data after transport layer), which are involved in a security attack, in binary. If multiple instances of content are defined, it should match all contents somewhere in the session stream. The content pattern should be matched based on the order given by the user. The scope of the payload to be matched can be defined by the depth and offset/distance fields.

Depth: This field specifies how far into a packet should be search for the specified content pattern defined in the content field. If this field is undefined, then the content pattern should be searched within the whole payload.

Starting-point: This field specifies the starting point of matching the content pattern to the payload. If this field is undefined, then the content pattern should be searched from the beginning of the payload. The starting point can be defined by either the offset value or distance value. The offset keyword specifies where to start searching for the specified content pattern. The offset is calculated from the beginning of the payload. The distance keyword specifies how far into a payload should be ignored before starting to search for the specified content pattern relative to the end of the previous specified content pattern match. This can be thought of as exactly the same thing as offset, except it is relative to the end of the last pattern match instead of the beginning of the packet. Note that this field cannot be used if the content is the first order of the list.

Updates of Payload Content

```
+--rw payload-content* [name]
   +--rw name              string
   +--rw description?      string
   +--rw contents* [content]
      +--rw content                binary
      +--rw depth?                 uint16
      +--rw (starting-point)?
         +--:(offset)
         |  +--rw offset?          int32
         +--:(distance)
            +--rw distance?       int32
```

Figure 17: Payload Content in YANG Data Tree

Major Updates of Consumer-Facing Interface (6/6)

This section is removed as it is not really involved to I2NSF guidance, but just general guidance of NACM.

6. Network Configuration Access Control Model (NACM) for I2NSF Consumer-Facing Interface

Network Configuration Access Control Model (NACM) provides a user group with an access control with the following features [RFC8341]:¶

- Independent control of action, data, and notification access is provided.
- A simple and familiar set of datastore permissions is used.
- Support for YANG security tagging allows default security modes to automatically exclude sensitive data.
- Separate default access modes for read, write, and execute permissions are provided.
- Access control rules are applied to configurable groups of users.

The data model of the I2NSF Consumer-Facing Interface utilizes the NACM's mechanisms to manage the access control on the I2NSF Consumer-Facing Interface. The NACM with the above features can be used to set up the access control rules of a user group in the I2NSF Consumer-Facing Interface.

Figure 17 shows part of the NACM module to enable the access control of a user group for the I2NSF Consumer-Facing Interface. To use the NACM, a user needs to configure either a NETCONF server [RFC6241] or a RESTCONF server [RFC8040] to enable the NACM module. Then, the user can simply use an account of root or admin user for the access control for the module of the I2NSF Consumer-Facing Interface (i.e., ietf-i2nsf-cons-facing-interface). An XML example to configure the access control of a user group for the I2NSF Consumer-Facing Interface can be seen in Section 9.

Minor Updates of Consumer-Facing Interface

Addition of a way to configure rate-limit value in bytes per second if the action is "rate-limit".

```
container primary-action {
  description
    "This represents primary actions (e.g., ingress and
     egress actions) to be applied to a condition.
     If this is not set, it cannot support the primary
     actions.";
  leaf action {
    type identityref {
      base primary-action;
    }
    mandatory true;
    description
      "Ingress actions: pass, drop, reject, rate-limit,
       and mirror.
       Egress actions: pass, drop, reject, rate-limit,
       mirror, invoke-signaling, tunnel-encapsulation,
       forwarding, and transformation.";
  }
  leaf limit {
    when "../action = 'i2nsfcfi:rate-limit'" {
      description
        "Rate-limit is valid only when rate-limit action is
         used.";
    }
    type decimal64 {
      fraction-digits 2;
    }
    units "bytes per second";
    description
      "Specifies how to rate-limit the traffic.";
  }
}
```

Clarification of the definition of User-Group, Device-Group, URL-Group, and Location-Group.

4.1. User-Group

The User-Group object represents the MAC addresses and IP (IPv4 or IPv6) addresses that are labeled as a group of users (e.g., employees). Figure 9 shows the YANG tree of the User-Group object. The User-Group object SHALL have the following information:

4.2. Device-Group

The Device-Group object represents the labeled network devices that provide services (e.g., servers) hosted on the IP addresses and application protocol. Figure 10 shows the YANG tree of the Device-group object. The Device-Group object SHALL have the following information:

4.3. Location-Group

The Location-Group object represents the IP (IPv4 or IPv6) addresses labeled as a geographic location (i.e., country, region, and city) [RFC8805]. Figure 11 shows the YANG tree of the Location-Group object. The Location-Group object SHALL have the following information:

4.4. URL-Group

The URL-Group object represents the collection of Uniform Resource Locators (URLs) or hostnames labeled into a group (e.g., sns-websites). Figure 12 shows the YANG tree of the URL-Group object. The URL-Group object SHALL have the following information:

I2NSF Registration Interface

draft-ietf-i2nsf-registration-interface-dm-22

Major Updates of Registration Interface (1/4)

OLD:

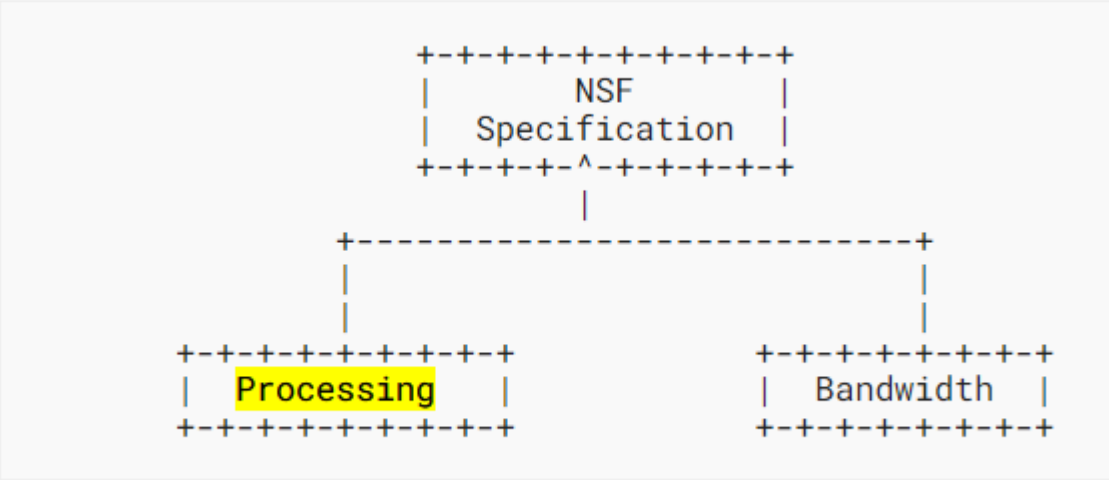

Figure 4: NSF Specification Overview

NEW:

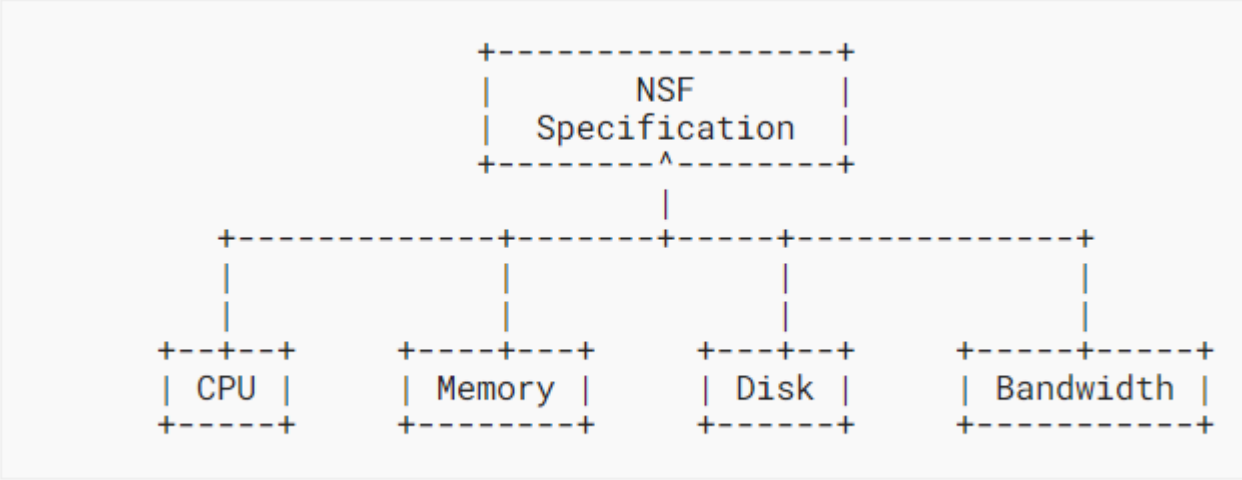

Figure 4: NSF Specification Overview

Clarification of the specification information and addition of memory and disk specification for NSFs

```
NSF Capability Registration
augment /i2nsfcap:nsf:
  +--rw nsf-specification
  |  +--rw cpu
  |  |  +--rw model?              string
  |  |  +--rw clock-speed?        uint16
  |  |  +--rw cores?              uint8
  |  |  +--rw threads?            uint16
  |  +--rw memory
  |  |  +--rw capacity?        uint32
  |  |  +--rw speed?           uint32
  |  +--rw disk
  |  |  +--rw capacity?        uint32
  |  +--rw bandwidth
  |     +--rw outbound?        uint64
  |     +--rw inbound?         uint64
  +--rw nsf-access-info
     +--rw ip?                          union
     +--rw port?                        inet:port-number
     +--rw management-protocol?         enumeration
     +--rw name?                        string
     +--rw password?                    ianach:crypt-hash
```

Figure 5: YANG Tree of NSF Capability Registration Module

Major Updates of Registration Interface (2/4)

The CPU information includes model, clock-speed, cores, and threads.

```
container cpu {
  description
    "The Central Processing Unit (CPU) specification of the NSF";

  leaf model {
    type string;
    description
      "The model name of the CPU used in the NSF.";
  }
  leaf clock-speed {
    type uint16;
    units "MHz";
    description
      "The number of cycles the CPU executes per second,
       measured in MHz (MegaHertz).";
  }
  leaf cores {
    type uint8;
    description
      "The number of independent CPU in a single computing
       component.";
  }
  leaf threads {
    type uint16;
    description
      "The number of total threads of the CPU";
  }
}
```

The memory information includes capacity and speed. The disk information includes the capacity.

```
container memory {
  description
    "Memory (i.e., Random Access Memory (RAM)) specification of
     an NSF.";

  leaf capacity {
    type uint32;
    units "MB";
    description
      "The total memory capacity in Megabytes (MB).";
  }
  leaf speed {
    type uint32;
    units "MHz";
    description
      "The data transfer rate of the memory in MegaHertz (MHz).";
  }
}

container disk {
  description
    "Disk or storage specification of an NSF";

  leaf capacity {
    type uint32;
    units "MB";
    description
      "The disk or storage maximum capacity in Megabytes (MB).";
  }
}
```

Major Updates of Registration Interface (3/4)

The bandwidth information shows the available network bandwidth for the inbound and outbound traffics.

```
container bandwidth {
  description
    "Network bandwidth available on an NSF
     in the unit of Bps (Bytes per second)";

  leaf outbound {
    type uint64;
    units "Bps";
    description
      "The maximum outbound network bandwidth available to the
       NSF in bytes per second (Bps)";
  }

  leaf inbound {
    type uint64;
    units "Bps";
    description
      "The maximum inbound network bandwidth available to the
       NSF in bytes per second (Bps)";
  }
}
```

The access information allows a Domain Name along with an IP address. Username and password information are included as well.

```
grouping nsf-access-info {
  description
    "Information required to access an NSF";
  leaf ip {
    type union {
      type inet:ip-address-no-zone;
      type inet:domain-name;
    }
    description
      "Either an IP (IPv4 or IPv6) address or the domain name of
       this NSF";
  }
  .
  .
  .
  leaf username {
    type string;
    description
      "The user name string identifying the credentials for the
       authentication.";
  }
  leaf password {
    type ianach:crypt-hash;
    description
      "The password for the username for the authentication.
       Any plain-text password must be converted to a hashed value
       as soon as possible";
  }
}
```

Major Updates of Registration Interface (4/4)

7. Security Considerations

The architecture of I2NSF Framework presents a risk to the implementation of security detection and mitigation activities. It is important to have an authentication and authorization method between the communication of the Security Controller and the DMS. The following are threats that need to be considered and mitigated:

- Compromised DMS with valid credentials: It can send falsified information to the Security Controller to mislead existing detection or mitigation devices. Currently, there is no in-framework mechanism to mitigate this, and it is an issue for such infrastructures. It is important to keep confidential information from unauthorized persons to mitigate the possibility of compromising the DMS with this information.

- Impersonating DMS: This involves a system trying to send false information while imitating as a DMS; client authentication would help the Security Controller to identify this invalid DMS.

Next Step

- We authors have addressed all the comments from AD Roman Danyliw for YANG Data Models of CFI and RI:
  - draft-ietf-i2nsf-consumer-facing-interface-dm-24
  - draft-ietf-i2nsf-registration-interface-dm-22

- We authors request the evaluation by IESG for those two drafts.

- I2NSF WG believes that it can discuss I2NSF WG Re-chartering.
  - All the five I2NSF YANG drafts (i.e., Capability, NFI, MI, CFI, and RI) are finished.