Revision Letter

I2NSF Consumer-Facing Interface YANG Data Model

- **Old Draft Name: draft-ietf-i2nsf-consumer-facing-interface-dm-08**
- **New Draft Name: draft-ietf-i2nsf-consumer-facing-interface-dm-09**

Jaehoon Paul Jeong

July 11, 2020

Dear Jan Lindblad,

I sincerely appreciate your valuable comments on the Consumer-Facing Interface Data Model document. Your comments use a bold font, and my answers use a regular font with the prefix [PAUL].

Document: draft-ietf-i2nsf-consumer-facing-interface-dm-09

Reviewer: Jan Lindblad

Review Date: March 20, 2020

Review Type: Working Group Last Call

Intended Status: Standards Track

1. **Line 107-204: The following identities are declared in the module, but never referenced. They should either have a common base with something or be referenced somewhere. If not, why are they defined here? They currently serve no purpose in this YANG module.**
   identity ddos {
   identity enforce-type {
   identity admin {
   identity time {

=> [PAUL] We set a common base in the identity ddos and remove the other identities (enforce-type, admin, and time) as it is not in used and serve no purpose.

OLD:

```
identity ddos {
  description
    "Identity for DDoS event types.";
}
```

NEW:

```
identity ddos {
  base security-event-type;
  description
    "Identity for DDoS event types.";
}
```

2. **Line 377: Defining a custom date-and-time type seems odd. You should probably use one that has already been defined**

typedef date-and-time {

=> [PAUL] We remove the custom date-and-time type and use a type defined from the existing ietf-yang-types.

OLD:

```
typedef date-and-time {
  type string {
    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d+)?'
      + '(Z|[\+\-]\d{2}:\d{2})';
  }
description
 "This is the format of date-and-time.";
  reference
   "RFC 3339: Date and Time on the Internet: Timestamps
    RFC 2579: Textual Conventions for SMIv2
    XSD-TYPES: XML Schema Part 2: Datatypes Second Edition";
}
leaf begin-time {
  type date-and-time;
  description
    "This is start time for time zone";
}
```

NEW:

```
import ietf-yang-types{
  prefix yang;
  reference "Section 3 of RFC 6021";
}
leaf start-time {
  type yang:date-and-time;
  description
    "This is start time for event.";
}
```

We also make a new typedef for time only without a date. This type is used to determine the period of repetition.

```
typedef time{
  type string {
    pattern '\d{2}:\d{2}:\d{2}(\.\d+)?'
      + '(Z|[\+\-]\d{2}:\d{2})';
  }
  description
    "This is the format of time.";
}
```

3. **Line 513: The leaf represents the name of a user, but the format is undefined. What should be the format for the string value? How would a user know what to configure here? Email addresses? If implementation dependent, say so.**

leaf name {

type string;

description

"This represents the name of a user.";

=> [PAUL] The leaf is for defining a user-group name. It is used for registering user-group to I2NSF database. We have made a different description of the leaf.

```
leaf name {
  type string;
  description
    "This represents the name of a user-group.
    A user-group name is used to map a user-group's
    name (e.g., employees) to an ip address.
    It is implementation dependent";
}
```

4. **Line 518: If no IP address information is specified for the user-group, what happens then? Is the user access accepted, rejected, or something else?**

uses ip-address-info;

=> [PAUL] The user-group is used to register user-group with the I2NSF database, and the IP address of a user is the required information. So, we let match-type be set to “mandatory true” in “uses ip-address-info”.

OLD:

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

NEW:

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

5. **Line 658: Key leaf declared mandatory. All keys are mandatory, so mandatory is not needed on this leaf.**

leaf policy-name {

type string;

mandatory true;

=> [PAUL] We removed “mandatory true” of policy-name as it is already mandatory as a key leaf.

```
leaf policy-name {
  type string;
  description
    "The name which identifies the policy.";
}
```

6. **Line 664: Users mentioned in the owners-ref should have full CRUD privileges to the policy. But what about everyone else? Should they have R(ead) privileges? Can anyone create new policies? If not, who can? If someone creates a policy but does not mention his own name among owners (e.g. misspells or does not get the format right), he will not be able to modify or remove the policy. If no owner is mentioned, then no one can.**

   uses owners-ref;

=> [PAUL] Revision 6 and 7 have a similar issue. Instead of using a custom security access control we use the available NACM (RFC 8341) access control to an entire policy as it is a better option. The NACM module provides more than enough for security access control. The data model uses the NACM default to handle CRUD privileges. We also make sure that non-registered users cannot create any security policy or rule.

```
nacm:default-deny-write;
```

7. **Line 682: Users mentioned in the owners-ref should have full CRUD privileges to the rule. But what about everyone else? Should they have R(ead) privileges? Can anyone create new rules, or only those that have full CRUD privileges for the policy? If someone creates a rule, but does not mention his own name among owners (e.g. misspells or does not get the format right), he will not be able to modify or remove the rule.**

   uses owners-ref;

=> [PAUL] Revision 6 and 7 have a similar issue. Instead of using a custom security access control we use the available NACM (RFC 8341) to control the security access as it is the better option. The NACM module provides more than enough for security access control. The data model uses the NACM default to handle CRUD privileges. We also make sure that non-registered users cannot create any security policy or rule.

```
nacm:default-deny-write;
```

8. **Line 673: Key leaf declared mandatory. All keys are mandatory, so mandatory is not needed on this leaf.**

   leaf rule-name {

   type string;

   mandatory true;

=> [PAUL] We removed “mandatory true” of rule-name as it is already mandatory as a key leaf.
NEW:

```
leaf rule-name {
  type string;
  description
    "This represents the name for the rule.";
}
```

9. **Line 697: Choice enforce-type has a description that I can't understand. What does this mean?**

   choice enforce-type {

   description

   "There are two different enforcement types;

admin, and time.
It cannot be allowed to configure
admin=='time' or enforce-time=='admin'.";

=> [PAUL] We removed the enforce-type admin since it is not used in our CFI data model.

10. Line 703: In case of enforce-type admin (whatever that means), a string value needs to be configured. What are the valid values for this leaf?

```
case enforce-admin {
    leaf admin {
        type string;
        description
            "This represents the enforcement type
            based on admin's decision.";
```

=> [PAUL] We removed the enforce-type admin since it is not used in our CFI data model.

11. Line 711: In case of enforce-type time, three times can be configured. What is the relation between enforce-time, and the other two (begin-time, end-time)?

```
case time {
    container time-information {
        description
            "The begin-time and end-time information
            when the security rule should be applied.";
        leaf enforce-time {
            type date-and-time;
            description
                "The enforcement type is time-enforced.";
        }
        leaf begin-time {
            type date-and-time;
            description
                "This is start time for time zone";
        }
        leaf end-time {
            type date-and-time;
            description
                "This is end time for time zone";
        }
    }
}
```

=> [PAUL] We designed a new data model for time-information. We remove the enforce-time as it is not in used. The new data model uses “start-date-time” and “end-date-time” to determine the starting and ending point of a policy. The security policy will start at “start-date-time” and end at “end-date-time”. Also, “container period” is added to determine the period of repetition. The period can only be accessed if the frequency is not only-once. We also add “leaf-list day” for weekly frequency to determine the repetition day every week, “leaf-list date” for monthly to determine the repetition date every month, and “leaf-list month-date” to determine the repetition date and month every year. Also, “yearly” is

added on the frequency leaf.

NEW:

```
container time-information {
  description
    "The time information when the security
    rule should be applied.";
  leaf start-date-time {
    type yang:date-and-time;
    description
      "This is the start date and time
      for policy.";
  }
  leaf end-date-time {
    type yang:date-and-time;
    description
      "This is the end date and time
      for policy. The policy will stop
      working after the specified
      end-date-time";
  }
  container period{
    when "/i2nsf-cfi-policy/rules/rule/event/frequency!='only-once'";
    description
      "This represents the repetition time. In case of
      frequency is weekly, the days can be set.";
    leaf start-time {
      type time;
      description
        "This is period start time for event.";
    }
    leaf end-time {
      type time;
      description
        "This is period end time for event.";
    }
    leaf-list day {
      when "/i2nsf-cfi-policy/rules/rule/event/frequency='weekly'";
      type identityref{
        base day;
      }
      description
        "This represents the repeated day of
        every week (e.g, monday, tuesday).
        More than one day can be specified";
    }
    leaf-list date {
      when "/i2nsf-cfi-policy/rules/rule/event/frequency='monthly'";
      type int32{
        range "1..31";
      }
      description
        "This represents the repeated date of
        every month. More than one date can be
        specified.";
    }
    leaf-list month-date {
      when "/i2nsf-cfi-policy/rules/rule/event/frequency='yearly'";
      type string{
        pattern '\d{2}-\d{2}';
      }
      description
        "This represents the repeated date and month
        of every year. More than one can be specified.
        Pattern used is (Month-Date / MM-DD).";
    }
  }
}
```

```
leaf frequency {
  type enumeration {
    enum only-once {
      description
        "This represents the rule is enforced
        only once immediately and not repeated.
        The policy will end at end-time.";
    }
    enum daily {
      description
        "This represents the rule is enforced
        on a daily basis. The policy will be
        repeated daily until the end-date.";
    }
    enum weekly {
      description
        "This represents the rule is enforced
        on a weekly basis. The policy will be
        repeated weekly until the end-date. The
        repeated days can be specified.";
    }
    enum monthly {
      description
        "This represents the rule is enforced
        on a monthly basis. The policy will be
        repeated monthly until the end-date.";
    }
    enum yearly {
      description
        "This represents the rule is enforced
        on a yearly basis. The policy will be
        repeated yearly until the end-date.";
    }
  }
  default only-once;
  description
    "This represents how frequent the rule
    should be enforced.";
}
```

12. Furthermore, the locally defined date-and-time type used includes both a date and time, which seems to be at odds with the example configurations in the draft. Example 9.2:

```
<rules>
    <rule>
        <rule-name>block_access_to_sns_during_office_hours</rule-name>
        <event>
            <time-information>
                <begin-time>2020-03-11T09:00:00.00Z</begin-time>
                <end-time>2020-03-11T18:00:00.00Z</end-time>
```

In the example, the rule-name "block_access_to_sns_during_office_hours " suggests that the begin-time and end-time should be times of day between which the policy should be enforced. E.g. every day between 9.00 and 18.00. If that is a valid use case, using a time type with a date doesn't make much sense. In the context of the policy that repeats "daily", how should the start date-and-time value "2020-03-11T09:00:00.00Z " be interpreted? What if it was "monthly"?

=> [PAUL] We have created a new data model to specify the repetition. An example can be seen below. In the example, the policy will start at 9:00:00 on 2020-03-11. The policy will be active during the interval of 09:00:00 to 18:00:00 on

every weekday (Monday, Tuesday, Wednesday, Thursday, and Friday). The policy will be terminated at 18:00:00 on 2020-12-31.

Example:
Usage example for weekly

```
<config>
  <i2nsf-cfi-policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy">
    <policy-name>security_policy_for_blocking_sns123</policy-name>
    <rules>
      <rule>
        <rule-name>block_access_to_sns_during_office_hours</rule-name>
        <event>
          <time-information>
            <start-date-time>2020-03-11T09:00:00.00Z</start-date-time>
            <end-date-time>2020-12-31T18:00:00.00Z</end-date-time>
            <period>
              <start-time>09:00:00Z</start-time>
              <end-time>18:00:00Z</end-time>
              <day>monday</day>
              <day>tuesday</day>
              <day>wednesday</day>
              <day>thursday</day>
              <day>friday</day>
            </period>
          </time-information>
          <frequency>weekly</frequency>
        </event>
        <condition>
          <firewall-condition>
              <source>employees</source>
          </firewall-condition>
        </condition>
        <actions>
          <primary-action>drop</primary-action>
        </actions>
      </rule>
    </rules>
  </i2nsf-cfi-policy>
</config>
```

13. Line 736: In the frequency leaf, the enumeration value only-once is for rules that don't repeat. But how long do they apply? A single packet? A single time the rule is triggered? How does a user know if the rule is still in effect, i.e. if the "once" has happened or not?

enum only-once {

=> [PAUL] We have made changes in the data model. For the case of only-once, the rule will happen continuously from start-date-time until end-date-time. We have edited the description to explain the value of only once.

NEW:

```
enum only-once {
  description
    "This represents the rule is enforced
    only once immediately and not repeated.
    The policy will continuously active from
    start time and terminated at end-time.";
}
```

14. Line 835: Maybe it's just my limited understanding of how threat-feeds work, but I wonder if this construct

with source and destinations for threat feeds is meaningful?

```
container threat-feed-condition {
    description
        "The condition based on the threat-feed information.";
    leaf-list source {
        type leafref {
            path "/i2nsf-cfi-policy/threat-preventions/threat-feed-list/name";
        }
        description
            "Describes the threat-feed condition source.";
    }
    leaf dest-target {
        type leafref {
            path "/i2nsf-cfi-policy/threat-preventions/threat-feed-list/name";
        }
        description
            "Describes the threat-feed condition destination.";
    }
}
```

=> [PAUL] We use Structured Threat Information Expression (STIX) as a reference for threat-feed information for threat-feed-info. This description of threat-feed-info can contain the information about a threat. This information is useful when security rule condition is based on the existing threat report (e.g., STIX-based threat information) gathered by other sources. The information carries a possible threat from a source or to a destination. We have also added STIX as a reference [STIX] for the I2NSF CFI YANG data model as follows.

```
grouping threat-feed-info {
  description
    "This is the grouping for the threat-feed-list";
  leaf threat-type {
    type identityref {
      base threat-feed-type;
    }
    description
      "This represents the type of the threat-feed.";
  }
  leaf server-ipv4 {
    type inet:ipv4-address;
    description
      "The IPv4 ip-address for the threat-feed server.";
  }
  leaf server-ipv6 {
    type inet:ipv6-address;
    description
      "The IPv6 ip-address for the threat-feed server.";
  }
  leaf description {
    type string;
    description
      "This represents the descriptions of a threat-feed.
      The description should include information, such as
      the type, related threat, method, and file type.
      Structured Threat Information Expression (STIX) can
      be used for description of a threat [STIX].";
  }
}
```

15. Line 920: Location groups can be configured, but there seems to be no references to them. How are they supposed to be used?

list location-group{

key "name";

uses location-group;

=> [PAUL] We added a new data model to use “list location-group”. The data model uses location-based source and destination as the condition.

NEW:

```
container location-condition {
  description
    "The condition for location based connection";
  leaf-list source {
    type leafref{
      path "/i2nsf-cfi-policy/endpoint-groups/location-group/name";
    }
    description
      "This describes the path to the location
      source reference.";
  }
  leaf-list destination {
    type leafref {
      path "/i2nsf-cfi-policy/endpoint-groups/location-group/name";
    }
    description
      "This describes the path to the location
      destination reference.";
  }
}
```

16. Line 931: Regarding point 16.1 in your revision letter, you say "We think list type of threat-feed-list can be configured more than one feed of the same type". I'm afraid that is not the case with the current YANG model. If you do wish to allow more than one threat-feed-list for the same threat-feed-type, you need to add an additional key to your threat-feed-list.

list threat-feed-list {

key "name";

description

"There can be a single or multiple number of

threat-feeds.";

uses threat-feed-info;

...

grouping threat-feed-info {

description

"This is the grouping for the threat-feed-list";

leaf name {

type identityref {

base threat-feed-type;

=> [PAUL] We change name into threat-type in the “grouping threat-feed-info” and add name in the “threat-feed-list”. We have changed the data model to enable multiple threat-feed-lists for the same threat-feed-type, including multiple threat feeds of the same type (e.g., DDoS attack and virus)

OLD:

```
grouping threat-feed-info {
  description
    "This is the grouping for the threat-feed-list";
  leaf name {
    type identityref {
      base threat-feed-type;
    }
    description
      "This represents the name of the threat-feed.";
  }

list threat-feed-list {
  key "name";
  description
    "There can be a single or multiple number of
    threat-feeds.";
  uses threat-feed-info;
```

NEW:

```
grouping threat-feed-info {
  description
    "This is the grouping for the threat-feed-list";
  leaf threat-type {
    type identityref {
      base threat-feed-type;
    }
    description
      "This represents the type of the threat-feed.";
  }

list threat-feed-list {
  key "name";
  description
    "There can be a single or multiple number of
    threat-feeds.";
  leaf name {
    type string;
    description
      "This represents the name of the threat-feed.";
  }
  uses threat-feed-info;
```

17. Generally, the indentation in the module is much improved. Some lines are still a bit off, however, so I would recommend using a tool that indents consistently.

=> [PAUL] We have tried to improve the indentation as your comments using a tool that indents consistently.

18. Generally, I also wonder whether there has been any discussion with implementors around the admin security model proposed here. As noted before, it's a bit different from everything else I have seen. Is it well thought through? Do implementors feel this is doable and user friendly? Currently there are no examples involving owner. Perhaps an example that sheds some light over how different users create, modify and see the various rules would shed some light over this.

=> [PAUL] The proposed security model might not be the best option to solve this problem. In Section 7, we discuss how to use NACM to control the module of the I2NSF Consumer-Facing Interface (CFI) for a user group. NACM provides a user group with a rule for the access control for the CFI. By using the part of the NACM YANG module below, we can give CRUD privileges on each user group. NACM can construct a user group of multiple users. Each user group can be controlled with a rule that gives privileges to the CFI module. The privileges are Create, Read, Update, and Delete. Thus, the access control for each user group can be set by using the NACM YANG data model. Also, in Section 10, we provide the audience with an XML configuration example of a user group's access control for I2NSF Consumer-Facing Interface.

```
list rule-list {
  key "name";
  ordered-by user;
  description
    "An ordered collection of access control rules.";
  leaf name {
    type string {
      length "1..max";
    }
    description
      "Arbitrary name assigned to the rule-list.";
  }
  leaf-list group {
    type union {
      type matchall-string-type;
      type group-name-type;
    }
    description
      "List of administrative groups that will be
     assigned the associated access rights
     defined by the 'rule' list.";
  }
  list rule {
    key "name";
    ordered-by user;
    description
      "One access control rule.";
    leaf name {
      type string {
        length "1..max";
      }
      description
        "Arbitrary name assigned to the rule.";
    }
    leaf module-name {
      type union {
        type matchall-string-type;
        type string;
      }
      default "*";
      description
        "Name of the module associated with this rule.";
    }
    leaf access-operations {
      type union {
        type matchall-string-type;
        type access-operations-type;
      }
      default "*";
      description
        "Access operations associated with this rule.";
    }
```

Thanks for your intensive and detailed comments to improve our draft.

Best Regards,

Paul

==========

Mr. Jaehoon (Paul) Jeong, Ph.D.

Associate Professor

Department of Software

Sungkyunkwan University

Office: +82-31-299-4957

Email: jaehoon.paul@gmail.com, pauljeong@skku.edu

Personal Homepage: http://iotlab.skku.edu/people-jaehoon-jeong.php