Revision Letter

I2NSF Consumer-Facing Interface YANG Data Model

(Old Draft Name: draft-ietf-i2nsf-consumer-facing-interface-dm-05 and New Draft Name: draft-ietf-i2nsf-consumer-facing-interface-dm-06)

Jaehoon Paul Jeong

07/25/2019

Hi Jan Lindblad,

I sincerely appreciate your detailed review on this draft. Your comments and questions use a bold font and my answers use a regular font, starting with the mark “=>”.

Reviewer: Jan Lindblad

Review result: On the Right Track

This is my YANG Doctor review of draft-ietf-i2nsf-consumer-facing-interface-dm-05.txt, and the YANG module ietf-i2nsf-cfi-policy in particular. The document is on the right track, but several important discussions remain, so is not quite ready for last call.

There are plenty of smaller things to fix in this module, see below for a list. There are a couple of potentially major issues in this module, however, that I would like to start with. The first relates to the management access control for this module.

1) Management access control

This RFC defines a new management access control mechanism, entirely unrelated to the traditional NACM model (RFC 8341/RFC 6536). It also specifies things like which auth mechanisms should be used by clients in specific domains when connecting to the management server. This appears highly disruptive to the existing base of NETCONF/RESTCONF servers.

To make it worse, the new security model described in this document is rather heavily unspecified. It may be that this security model is well thought through, but there is no language or other evidence that shows this in the text. The YANG model representation of it is broken in many places.

The RFC text refers to a device called "Security Controller". By reading the parent RFC 8329, which contains an overview of the architecture, I believe the Security Controller is the management (NETCONF/RESTCONF/...) server with the ietf-i2nsf-cfi-policy YANG model, and the user is supposed to connect directly with it. This means, I would think, that the management server would need to be the component to implement the new management access control model.

Key in this new mechanism is a leaf owner. There isn't much information about the owner leaf, but the RFC text relating to "owner" above Fig. 3 says:

Owner: This field contains the owner of the rule. For example, the person who created it, and eligible for modifying it.

And further, in the YANG module itself, we find:

```
  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.";
```

This is all that is mentioned about the owner leaf. That the type of the leaf is string does not give much clues as to how this is meant to work.

Then there are Policy Domains. A Policy Domain has an (optional) auth method (which is not correctly modeled in YANG) and a list of Policy Tenants. Each Policy Tenant represents one department or other part of the organization.

Policy Users are individual users that are allowed to access the management server to CRUD policies and rules. Either within a tenant (default) or domain. There is no linkage to which tenant or domain, however. Each Policy User has exactly one reference to an access profile in the Policy Role object. The only information this linkage provides is whether the access is read-only or read-write.

In summary, I believe significant work is required to get this to a workable state. In order to invent a new management access control mechanism, a wider discussion in the NETMOD working group would be needed.

Below I note some details that I came up with while trying to figure this out. Fixing them will not fix this issue as a whole.

- container policy-mgnt-auth-method should probably be a list

=> The container policy-mgnt-auth-method is now a list.

OLD: container policy-mgnt-auth-method
container policy-mgnt-auth-method {

description "This represents the list of authentication methods."; leaf auth-method {
    type string;
    description "This represents the authentication method name.";
}
...
}

NEW: list policy-mgmt-auth-method-instance

list policy-mgmt-auth-method-instance {
    ....
    choice policy-mgmt-auth-method {
        ....
        case password-based {...}
        case token-based {...}
        case certificate-based {...}

- container policy-role should probably be a list, and the list access-profile removed

=> policy-role is now a list.

OLD: poliy role	

container policy-role {
    uses meta;
    description
    list access-profile {
        uses meta;
        key "name";
 	
        leaf permission-type {
        type identityref {
        base permission-type;
        }
      default read-only;
      }	

NEW: list policy-mgmt-auth-method-instance	

The list policy role:
    list policy-role {
        key "role-type";
        leaf role-type {
            type identityref {
                base role-type;
            }
            description
          "This represents the role";
          }
	The role-type identities:
	
	 identity role-type {
	    description
	    "This is the base identity for the roles.";
	  }
	  identity user {
	    base role-type;
	    description
	    "This represents the identity of the user role.";
	  }
	  identity group {
	    base role-type;
	    description
	    "This represents the identity of any member of the
	    security policy's defined group.";
	  }
	  identity other {
	    base role-type;
	    description
	    "This represents the identity of anyone else.";
	  }

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

- list policy-tenant should probably not have any leaf domain

=> leaf domain in the policy-tenant have been removed.

- the leaf owner description talks about owner of the policy, while the leaf sits on an individual rule. Either the description or the leaf placement must be wrong.

=> the description and the leaf placement are now redone.

OLD: leaf owner	

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.";
 	
}	

NEW: new leaf owner	

leaf owner {
  type identityref {
    base owner;
  }
  mandatory true;
  description
   "This field defines the owner of this rule. Only the owner is authorized to modify the contents of the rule.";
}
	identity owner {
	  description
	  "This is the base identity for the owner";
	}
	identity dept-head {
	  base owner;
	  description
	  "This represents the identity of the head of department.";
	}
	identity manager {
	  base owner;
	  description
	  "This represents the identity of the manager of the department.";
	}
	identity employee {
	  base owner;
	  description
	  "This represents the identity of department employees.";
	}
	identity sec-head {
	  base owner;
	  description
	  "This represents the identity of the head of security.";
	}

	identity sec-admin {
	  base owner;
	  description
	  "This represents the identity of security admin.";
	}

- I believe there are more issues to look at as part of the "translation from UML to YANG" that this module has been subject to.

=> After reflecting your comments, I have not found any issue with translating UML to YANG yet.

2) container policy

The top level container policy can only exist in a single instance as currently modeled. Even if not stated explicitly in the document, I get the feeling that the author's intent is that it should be possible to have more than one policy in the system. If this is the case, container policy needs to change to list policy (and probably add a surrounding container policies). Doing so would also require updating all the leafrefs used throughout the module, to only select nodes in the current policy.

=> the policy is now a list.

OLD: container policy	NEW: list policy

container policy {
  leaf polict-name {...}
  list rule {...}
}
	list i2nsf-cfi-policy {
	  key "policy-name";
	  leaf policy-name {
	    type string;
	    mandatory true;
	  }
	  list rule {...}
	}

Another concern is that the name "policy" is very generic. There are many modules out there which already define a top level object called policy. Searching for container policy and list policy on yangcatalog.org gives several hundred pages with hits. Most of them will not be on the top level, but this shows that "policy" is a popular concept. How about calling the top level container cfi-policy or i2nsf-cfi-policy?

=> The policy name is changed to i2nsf-cfi-policy.

3) Are rules ordered?

In the RFC section 4, it says regarding list rule:

> **This field contains a list of rules. If the rule does not**
>
> **have a user-defined precedence, then any conflict must be**
>
> **manually resolved.**

I'm not sure what a "user-defined precedence" is, or at least I can't find anything like that in the YANG. So how are the rules meant to be applied on the devices? What does "manually resolved" mean, who does that when and how? If different operators cannot see each other's rules, does that mean that "manually resolved" just got harder?

=> The description of list rule is changed as follows:

OLD
"This field contains a list of rules. If the rule does not have a user-defined precedence, then any conflict must be manually resolved."
NEW: list policy
"This field contains a list of rules. These rules are defined for 1) communication between two Endpoint Groups, 2) for preventing communication with externally or internally identified threats, and 3) for implementing business requirement such as controlling access to internal or external resources for meeting regulatory compliance or business objectives. An organization may restrict certain communication between a set of user and applications for example. The threats may be from threat feeds obtained from external sources or dynamically identified by using specialty devices in the network." Rule conflict analysis should be triggered by the monitoring service to perform an exhaustive detection of anomalies among the configuration rules installed into the security functions."

Each policy contains a set of rules in a list. This list is keyed by rule name and modeled as ordered-by system, i.e. rules will be sorted alphabetically. Is it up to the server to decide in which sequence these rules are applied. I suppose they could be overlapping, leading to different results based on the order rules are applied, and there could be performance implications depending on how this is done. If several policies are allowed, the issue only grows.

=> There should be some kind of rule-ordering and conflict management mechanism residing on the server side. This is out of the scope of the Consumer-Facing Interface.

4) Restricted to IPv4

The module specifically uses IPv4 in many places. Would it not make sense to use ip-address instead, to allow the use of IPv6 also, or is there some particular reason to exclude IPv6? If so, it would be good to mention this reason.

=> The module now supports both IPv4 and IPv6.

In a couple of places, an address range is specified, and the start address is an ipv4-address while the end address is an ip-address. Surely that must be a mistake.

=> Yes, you are right. It should be ipv4-address.

5) Many optional leafs

The RFC text repeats many times: The XYZ object SHALL have the following information... This could be understood as the succeeding information items are mandatory. There are not a single mandatory element (apart from keys) in the entire module, however. And only three leafs with a default statement.

It would be appropriate to go through the module and add mandatory and defaults where they make sense. For the leafs that should remain optional, it would be good if something could be added to the description regarding what the server behavior is when the leaf has no value, when

that is not entirely obvious. Actually, it is often good to write even when it is obvious, because otherwise the behavior is technically undefined, and implementations could get away with bad behavior even when all know it's wrong.

A couple of examples: what happens if the owner leaf is left unset?

primary-action? enforce-type? recur? policy-tenant/domain?

=> The leafs in the module, including the ones you mentioned, now have mandatory statement if required. Secondary-action, for example, does not include the mandatory statement because it can be left out if it is not necessary, as the primary action will always have a set value.

6) Many strings

In the module, the type string is used for names, which is great, but also for a many cases where some certain format of the content is expected, but not defined. There is no reason to believe that will lead to interoperable solutions. It will also leave users frustrated with little information to guess what type of data in which format to fill in. This needs fixing.

=> According to the comments, appropriate types are used rather than the type string.

OLD
294: leaf-list 322: leaf-list content 388: leaf begin-time 393: leaf end-time 553: leaf primary-action 561: leaf secondary-action 581: leaf owner 697: leaf auth-method 823: leaf threat-feed-description 839: leaf-list signatures
NEW: fixed types.
294: leaf-list protocol ==> protocol type is added (FTP, SSH, HTTP, HTTPS, etc.) 322: leaf-list content ==> this is for the admin (or an entity who is creating or modifying the rule with such content) to copy & paste the payload content. It is meant to be string. 388: leaf begin-time ==> This type is replaced with yang:date-and-time 393: leaf end-time ==> This type is replaced with yang:date-and-time 553: leaf primary-action ==> It is no longer a string type; primary-action identities are added. 561: leaf secondary-action ==> It is no longer a string type; secondary-action identities are added. 581: leaf owner ==> identities for owner is now added. The type is now identitref not string. 697: leaf auth-method ==> leaf auth-method is removed. Instead, auth-intance-type is added with auth-type (authentication type) identities. Its type is identityref now.

823: leaf threat-feed-description ==> this is for the admin (or an entity who is creating or modifying the rule related to threat feeds) to describe what information is obtained from a treat feed. It is meant to be string.

839: leaf-list signatures ==> The signatures are like track of a security threats. They are usually a bunch of strings (or binary codes), and used to generate a security rule as part of its contents. Therefore, the type for the entries in the leaf-list signature should remain as strings.

7) leaf date

The grouping meta contains a name and a date leaf. The name is clearly meant to be a configuration item filled in by the client. I'm not sure about the date leaf, however. The description says

```
description "This is the date when the entity is

created or modified.";
```

The date leaf is currently modeled as config true, i.e. is expected to be filled in by the client. It is not customary to have this sort of meta information in YANG modules, but it would make some sense if this was config false, i.e. computed by the server itself.

If that is the intent, the specification should only make the date leaf config false and clarify how these time stamps should be calculated. Do they pertain to this particular list entry, or would they be updated if any child entry to this list entry was modified (similar to etags).

If the intent is that clients should fill this in if they want, for their own optional housekeeping, I think that needs to be stated clearly. If the intent is to make it mandatory for clients to keep these timestamps up to date, I think the model is broken. What implications would it have if they were not used?

=> You are right. The clients should not be entering the date information by themselves. The date is there to specify the date of entry. This information is automatically recorded from the server side where the policy/rules are stored in a secure databse (e.g., cloud).

8) container policy-mgnt-auth-method

The container has a description that says

```
    "This represents the list of authentication methods.";
```

A container is obviously not a list, so exactly what the author has in mind is somewhat unclear. At the top of the container, there is a leaf

```
leaf auth-method {

  type string;
```

description

"This represents the authentication method name.";

The format and usage of this string is highly unclear, but more importantly, it seems to speak of a single authentication method, not a list.

=> The policy-mgnt-auth-method is now choice with different authentication cases.

Following this leaf is a number of lists; list password-based, list token-based, list certificate-based, list ipsec-method, list single-sign-on. Are these lists under container policy-mgnt-auth-method mutually exclusive, or can several of them be configured at the same time?

=> They are mutually exclusive with a choice statement.

This would need some clearing up. As this is an area where we can anticipate new methods being introduced over time, it would be good if the model advised how to extend with further auth methods.

=> This is clarified as follows.

OLD

```
container policy-mgnt-auth-method {
      leaf auth-method {...}
      leaf mutual-authentication {...}
      list password-based {
        key "password";
        leaf password {...}
      }
      list token-based {
        key "token";
        ...
      ...
```

NEW: The authentication methods are now choices. If this model needs to be extended for new authentication methods, simply create an identity, grouping, and case for the new method.

```
    list policy-mgmt-auth-method-instance {
      key "auth-instance-type";
      description
      "This represents the list of instances for
      policy management authentication methods.";

      leaf auth-instance-type {
        type identityref {
          base auth-type;
        }
        description
        "This identifies whether the authentication type
        is server authentication, client authentication,
        or both.";
      }
      choice policy-mgmt-auth-method {
        description
        "This represents the choices for which
        authentication method is used.";
        case password-based {
        uses password-based-method;
```

```
         }
         case token-based {
           description
           "This represents the token-based method.";
           uses token-based-method;
         }
         case certificate-based {
           description
           "This represents the certificate-based-method.";
           uses certificate-based-method;
         }
         case ipsec {
           description
           "This repreents authentication method based on IPSEC.";
           uses ipsec-method;
         }
       }
     }
```

9) More on container policy-mgnt-auth-method

RFC text sec 5.1 says

> **Authentication method to be used for this**
>
> **domain.   It should be a reference to a "Policy-Management-**
>
> **Authentication-Method" object**

But in fact there is only one instance of container policy-mgnt-auth-method. Maybe this is meant to be a list?

=> Yes, it is a list now as shown in the previous table.

In the RFC Fig. 8 YANG tree representation authentication-method points like this

```
+--rw authentication-method?     ->
/policy/multi-tenancy/policy-mgnt-auth-method/name
```

In the YANG module itself, the path is different:

```
path
"/policy/multi-tenancy/policy-mgnt-auth-method/ipsec-method/method";
```

Also, the abbreviation of management as "mgnt" is not very common. It is usually "mgmt". In order to reduce misspellings of policy-mgnt-auth-method I suggest renaming it to policy-mgmt-auth-method. There is actually already one such misspelling in this YANG module itself.

=> The paths are corrected and checked. The abbreviation “mgnt” is not corrected to “mgmt.”

OLD: current path
“/policy/multi-tenancy/policy-mgnt-auth-method/name” & "/policy/multi-tenancy/policy-mgnt-auth-method/ipsec-method/method"
NEW: corrected path
"/i2nsf-cfi-policy/multi-tenancy/policy-mgmt-auth-method-instance/ipsec-method/method"

10) One cert server per cert type?

List certificate-based allows the configuration of certificate servers. Is there usually a different server based on certificate type? Max one server can be configured per certificate type, and max three total as there are three certificate types. If this is what you want, the YANG says it nicely.

=> The list now has the key as certificate-server name, and can have multiple certificate types (leaf-list)

OLD: once certificate-server per certificate type	**NEW**: The grouping for the certificate-based-method is defined as below. In the below grouping, for single certiciate server, there are a list of certificates managed.

list certificate-based {
 	
  key "certificate";
 	
  leaf certificate {...}
 	
  leaf certificate-server {...}
 	
}
	list cert-server-list {
	    ...
	    leaf cert-server-name {...}
	    leaf cert-server-ipv4 {...}
	    leaf cert-server-ipv6 {...}
	    list certificate {
	      key "cert-type";
	      description
	      "This represents the certificate-types.";
	      leaf cert-type {
	        type identityref {
	          base certificate-type;
	        }
	        description
	        "This represents a certificate type.";
	      }
	    }
	  }

11) Enumerations vs. identities

203: certificate-type is defined as an enumeration. This means the module will have to be revised if any new certificate types need to be supported one day. An identity may be more appropriate here, as I expect new certificate types (or formats) may come out.

=> Enum -> identity

366: enforce-type is defined as an enumeration. This means the module will have to be revised if any new enforcement types need to be supported one day. An identity or choice may be more appropriate here. Then an additional module could add new identities or augment the choice.

=> Enum -> identity

55: permission-type is defined as an identity. Unless you foresee the need for other values than read-only and read-write, you could use enumeration here instead. It could simplify the module a little, but identity is fine too.

=> identity -> enum

168: continent is defined as an identity. Unless you foresee the need for new continents, you could use enumeration here instead. It could simplify the module a little, but identity is fine too.

=> identity -> enum

145: identity ransomeware is misspelled. Should be identity ransomware

=> typo corrected

OLD	NEW
203: certificate-type: enumeration 366: enforce-type: enumeration 55: permission-type: identity 168: continent: identity 145: ransomeware	203: certificate-type: identity (cer, crt, key...) 366: enforce-type: identity (admin, time, event) 55: permission-type: identity (read, write, execute, read-and-write, read-and-execute, write-and-execute, no-permission) 168: continent: identity 145: ransomware

12) Reference RFC 6087

The RFC text refers to RFC 6087, which is obsoleted by RFC 8407. Update the reference?

=> Changed to RFC8407

13) YANG trees

The RFC text has many figure descriptions like this

Figure X show the XML instance

What is shown is actually a YANG tree, so the text should be updated to say

Figure X show the YANG tree

=> Changed the descriptions

OLD	NEW
Figure X show the XML instance of ...	Figure X show the YANG tree of ...

14) Indentation

The indentation of the YANG module is broken on many, many lines. Make sure to use a YANG indentation tool before publishing the result.

=> The indentation is now corrected.

15) Revision statement

The revision statement at the top of the file needs to be rewritten before publication.

=> I will rewrite the revision statement after YANG correction

16) container recursive

The container recursive on line 399 seems to be about how rules recur, not recurse (which is something completely different). Maybe change to the whole container recursive to

=> Changed the container recursive to leaf frequency

OLD	**NEW:** changed the leaf name

leaf recursive-type{
   type enumeration{
      enum daily {...}
      enum weekly {...}
      enum monthly {...}
   }
}
	leaf frequency {
	   type enumeration {
	      enum only-once {...}
	      enum daily {...}
	      enum weekly {...}
	      enum monthly {...}
	   }
	   default only-once;
	}

17) String passwords

Passwords are modeled as strings in a couple of places. This is not acceptable from a security point of view. Use one of the cryptographic types for passwords, such as ianach:crypt-hash RFC 7317.

666: leaf password

710: leaf password

The leaf password on line 710 has the description

> **"This should be defined using the**
>
> **regular expression.";**

What does that mean? Authentication, password, regular expression? I don't get it.

=> Changed the cryptographic types for passwords to ianach:crypt-hash, RFC 7317

=> Changed the description to “The crypto-hash mechanism for this entry is ianach:crypt-hash.”

18) Grouping descriptions

A number of groupings have descriptions like

This grouping is to remove repetition of

'name' and 'ip-address' fields.";

This is not exactly true since these groupings are used only once (hence no repetition). Describe instead what the grouping is used for or what it contains.

=> Grouping meta is removed in the latest version

231: grouping meta

=> grouping meta is removed.

280: grouping user-group

=> Description is changed to "The grouping for user-group entities, and contains information such as name & ip-address."

301: grouping location-group

=> Description is changed to "The grouping for location-group entities, and contains name & continent information."

316: grouping payload-string

=> Description is changed to "The grouping for payload-string content. It contains information such as name and string content."

19) container time-information

Is container time-information only relevant for enforce-type == time-enforced? If so, a when expression on the container may be useful. Or, even better, make the enforce-type a choice and have the time related content inside the choice case instead.

=> The enforce-type is now a choice, and time-information container is included in that choice.

OLD: leaf, enum type.	**NEW:** choice-case, identityref type.	

leaf enforce-type {
  type enumeration{
    enum admin-enforced {...}
    enum time-enforced {...}
    enum event-enforced {...}
  }
}
	choice enforce-type {
	  case admin {
	    leaf admin {
	      type identityref {
	        base enforce-type;
	      }
	    }
	  }
	  case time {
	    container time-information {
	      leaf enforce-time {...}
	      leaf begin-time {...}
	      leaf end-time {...}
	    }
	  }
	}
		identity enforce-type {...}
		  identity admin {
		    base enforce-type;
		  }
		  identity time {
		    base enforce-type;
		  }
		}

Thanks for your valuable comments.

Best Regards,

Paul Jeong

20) container rate-limit

Is uint8 a good type to use for the rate limiting function? Would it never (now or in the future) make sense to limit to more than 255 packets per second?

486: container rate-limit

==> increased the size to uint16.

21) container condition

This container seems to hold the configuration of many different triggering conditions. Would several of these apply at the same time in a given rule?

==> No. the user (admin) can pick a specific condition to apply for each policy.

There are many instances of container source-target and container destination-target nested underneath. They seem to serve no purpose. Maybe consider simplifying the model by removing them, unless they have some clever function for the future.

==> The source and destination-targets nested in each condition are bound to their conditions.

OLD: container condition	**NEW:** choice-case condition
container condition {	container condition {
container firewall-condition {	choice condition {
container source-target {...}	case firewall-condition {
container destination-target {...}	container firewall-source {...}
}	container firewall-destination {...}
container ddos-condition {	}
container source-target {...}	case ddos-condition {
container destination-target {...}	container ddos-source {...}
container rate-limit {...}	container ddos-destination {...}
}	container rate-limit {...}
container custom-condition {	}
container source-target {...}	case custom-condition {
container destination-target {...}	container custon-source {...}
}	container custom-destination {...}
container threat-feed-condition {	}
container source-target {...}	case threat-feed-condition {
container destination-target {...}	container threat-feed-source {...}
}	container threat-feed-destination {...}
}	}
	}
	}

22) Transactionality

Section 10.1 starts with the words

In order to create a rule of a security policy, it is essential to

first register data (those which are used to form such rule) to the database.

This could lead some implementors to believe it is acceptable to require that the endpoints are committed separately from the rest of the configuration. The actual requirement is that the referenced endpoints exist at the time the transaction is committed. I.e. they could very well be part of the same

transaction.

OLD: current description
In order to create a rule of a security policy, it is essential to first register data (those which are used to form such rule) to the database.
NEW: new description
“If new endpoints are introduced to the network, it is necessary to first register their data to the database. For example, if new members are newly introduced in either of three different groups (i.e., user-group, device-group, and payload-group), each of them should be registered with information such as ip-addresses or protocols used by devices.”

23) Incomplete XML examples

The XML snippets in examples in Fig 24-27 use XML namespaces incorrectly. It's easy to fix. Just make sure the first line with <ietf-i2nsf-cfi-policy:policy> looks like this instead:

<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy">

==> Changed to “<policy xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy">”

The last example, in section 10.4 is talking about <encrypt>. There is no such

in the YANG module (today).

```
<encrypt>
   <ipsec-method>ipsec-ike</ipsec-method>
</encrypt>
```

The XML loads fine if you just remove the encrypt tag, so maybe you mean:

```
<ipsec-method>ipsec-ike</ipsec-method>
```

OLD: xml example	**NEW:** xml example
<encrypt> <ipsec-method>ipsec-ike</ipsec-method> </encrypt>	<ipsec-method> <method>ipsec-ikeless</method> </ipsec-method>

(End of list)