NACM restructuring proposal

IETF 80

Martin Björklund
mbj@tail-f.com

Andy Bierman
andy.bierman@brocade.com
Problems with current NACM 1(2)

- Recall that there are four lists with rules:
  - module-rule, rpc-rule, data-rule, notification-rule

- Each such list is flat.
  - No mechanism to group related rules

- Mixes *who* has access to some objects with *what* those objects are
  - Makes task / feature based rules difficult to maintain (see example on next slide)
### Problems with current NACM 2(2)

Since the rules are spread out over four different tables, it is difficult to see which rules logically belong together.

<table>
<thead>
<tr>
<th>Module Rule</th>
<th>Description</th>
<th>Allowed Group</th>
<th>Rights</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>module-rule acme-system sys1</code></td>
<td></td>
<td>*</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>module-rule ietf-routing r1</code></td>
<td></td>
<td>[ router-adm ]</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>module-rule ietf-system sys2</code></td>
<td></td>
<td>[ oper ]</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>rpc-rule acme-interface reset rp1</code></td>
<td></td>
<td>[ admin oper ]</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>rpc-rule acme-interface reset rp2</code></td>
<td></td>
<td>*</td>
<td></td>
<td>deny</td>
</tr>
<tr>
<td><code>rpc-rule ietf-system reboot rp3</code></td>
<td></td>
<td>[ sys-admin ]</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>data-rule allowuser</code></td>
<td></td>
<td>*</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>data-rule readif</code></td>
<td></td>
<td>[ sys-adm ]</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>data-rule allowpasswd</code></td>
<td></td>
<td>/users/user[name=${USER}]/password</td>
<td></td>
<td>permit</td>
</tr>
<tr>
<td><code>data-rule denyuser</code></td>
<td></td>
<td>/users/user</td>
<td></td>
<td>deny</td>
</tr>
<tr>
<td><code>notification-rule ietf-system config-change chg</code></td>
<td></td>
<td>*</td>
<td></td>
<td>deny</td>
</tr>
</tbody>
</table>
Proposed solution 1(2)

- Introduce named collections of rules, *rule lists*. Each such rule-list contains all functionally related rules.
  - Example: an administrator can define one rule-list per common task in the system: *system, routing, vpn, accounting, …*

- Make a choice of the current four different rule types, so there is just one list of rules in a rule list.

- So, instead of four flat lists, we have one list nested in another:

OLD:

```plaintext
list module-rule {
  key "module-name rule-name";
  ...
}
list rpc-rule {
  key "module-name rpc-name rule-name";
  ...
}
list data-rule {
  key "rule-name";
  ...
}
list notification-rule {
  key "module-name notification-name rule-name";
  ...
}
```

NEW:

```plaintext
list rule-list {
  key name;
  ...
  leaf module-name { ... }
  choice rule-type {
    case rpc { ... }
    case notification { ... }
    case path { ... }
  }
  leaf action { ... }
}
```
Proposed solution 2(2)

- Move the `allowed-groups` leaf from the rule into the `rule-list`. This makes it possible to define the rules for one task without worrying about which groups have access to it.

  - Example: A vendor can choose to pre-populate the data store with rule-lists for common tasks applicable to his type of device. An operator can then assign groups to these tasks. Another operator might add his own tasks.

```
list rule-list {
  key name;
  ordered-by user;
  leaf name { ... }
  leaf-list allowed-groups { ... }
  leaf module-name { ... }
  choice rule-type {
    case rpc { ... }
    case notification { ... }
    case path { ... }
  }
  leaf action { ... }
}
```
Example

```plaintext
rule-list common-system
  allowed-group *
  rule own-passwd
    path /users/user[name=$USER]/password
    allowed-rights *
    action permit

rule ietf-sys
  module ietf-system
  allowed-rights read
  action permit

rule acme-sys
  module acme-system
  allowed-rights *
  action permit

rule-list system-adm
  allowed-group [ sys-adm ]
  rule users
    path /users/user
    allowed-rights *
    action permit

rule ietf-sys
  allowed-rights *
  action permit
```
Open Issues

- Is two levels of nesting enough?
- A common (?) use case is to define one rule-list for a task, and let some groups access it read-write, and some read-only. This is not directly supported – you would need to define two different rule-lists, e.g. `routing-admin` and `routing-read`.
- By moving the allowed-groups check from the rule to the rule-list, we loose some flexibility. If we really need special handling of a rule for some group, this rule needs to be defined in a separate rule-list.
- Would it be useful with any objects to help debug a NACM configuration?
  - `rpc get-rules group-name ---> list of rules`
  - `rpc check-path group-name path ---> rule execution trace`