SNMP Version 3

More about VACM and USM

Nick Urbanik <nicku(at)nicku.org>

Goals of SNMPv3 (RFC 3411)

- Avoid reinventing the wheel—use existing work
- Support secure `set` operation
- Support forward and backward compatibility
- Support remote configuration
  - USM and VACM configuration is through SNMP tables and variables
- Security protection against:
  - modification of information by unauthorised parties
  - an unauthorised person masquerading as an authorised person
  - message stream modification by reordering, delaying or replaying exchanges
  - disclosure (eavesdropping)

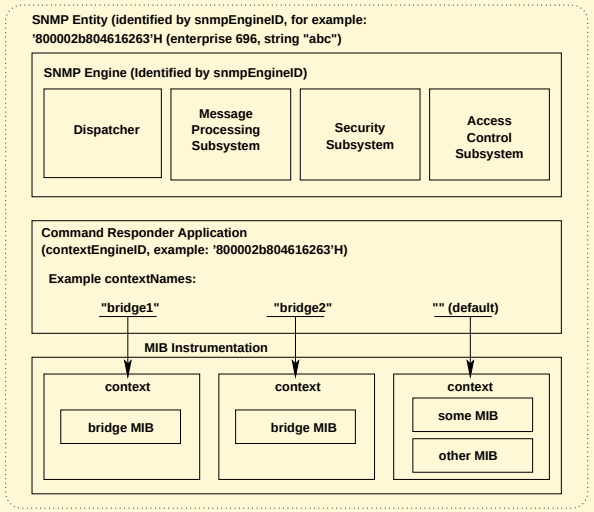
VACM

- The *View-based Access Control Model* (*VACM*)
- VACM has five main components, as we mentioned earlier:
  - *groups* of users
  - *security level*, i.e., `v1`, `v2c`, `usm`
  - *contexts* — see slide §4
  - *MIB views, view families* — see slide §15
  - *access policy*, i.e., read only, read-write, notify, no access.
- How do we set up SNMPv3 users on agents and network management software?
- How do we control access to a subset of MIB variables on an agent?

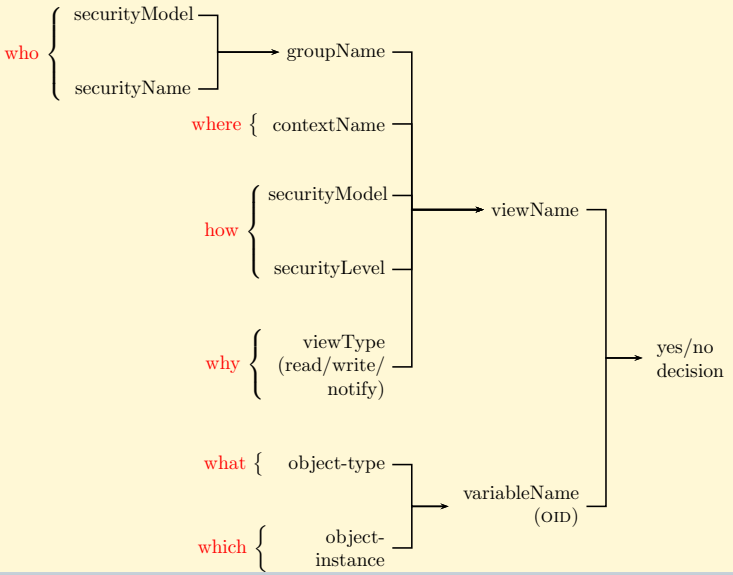
Context

- An SNMP *context* is a collection of management variables accessible by an SNMP entity.
- Gives a way to group variables into collections with different access policies.
- Example from RFC 3411: See slide §5
  - The engine uses the bridge MIB defined in RFC 1493
  - but the engine keeps management information for two separate bridges labeled `bridge1` and `bridge2`
  - Could be that neither bridge directly supports SNMP, so another device on the LAN collects data from the bridges using some other method
  - Makes this information available within the *context*

Context Example from RFC 3411



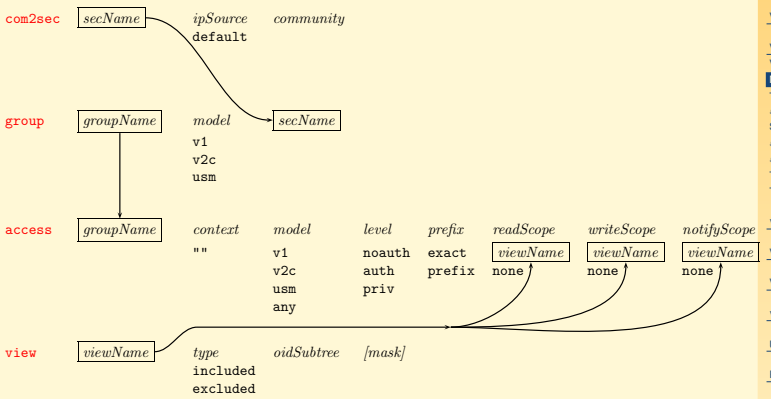
isAccessAllowed from RFC 3415



VACM on Net-SNMP

- Net-SNMP uses four keywords to set up VACM in `/etc/snmp/snmpd.conf`:
  - **`com2sec`**
  - **`group`**
  - **`view`**
  - **`access`**
- These set up access control to variables on the agent.
  - `access` and `view` determine *what* access is being controlled to.
  - `group` and `com2sec` determine *who* has this access.

Net-SNMP VACM



The `access` Keyword

- Specifies which group has access to which parts of the MIB tree
- Has 8 parameters. Syntax (all on one line):

`access` ⟨*group*⟩ ⟨*context*⟩ ⟨*secmodel*⟩ ⟨*seclevel*⟩ ⟨*prefix*⟩ ⟨*readview*⟩ ⟨*writeview*⟩ ⟨*notifyview*⟩

- Last three parameters ⟨*readview*⟩ ⟨*writeview*⟩ ⟨*notifyview*⟩ are *views*, defined by `view` statements.
  - Indicate which part of the MIB tree has read access, which part of tree has write access, and which part has permission for access to send notifications (i.e., traps or inform requests)
- The ⟨*group*⟩ parameter is defined by a `group` statement
  - Represents a group of users
- Default ⟨*context*⟩ is the empty string `""`. See slide §4.

`access`: Security Model, Security Level

- The parameter ⟨*secmodel*⟩ is the *Security Model*.
  - Can be one of: `any`, `v1`, `v2c` or `usm`.
  - Should be set to match the SNMP version of clients that will connect to this agent.
- Parameter ⟨*seclevel*⟩ *Security Level* tells whether we use authentication or encryption
  - Can be one of `noauth`, `auth`, or `priv`
  - Note that community strings are not counted as authentication, so for SNMPv1 and SNMPv2 we specify `noauth`
  - `priv` (privacy) means that we use both strong authentication *and* encryption.

`access`: The ⟨*prefix*⟩ Parameter

- The ⟨*prefix*⟩ parameter to `access` can be either `exact` or `prefix`.
- Indicates whether context name needs to match exactly or whether only the first part of the context name needs to match.
- The default value is `exact`.

`access` with SNMPv1, v2c

- For SNMPv1 and SNMPv2c clients
  - Security Level will be `noauth`, and
  - `context` will be empty (the empty string).

The `com2sec` keyword

- Maps a *community string* and a source IP or network address to a *security name* (user name).
- Syntax:
  `com2sec` ⟨*securityName*⟩ ⟨*source*⟩ ⟨*community*⟩
  - The security name is used by the `group` keyword — see §14
  - Source can be a hostname, a subnet or the word "`default`"
    - A subnet can be written as IP/mask or IP/BITS, e.g., our lab subnet can be written as 172.19.64.0/255.255.192.0 or 172.19.64.0/18.
- Only needed for access control with SNMPv1 and v2c
  - Not used with SNMPv3

The `group` Keyword

- maps pairs of *Security Model* and *Security Name* to a group name.
- Syntax:
  `group` ⟨*groupName*⟩ ⟨*securityModel*⟩ ⟨*securityName*⟩
- A Security Model is one of `v1`, `v2c` or `usm`.
- The *Security Name* is the *user name*.
- All members of one group have the same access rights.
- A user cannot belong to more than one group for each of the three security models.

Views and the `view` Keyword

- The view determines what part of the MIB access is controlled to.
- Uses concept of a *subtree*.
  - A *subtree* is a node in the MIB tree and all the elements under that node.
  - In other words, all the MIB elements in a subtree have the same common prefix.
- Syntax:
  `view` ⟨*viewName*⟩ ⟨*incl/excl*⟩ ⟨*subtree*⟩ ⟨*mask(optional)*⟩

The `view` Keyword — 2

- ⟨*incl/excl*⟩ can be either "`included`" or "`excluded`"
  - "`included`" means that the MIB view includes all the elements of the subtree;
  - "`excluded`" means that the MIB view excludes all the elements of the subtree.

The View Mask — 1

- The optional view mask allows the access control to select individual rows in a table.
- RFC 3415 calls this a *family of subtrees*, since a row of n elements can be also represented by n subtrees
- RFC 3415 calls the mask the *family mask*

The Network Interface Table, `ifTable`

- Under `mib-2` is the important `ifTable`
  - Provides statistics on each network interface
  - includes such things as network traffic, errors,. . .
  - One row in the table for each network interface

Walking `ifTable` — 1

```
$ snmpbulkwalk -v 2c -c public localhost ifTable
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifMtu.1 = INTEGER: 16436
IF-MIB::ifMtu.2 = INTEGER: 1500
IF-MIB::ifSpeed.1 = Gauge32: 10000000
IF-MIB::ifSpeed.2 = Gauge32: 100000000
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:1:3:45:99:12
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
IF-MIB::ifAdminStatus.2 = INTEGER: up(1)
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
IF-MIB::ifOperStatus.2 = INTEGER: up(1)
IF-MIB::ifInOctets.1 = Counter32: 1073820735
IF-MIB::ifInOctets.2 = Counter32: 1620632733
```


Walking `ifTable` — 2

```
IF-MIB::ifInUcastPkts.1 = Counter32: 2950449
IF-MIB::ifInUcastPkts.2 = Counter32: 105216646
IF-MIB::ifInDiscards.1 = Counter32: 0
IF-MIB::ifInDiscards.2 = Counter32: 0
IF-MIB::ifInErrors.1 = Counter32: 0
IF-MIB::ifInErrors.2 = Counter32: 0
IF-MIB::ifOutOctets.1 = Counter32: 1073821769
IF-MIB::ifOutOctets.2 = Counter32: 2594849796
IF-MIB::ifOutUcastPkts.1 = Counter32: 2950461
IF-MIB::ifOutUcastPkts.2 = Counter32: 81734428
IF-MIB::ifOutDiscards.1 = Counter32: 0
IF-MIB::ifOutDiscards.2 = Counter32: 0
IF-MIB::ifOutErrors.1 = Counter32: 0
IF-MIB::ifOutErrors.2 = Counter32: 0
IF-MIB::ifOutQLen.1 = Gauge32: 0
IF-MIB::ifOutQLen.2 = Gauge32: 0
IF-MIB::ifSpecific.1 = OID: SNMPv2-SMI::zeroDotZero
IF-MIB::ifSpecific.2 = OID: SNMPv2-SMI::zeroDotZero
```

ifTable in Numbers — 1

```
$ snmpbulkwalk -v 2c -On -c public localhost ifTable
.1.3.6.1.2.1.2.2.1.1.1  = INTEGER: 1
.1.3.6.1.2.1.2.2.1.1.2  = INTEGER: 2
.1.3.6.1.2.1.2.2.1.2.1  = STRING: lo
.1.3.6.1.2.1.2.2.1.2.2  = STRING: eth0
.1.3.6.1.2.1.2.2.1.3.1  = INTEGER: softwareLoopback(24)
.1.3.6.1.2.1.2.2.1.3.2  = INTEGER: ethernetCsmacd(6)
.1.3.6.1.2.1.2.2.1.4.1  = INTEGER: 16436
.1.3.6.1.2.1.2.2.1.4.2  = INTEGER: 1500
.1.3.6.1.2.1.2.2.1.5.1  = Gauge32: 10000000
.1.3.6.1.2.1.2.2.1.5.2  = Gauge32: 100000000
.1.3.6.1.2.1.2.2.1.6.1  = STRING:
.1.3.6.1.2.1.2.2.1.6.2  = STRING: 0:1:3:45:99:12
.1.3.6.1.2.1.2.2.1.7.1  = INTEGER: up(1)
.1.3.6.1.2.1.2.2.1.7.2  = INTEGER: up(1)
.1.3.6.1.2.1.2.2.1.8.1  = INTEGER: up(1)
.1.3.6.1.2.1.2.2.1.8.2  = INTEGER: up(1)
.1.3.6.1.2.1.2.2.1.10.1 = Counter32: 1073820735
.1.3.6.1.2.1.2.2.1.10.2 = Counter32: 1620632733
```


ifTable in Numbers — 2

```
.1.3.6.1.2.1.2.2.1.11.1 = Counter32: 2950449
.1.3.6.1.2.1.2.2.1.11.2 = Counter32: 105216646
.1.3.6.1.2.1.2.2.1.13.1 = Counter32: 0
.1.3.6.1.2.1.2.2.1.13.2 = Counter32: 0
.1.3.6.1.2.1.2.2.1.14.1 = Counter32: 0
.1.3.6.1.2.1.2.2.1.14.2 = Counter32: 0
.1.3.6.1.2.1.2.2.1.16.1 = Counter32: 1073821769
.1.3.6.1.2.1.2.2.1.16.2 = Counter32: 2594849796
.1.3.6.1.2.1.2.2.1.17.1 = Counter32: 2950461
.1.3.6.1.2.1.2.2.1.17.2 = Counter32: 81734428
.1.3.6.1.2.1.2.2.1.19.1 = Counter32: 0
.1.3.6.1.2.1.2.2.1.19.2 = Counter32: 0
.1.3.6.1.2.1.2.2.1.20.1 = Counter32: 0
.1.3.6.1.2.1.2.2.1.20.2 = Counter32: 0
.1.3.6.1.2.1.2.2.1.21.1 = Gauge32: 0
.1.3.6.1.2.1.2.2.1.21.2 = Gauge32: 0
.1.3.6.1.2.1.2.2.1.22.1 = OID: SNMPv2-SMI::zeroDotZero
.1.3.6.1.2.1.2.2.1.22.2 = OID: SNMPv2-SMI::zeroDotZero
```


Instance Number

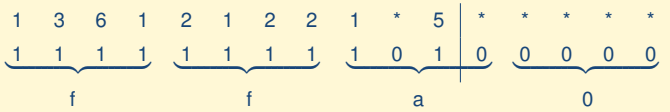
- Notice that the index is the number at the end of the OID
- Called an *instance number*. Index starts from 1
- Suppose we are an ISP, want to allow customer A to view their own network interface, but not that of customer B, their competitor.
- Note that as we go along a row, the OID element just before the instance number changes
- Suppose customer A has a network interface with the index 5.

  ```
  $ snmptranslate -On IF-MIB::ifOutOctets.5
  .1.3.6.1.2.1.2.2.1.16.5
  ```

- So want to allow access for customer A to .1.3.6.1.2.1.2.2.1.*.5

The View Mask — 2

- We can provide a view mask to specify this:



- A *zero in the bit mask* is like a wildcard or “don’t care” specifier
- A mask of all 1’s is the same as a single view subtree specified by the family name (it’s the same as not specifying a mask)
- Here the mask is specified as ff.a0
- For Net-SNMP, the mask is specified as a list of hexadecimal *bytes* separated with ‘.’ or ‘:’.

The View Mask — 3

- Note that in creating a view mask, we start from the left, writing hexadecimal digits.
- We don't care about the bits representing non-existent elements after the end of the subtree parent.
  - I mean the bits to the right of the vertical line in slide §24
  - These bits could be one or zero; I chose zero, since zero means "don't care; you can use any value here"
- We can specify this *family of view subtrees* like this:
  `view custA included interfaces.ifTable.ifEntry.ifIndex.5 ff.a0`
- This view can then be used in an `access` statement
  - see the example in slide §29

The View Mask — 4

- One bit in the view mask determines access to one element in the OID
  - It doesn't matter how big or small the numerical component of the OID is
  - one bit controls whether different values for that component are included in the family of view subtrees or not
- RFC 3415 says that any bit mask is extended with 1's to the same length in bits as the number of identifiers in the OID if it is shorter.
- As a consequence, a family mask of zero length corresponds to a single view subtree.

Net-SNMP VACM Example 1

```
#       sec.name    source          community
com2sec local       localhost       mypP?rC32
com2sec ictnetwork 172.19.64.0/18   public

#     group.name sec.model  sec.name
group MyRWGroup  v1         local
group MyRWGroup  v2c        local
group MyROGroup  v1         ictnetwork
group MyROGroup  v2c        ictnetwork

#    viewname incl/excl subtree
view all      included  .1

#      group.name context sec.model sec.level match  read write notif
access MyROGroup  ""      any       noauth    exact  all  none  none
access MyRWGroup  ""      any       noauth    exact  all  all   none
```


Net-SNMP VACM Example 1

- In the example in §27, read-write access using the community string "mypP?rC32" is allowed from the same machine only (localhost).
- read only access is allowed from any machine in the ICT laboratory subnet using the (badly chosen) community string "public".
- No traps or inform requests can be sent by the agent.

Net-SNMP VACM Example 2

```
group companyA usm companyAManager
group companyB usm companyBManager

view viewA included IF-MIB::ifIndex.5 ff.a0
view viewB included IF-MIB::ifIndex.2 ff.a0

access companyA "" usm priv exact viewA none  none
access companyB "" usm priv exact viewB none  none
```

- `companyAManager` is a USM user that has read-only access to the `ifTable` row that corresponds to the company A's own network interface, and no other access.
- `companyBManager` is a USM user that has read-only access to the `ifTable` row that corresponds to the company B's own network interface, and no other access.

Cisco VACM Configuration

- Cisco IOS specifies a view with the following syntax:
  ```
  snmp-server view viewA ifEntry.*.5 included
  snmp-server view viewB ifEntry.*.2 included
  ```
- Can specify a group with:
  ```
  snmp-server group groupA v3 auth read viewA
  ```
- Cisco uses the `snmp-server user` command to specify users and group membership
- See also pages 284–285 of *Essential SNMP*.

User-based Security Model

- USM allows remote configuration of users
- Securely supports strong authentication using MD5 or SHA1 and encryption using DES
- Remotely create new users by *cloning* existing users
- Can only clone a user once
- Each user must be given access using VACM or that user account cannot be used
  - Add the user to a *group*
  - provide *access* to that group through *views*

Configuring USM Users — 1

- USM users can be created with the `net-snmp-config` program:
- Stop the agent first, then create the initial user:
  ```
  $ sudo service snmpd stop
  $ sudo net-snmp-config --create-snmpv3-user \
    -a "my_password" myuser
  ```
- SNMPv3 pass phrases must be at least 8 characters long.
- We have created a user "`myuser`" with a password of "`my_password`" and using MD5 for authentication and DES for encryption.
- Very simple access control has been added to `/usr/share/snmp/snmpd.conf` allowing the user write access to entire tree

Configuring USM Users — 2

- Now start the agent, and test the user. First we test without encryption, then with encryption:

```
$ sudo service snmpd start
$ snmpget -v 3 -u myuser -l authNoPriv -a MD5 \
-A my_password localhost sysUpTime.0
$ snmpget -v 3 -u myuser -l authPriv   -a MD5 \
-A my_password -x DES -X my_password localhost sysUpTime.0
```

- Can create as many users as you like in this way.
- Better to improve access control using VACM over the default of write access everywhere

Remotely Creating USM Users

- We clone the first user we created:

```
$ snmpusm -v 3 -u myuser -l authNoPriv -a MD5 \
-A my_password localhost create nicku myuser
```

- We now have created user `nicku` with the same password as the "`myuser`" user.
- Now change the password:

```
$ snmpusm -v 3 -u nicku -l authNoPriv -a MD5 \
-A my_password localhost passwd my_password \
new_passphrase
```

  - See `man snmpusm` and `man snmpcmd`
- Can put account information into a local `~/.snmp/snmp.conf` that is readable only by you
  - See `man snmp.conf`

SNMP Standards and RFCs

- The standards were updated in December 2002
  - Most (all?) text books are out of date

RFC 1155	SNMPv1	RFC 3411	SNMPv3 architecture
RFC 1157	SMIv1	RFC 3412	SNMPv3 message processing
RFC 1212	Concise MIB definitions	RFC 3413	SNMPv3 applications
RFC 1215	SNMPv1 traps	RFC 3414	SNMPv3 USM
RFC 1901	SNMPv2c	RFC 3415	SNMPv3 VACM
RFC 2570	Old SNMPv3 overview	RFC 3416	SNMPv2 protocol operations
RFC 2578	SMIv2	RFC 3417	SNMPv2 transport mappings
RFC 2579	SMIv2 textual conventions	RFC 3418	SNMPv2 MIB
RFC 2580	SMIv2 conformance	RFC 3512	SNMP configuring networks info
		RFC 3584	SNMP coexistence v1 v2 v3 best practice

References

- RFCs 3411–3415. Available from many sites, including `http://www.rfc-editor.org`.
- See the Net-SNMP FAQ, in `/usr/share/doc/net-snmp-5.2.1/FAQ`. Also see `/usr/share/doc/net-snmp-5.2.1/README.snmpv3`.
- William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Third edition, Addison-Wesley, 1999, 0-201-48534-6.
  - Pages 526, 527 explain the context example from RFC 2271 well. Actually, the example is changed slightly in RFC 3411
- David Zeltersman, *A Practical Guide to SNMPv3 and Network Management*, Prentice Hall, 1999, 0-13-021453-1.
- Stephen B. Morris, *Network Management, MIBs and MPLs: Principles, Design and Implementation*, Prentice Hall, 2003, 0-13-101113-8.
- James Boney, *Cisco IOS In a Nutshell*, O'Reilly, January 2002, 1-56592-942-X.
- Douglas R. Mauro and Kevin J. Schmidt, *Essential SNMP*, O'Reilly, July 2001, 0-596-00020-0.