Sri Lanka Institute of Information Technology
Year 3 – Semester 1
Network Design and Management

Practical 05 – SNMP

STEP 01----------

1) Stop all the running services.

```
[root@server ~]# service dhcpd stop

[root@server ~]# service named stop
```

Note: Root privilege is needed for the terminal.

2) Set VMware network settings to vmnet8-NAT and make your server IP settings to obtain IP address automatically.

3) Restart the network service

Note: You should have a look at your IP address to check whether the IP is within the network range defined by vmnet8-NAT configured DHCP server.

STEP 02----------

4) Install SNMP using the following command.

```
[root@server ~]# yum install -y net-snmp net-snmp-utils
```

STEP 03----------

5) Set VMware network settings to vmnet2-Host_only and make your server IP settings to obtain IP address manually.

IPADDRESS = 10.0.1.2
NETMASK=255.255.255.0
GATEWAY=10.0.1.1 6)

Restart the network service

Note: You should have a look at your IP address.

STEP 04----------

7) Use the following command to view the SNMP configuration file.

```
[root@server ~]# vi /etc/snmp/snmpd.conf
```

Add below configuration to /etc/snmp/snmpd.conf

```
rocommunity  public  xxx.xxx.xxx.xxx
rocommunity  public   127.0.0.1 syslocation
"HYD, UM DataCenter" syscontact
admin@ndm.lk
```

Replace xxx.xxx.xxx.xxx with the IP address of the server that you want to allow SNMP lookups from:

```
rocommunity public xxx.xxx.xxx.xxx
```

8) Start SNMP service using the following command

```
[root@server ~]# service snmpd start
```

STEP 05----------

9) Use the following commands to view the MIB-II and to retrieve information using SNMP get requests.

I. **snmptranslate** : learning about the MIB tree. (translate MIB OID names between numeric and textual forms)

The **snmptranslate** tool is a very powerful tool that allows you to browse the MIB tree in various ways from the command line.

In its simplest form, it merely looks up an OID and spits it back out in textual form:

```
% snmptranslate .1.3.6.1.2.1.1.3.0 SNMPv2-MIB::sysUpTime.0
```

It can also translate into numerical results as well, by adding the -On flag to its options

```
% snmptranslate -On SNMPv2-MIB::system.sysUpTime.0
.1.3.6.1.2.1.1.3.0
```

Note that the argument passed can describe a OID in any fashion, and the -On flag merely toggles which type of output is displayed:

```
% snmptranslate .iso.3.6.1.private.enterprises.2021.2.1.prNames.0
NET-SNMP-MIB::prNames.0
% snmptranslate -On .iso.3.6.1.private.enterprises.2021.2.1.prNames.0
.1.3.6.1.4.1.2021.2.1.2.0
```

Note how the oid was abbreviated for you? You can change this behaviour as well with -Of:

```
% snmptranslate -Of .iso.3.6.1.private.enterprises.2021.2.1.prNames.0
.iso.org.dod.internet.private.enterprises.ucdavis.procTable.prEntry.p
rNames.0
```

The problem with the above commands is that you have to remember the entire OID for what you're looking for. Now, we wouldn't be a very good package if we didn't provide some sort of random-access lookup function right? Well, the good news is that we do. *-IR* nicely does this for us, and searches the MIB tree for the node you want:

```
% snmptranslate sysUpTime.0
Invalid object identifier: sysUpTime.0
% snmptranslate -IR sysUpTime.0 SNMPv2-MIB::sysUpTime.0
```

Finally, it'll even try to do regex pattern matching to find the exact node you want given only a piece of its name by using the -Ib (best match) option:

```
% snmptranslate -Ib 'sys.*ime' system.sysUpTime
```

To get a list of all the nodes that match a given pattern, use the -TB flag:

```
% snmptranslate -TB 'vacm.*table'
SNMP-VIEW-BASED-ACM-MIB::vacmViewTreeFamilyTable
SNMP-VIEW-BASED-ACM-MIB::vacmAccessTable
SNMP-VIEW-BASED-ACM-MIB::vacmSecurityToGroupTable SNMP-VIEW-BASED-
ACM-MIB::vacmContextTable
```

To get extended information about a mib node, use the -Td (description) flag:

```
% snmptranslate -On -Td -Ib 'sys.*ime'
.1.3.6.1.2.1.1.3 sysUpTime
OBJECT-TYPE
  -- FROM SNMPv2-MIB, RFC1213-MIB
  SYNTAX TimeTicks
  MAX-ACCESS       read-only
  STATUS current
  DESCRIPTION "The time (in hundredths of a second) since the network
           management portion of the system was last reinitialized."
::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1) 3 }
```

Finally, last but certainly not least, if you want a pretty diagram of a section of the mib tree, check out the -Tp flag:

```
% snmptranslate -Tp -IR system
```

```
+--system(1)
   |
   +-- -R-- String    sysDescr(1)
   |        Textual Convention: DisplayString
   +-- -R-- ObjID     sysObjectID(2)
   +-- -R-- TimeTicks sysUpTime(3)
   +-- -RW- String    sysContact(4)
   |        Textual Convention: DisplayString
   +-- -RW- String    sysName(5)
   |        Textual Convention: DisplayString
+-- -RW- String    sysLocation(6)
   |        Textual Convention: DisplayString
   +-- -R-- Integer   sysServices(7)
   +-- -R-- TimeTicks sysORLastChange(8)
|         Textual Convention: TimeStamp
   |
   +--sysORTable(9)
      |
      +--sysOREntry(1)
         |
         +-- ---- Integer   sysORIndex(1)
         +-- -R-- ObjID     sysORID(2)
         +-- -R-- String    sysORDescr(3)
         |        Textual Convention: DisplayString
         +-- -R-- TimeTicks sysORUpTime(4)
                  Textual Convention: TimeStamp
```

As a homework assignment, we'll leave it up to you to run *snmptranslate -Tp* without an oid argument, which prints the known MIB tree in its entirety.

Note: You are required to look into the manual pages to find the definitions of the switches used.

II. **snmpget** : retrieving data from a host.

The **snmpget** command can be used to retrieve data from a remote host given its host name, authentication information and an OID. As a simple example:

```
% snmpget -v 1 -c demopublic test.net-snmp.org system.sysUpTime.0
system.sysUpTime.0 = Timeticks: (586731977) 67 days, 21:48:39.77
```

In the above example, *test.net-snmp.org* is the host name we wanted to talk to, using the SNMP community string *demopublic* and we requested the value of the OID *system.sysUpTime.0*.

Earlier versions of the ucd-snmp utilities used SNMPv1 by default and expected the community name to follow the host name. The net-snmp versions of these tools now typically use SNMPv3 by default, and require both the version **and** the community string to be given as command line options (as illustrated in these examples). SNMPv2c, which is similar in nature to SNMPv1 with small modifications, still used clear-text community names as "passwords" to authenticate the issuer of the

command. The result from a command using the SNMPv2c version would have been the same:

```
% snmpget -v 2c -c demopublic test.net-snmp.org system.sysUpTime.0
system.sysUpTime.0 = Timeticks: (586752671) 67 days, 21:52:06.71
```

All of the utilities allow abbreviation of the OIDs and do random searches by default, and hence you can only specify a small portion of the oid if you would prefer:

```
% snmpget -v 2c -c demopublic test.net-snmp.org sysUpTime.0
system.sysUpTime.0 = Timeticks: (586752671) 67 days, 21:52:06.71
```

A common mistake when using the snmpget command is to leave off the index into the data you're looking for. In the above commands, the variable requested by the OID is a scalar and the index to scalars is always a simple '0' (zero), hence the trailing '.0' in all the oids above. If you had left it off, you would have gotten an error. Note that the errrors differ slightly between SNMPv1 and SNMPv2c:

```
% snmpget -v 1 -c demopublic test.net-snmp.org sysUpTime
Error in packet
Reason: (noSuchName) There is no such variable name in this MIB.
This name doesn't exist: system.sysUpTime
% snmpget -v 2c -c demopublic test.net-snmp.org sysUpTime
system.sysUpTime = No Such Instance currently exists
```

Multiple variables can be retrieved in one transaction as well:

```
% snmpget -v 2c -c demopublic test.net-snmp.org sysUpTime.0
ucdDemoUserList.0
system.sysUpTime.0 = Timeticks: (586903243) 67 days, 22:17:12.43
enterprises.ucdavis.ucdDemoMIB.ucdDemoMIBObjects.ucdDemoPublic.ucdDem
oUserList.0 = " noAuthUser MD5User MD5DESUser SHAUser SHADESUser"
```

Note: You are required to look into the manual pages to find the definitions of the switches used.

III. **snmpgetnext** : retrieving unknown indexed data.

The **snmpgetnext** command, which is similar in usage to the snmpget command, is used to retrieve the **next** oid in the mib tree of data. Instead of returing the data you requested, it returns the **next** OID in the tree and its value:

```
% snmpgetnext -v 2c -c demopublic test.net-snmp.org
system.sysUpTime.0
system.sysContact.0 = Wes Hardaker wjhardaker@ucdavis.edu
```

You could use the snmpgetnext command to manually walk down the mib tree in the remote host, by always specifying the last OID that you saw on the command line for the next command:

```
% snmpgetnext -v 2c -c demopublic test.net-snmp.org
system.sysUpTime.0 system.sysContact.0 = Wes Hardaker
wjhardaker@ucdavis.edu % snmpgetnext -v 2c -c demopublic
test.net-snmp.org system.sysContact.0 system.sysName.0 =
net-snmp
% snmpgetnext -v 2c -c demopublic test.net-snmp.org system.sysName.0
system.sysLocation.0 = UCDavis
```

In fact, the snmpwalk command described in the next section, implements exactly this but in one command!

Unlike the snmpget command, the snmpgetnext command does return data for a OID which is too short or is missing the index part of the OID. For instance, if you remember from the last snmpget discussion, if you left off the .0 on the end of the OID you were requesting on a snmpget command, you were issued an error. With snmpgetnext, you're still issued an answer, because you will always get the next value in the tree, regardless of whether or not you specified a valid OID for a variable or not:

```
% snmpgetnext -v 2c -c demopublic test.net-snmp.org system.sysUpTime
system.sysUpTime.0 = Timeticks: (586978184) 67 days, 22:29:41.84 %
snmpgetnext -v 2c -c demopublic test.net-snmp.org system system.sysDescr.0
= HP-UX net-snmp B.10.20 A 9000/715 % snmpgetnext -v 2c -c demopublic
test.net-snmp.org .1.3.6 system.sysDescr.0 = HP-UX net-snmp B.10.20 A
9000/715
```

Note: You are required to look into the manual pages to find the definitions of the switches used.

STEP 08 – DO THIS BY YOUR SELF----------

10) Tryout *snmpwalk, snmptable, snmpset, snmptrap* functions by yourself.