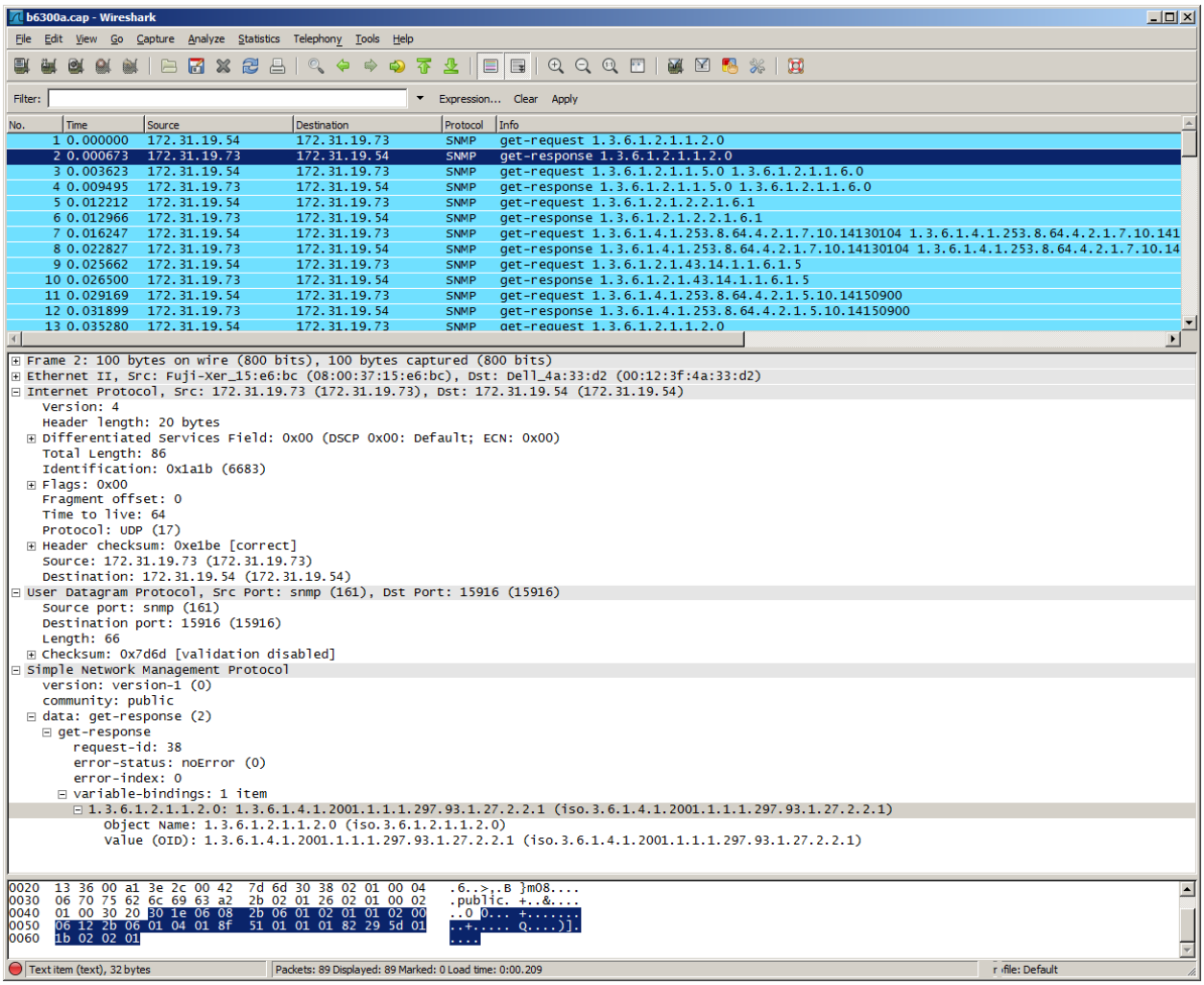
CS 75 – SNMP Lab
March 29, 2014s

1. Analyzing snmp packets

Download the SNMP packet capture at http://wiki.wireshark.org/SampleCaptures#SNMP . The file is named b6300a.cap. it is also provided on the class Web site.

Open the file with Wireshark

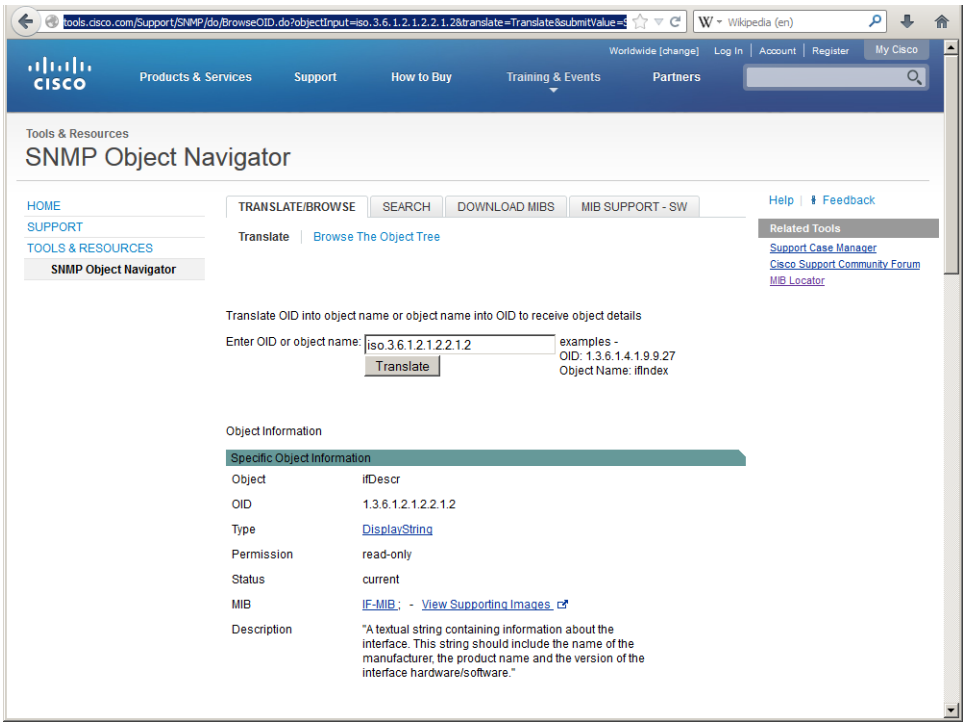


Note that packets 1 through 52 are a series of snmp get-request, get-response, and set-request queries.

a. Which of these are sent by the manager? Which by the agent?

b. What is the IP address of the manager? The address of the agent?

c. Look at one of the get-request packets. Note under that variable bindings the OID for the request is specified as a dotted numeric format (an OID or Object Identifier). There is a Web site you can use to translate the numeric OID values into text:http://www.cisco.com/go/mibs.



See if you can find out which value is being requested. NOTE: if the last numeric value is zero, try deleting this trailing zero when looking up the OID.

d. Look at the next packet, showing the get-response. Look up the OID at http://www.cisco.com/go/mibs. What value is being returned?

2. Gathering router information through SNMP commands

The test router (a Cisco 3640) has been configured as follows:

IP address **192.168.9.88**

SNMP v1 community string **public**

Install the SNMP utilities on your class laptop by typing:

```
yum install net-snmp-utils
```

Note that this may take a while.

Verify that the snmpwalk utility installed properly by typing:

```
snmpwalk –help
```

The output will look like this:

```
vleveque@lucky:~$ snmpwalk -h
USAGE: snmpwalk [OPTIONS] AGENT [OID]
```

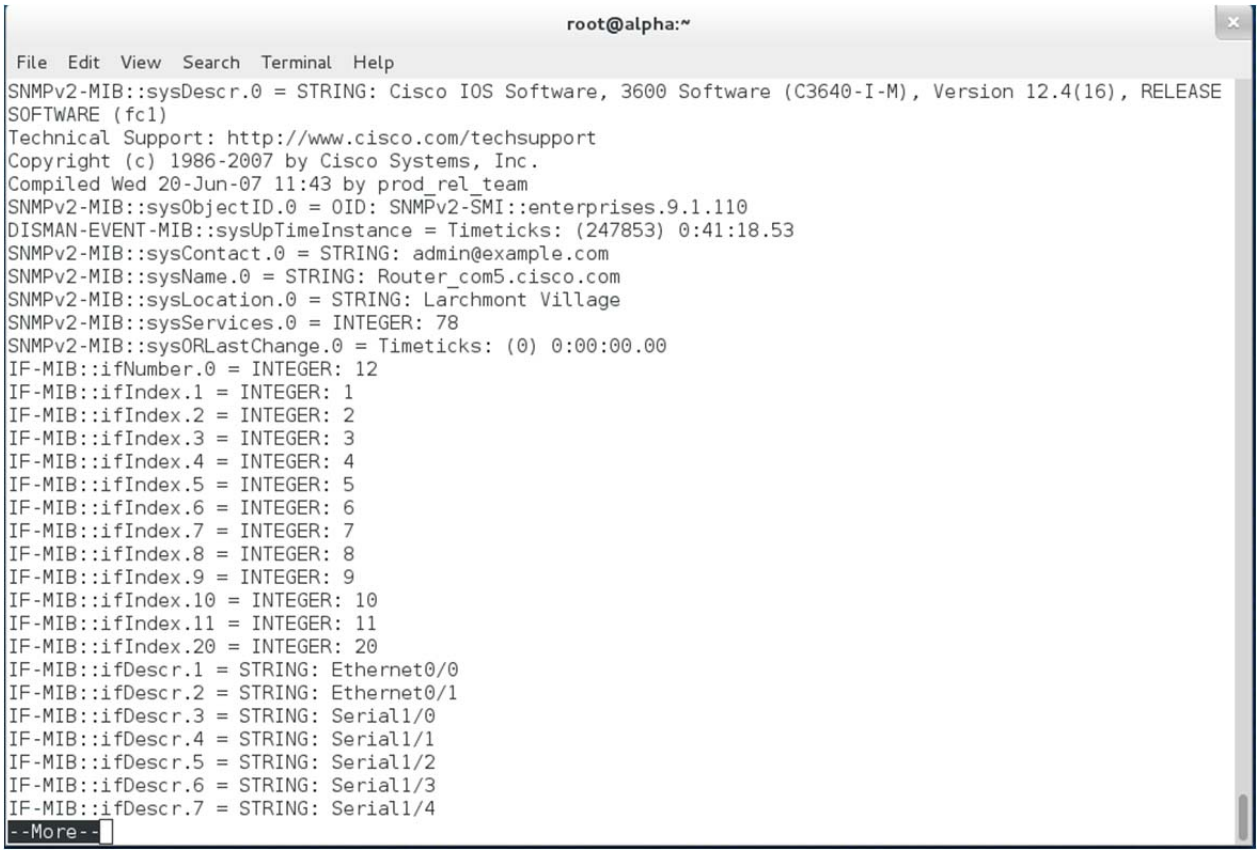
```
  Version:  5.4.3
  Web:      http://www.net-snmp.org/
  Email:    net-snmp-coders@lists.sourceforge.net

OPTIONS:
  -h, --help            display this help message
  -H                    display configuration file directives understood
  -v 1|2c|3             specifies SNMP version to use
  -V, --version         display package version number
SNMP Version 1 or 2c specific
  -c COMMUNITY          set the community string
SNMP Version 3 specific
  -a PROTOCOL           set authentication protocol (MD5|SHA)
  -A PASSPHRASE         set authentication protocol pass phrase
  -e ENGINE-ID          set security engine ID (e.g. 800000020109840301)
  -E ENGINE-ID          set context engine ID (e.g. 800000020109840301)
  -l LEVEL              set security level (noAuthNoPriv|authNoPriv|authPriv)
  -n CONTEXT            set context name (e.g. bridge1)
  -u USER-NAME          set security name (e.g. bert)
  -x PROTOCOL           set privacy protocol (DES|AES)
  -X PASSPHRASE         set privacy protocol pass phrase
  -Z BOOTS,TIME         set destination engine boots/time
General communication options
  -r RETRIES            set the number of retries
  -t TIMEOUT            set the request timeout (in seconds)
Debugging
  -d                    dump input/output packets in hexadecimal
  -D TOKEN[,...]        turn on debugging output for the specified TOKENs
                           (ALL gives extremely verbose debugging output)
General options
  -m MIB[:...]          load given list of MIBs (ALL loads everything)
  -M DIR[:...]          look in given list of directories for MIBs
```

and much more…

A simple use of snmpwalk would involve dumping all the information available from the router, using the more command to present one screen at a time:

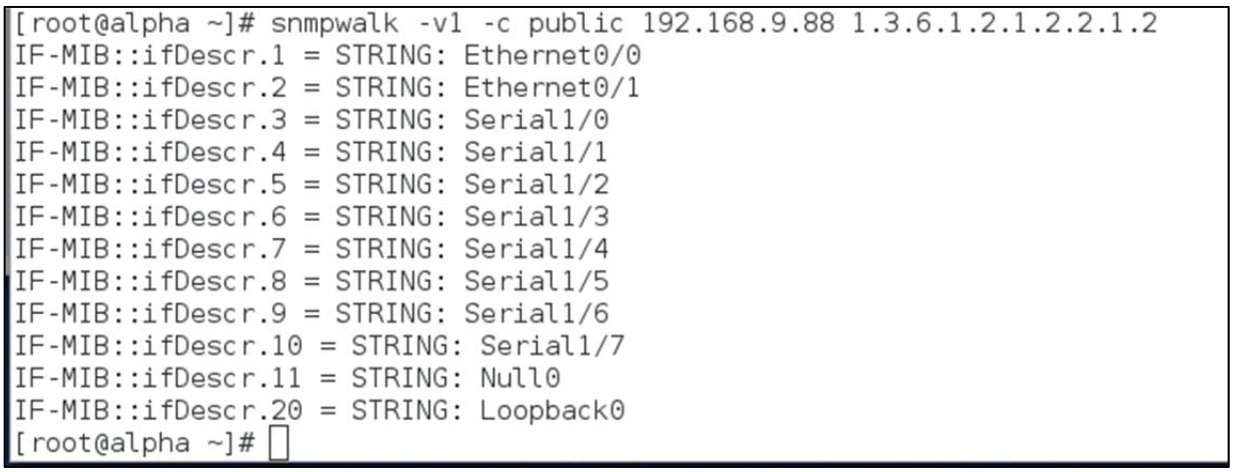
```
snmpwalk –v1 –c public 192.168.9.88 | more
```



Try this command

```
snmpwalk –v1 –c public 192.168.9.88 1.3.6.1.2.1.2.2.1.2
```

What sort of information is snmp returning?



SNMP returns a table of values, one for each interface on this device. If you queried other devices, you would get a similar table for the interfaces supported on that device.

Now try the following:

```
snmpwalk -v1 -c public 192.168.9.88 1.3.6.1.2.1.2.2.1.10
```

These will give you the total number of octets (8 bit groupings) coming into the interface:

```
[root@alpha ~]# snmpwalk -v1 -c public 192.168.9.88 1.3.6.1.2.1.2.2.1.10
IF-MIB::ifInOctets.1 = Counter32: 0
IF-MIB::ifInOctets.2 = Counter32: 401399
IF-MIB::ifInOctets.3 = Counter32: 0
IF-MIB::ifInOctets.4 = Counter32: 0
IF-MIB::ifInOctets.5 = Counter32: 0
IF-MIB::ifInOctets.6 = Counter32: 0
IF-MIB::ifInOctets.7 = Counter32: 0
IF-MIB::ifInOctets.8 = Counter32: 0
IF-MIB::ifInOctets.9 = Counter32: 0
IF-MIB::ifInOctets.10 = Counter32: 0
IF-MIB::ifInOctets.11 = Counter32: 0
IF-MIB::ifInOctets.20 = Counter32: 0
[root@alpha ~]#
```

The following will give you the octets exiting the interface

```
snmpwalk -v1 -c public 192.168.9.88 1.3.6.1.2.1.2.2.1.16
```

If you capture this now, then capture again 5 minutes later, you will be able to subtract the two values and fin the traffic in/out of each interface for that 5 minute period. This is exactly how programs like MRTG and Cacti generate interface traffic graphs. The octets in/out is a type of metric called a **counter**. A counter increments over time, is cumulative from the start of monitoring (with the exception of when it overflows). A metric that show an inherently instantaneous value is a **gauge**. A real life example of a gauge is a thermometer.

You can also query other layers of the protocol stack. For example, to find the IP addresses of each router interface:

```
snmpwalk -v1 -c public 192.168.9.88 1.3.6.1.2.1.4.20.1.1
```

To find the Layer 2 MAC addresses of each router interface:

```
snmpwalk -v1 -c public 192.168.9.88 1.3.6.1.2.1.2.2.1.6
```

```
[root@alpha ~]# snmpwalk -v1 -c public 192.168.9.88 1.3.6.1.2.1.2.2.1.6
IF-MIB::ifPhysAddress.1 = STRING: 0:50:73:5d:df:c1
IF-MIB::ifPhysAddress.2 = STRING: 0:50:73:5d:df:c2
IF-MIB::ifPhysAddress.3 = STRING:
IF-MIB::ifPhysAddress.4 = STRING:
IF-MIB::ifPhysAddress.5 = STRING:
IF-MIB::ifPhysAddress.6 = STRING:
IF-MIB::ifPhysAddress.7 = STRING:
IF-MIB::ifPhysAddress.8 = STRING:
IF-MIB::ifPhysAddress.9 = STRING:
IF-MIB::ifPhysAddress.10 = STRING:
IF-MIB::ifPhysAddress.11 = STRING:
IF-MIB::ifPhysAddress.20 = STRING:
```