TLEN 5410 – Network Management and Automation

Lab 1

Network Management using SNMP and NMAP

University of Colorado Boulder

Interdisciplinary Telecom Program

Professor Levi Perigo, Ph.D.

# Summary

SNMP is used widely by network and system administrators to monitor the health and metrics of a diverse array of network devices.

The objectives in this lab will enable you to understand how different SNMP versions work, gather operational statistics and monitor your network using simple commands, and modify parameters remotely on SNMP agents.

# Pre-Lab

Here is the link for some important Cisco SNMP commands: [SNMP commands](http://www.cisco.com/c/en/us/td/docs/ios/12_2/configfun/command/reference/ffun_r/frf014.html)

You will need the following commands to enable SNMP on the Cisco router in the VM's GNS3. (Note: Use the instructions from Lab 0 for gaining access to the VM and GNS3 setup.)

- Add another interface on the router by clicking on the router. (Double-Click on the **router > Slots > Slot1 > NM-1FE-TX**)  
  
- Run the simulation by clicking on the Play button in GNS3.  
  
- Console into the router, check if SNMP is running using **show snmp host**.

If SNMP is not enabled, follow these steps to configure SNMP host on a Cisco router:  
  
- Enable SNMP traps on the router by entering: (config)#**snmp-server enable traps**   
  
- Assign an IP address (make sure it is in a different subnet than the primary interface, use any private subnet) to the 2nd interface of the router that you added & bring the interface up.  
- Enter configuration commands, one per line. End with CNTL/Z.

(config)# **snmp-server host 198.51.100.2 public**

(config)# **snmp-server community public rw**

\*Note: The "snmp-server host" IP address is the IP address of the VM terminal. Thus, in this example the IP address would be 198.51.100.2.

On the terminal of the VM, found at the bottom of the VM, start Wireshark and monitor the tap0 interface.

Next type the below commands in the VM terminal and check the output (you can receive SNMP data from the router using **SNMPGET/SNMPWALK**).

netman@netman:~$ snmpget -v 1 -c public 198.51.100.3 ifName.1

IF-MIB::ifName.1 = STRING: Fa0/0 ------( This is the output )

netman@netman:~$ snmpget -v 1 -c public 198.51.100.3 .1.3.6.1.2.1.2.1.0

IF-MIB::ifNumber.0 = INTEGER: 5 ------( This is the output )

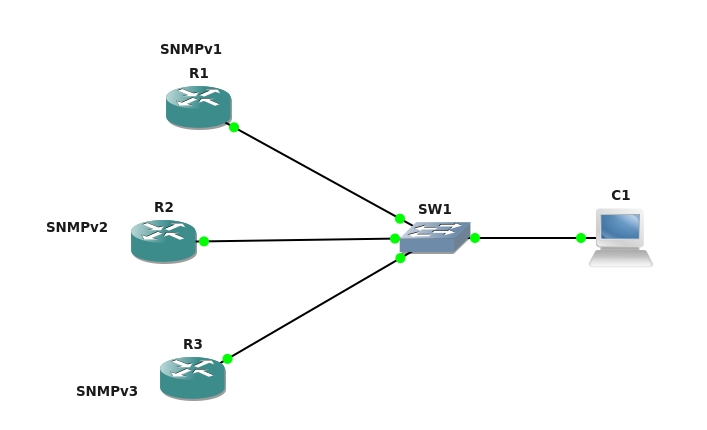
\*NOTE: The IP address used within the terminal is the IP address of the Cisco router. In this example the Cisco router has the IP address of 198.51.100.3.

You should be able to see a similar output on the terminal as well as an SNMP packet on Wireshark.

# Objective 1: Configuring SNMP on Cisco IOS

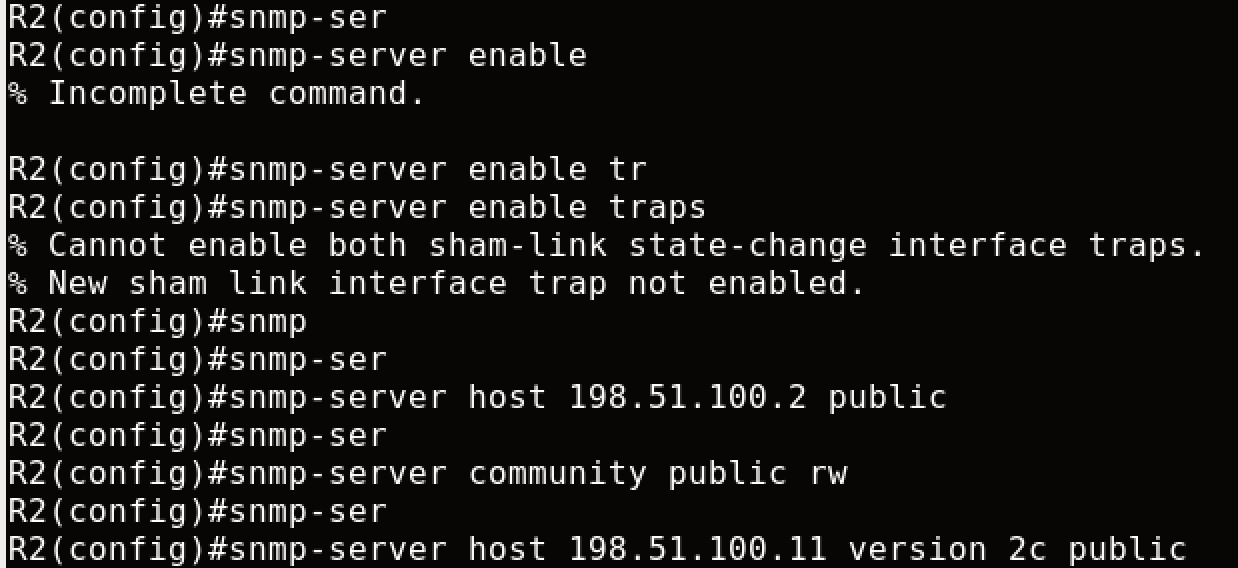
Create the topology in GNS3 as shown below and assign management IPs (198.51.100.0/24 subnet) to them on fa0/0. Configure the nodes for different versions of SNMP & enable traps.

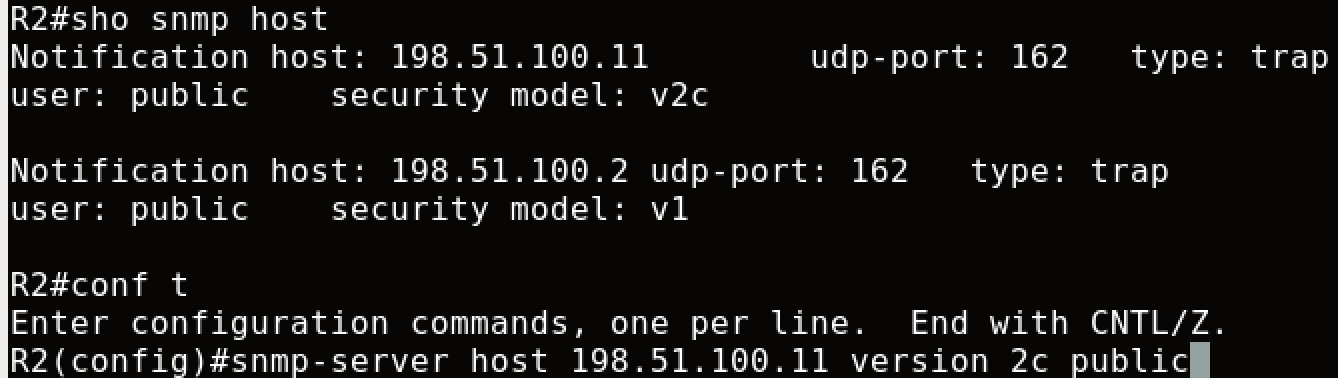
- R1: SNMPv1 (Already configured)  
- R2: SNMPv2   
- R3: SNMPv3



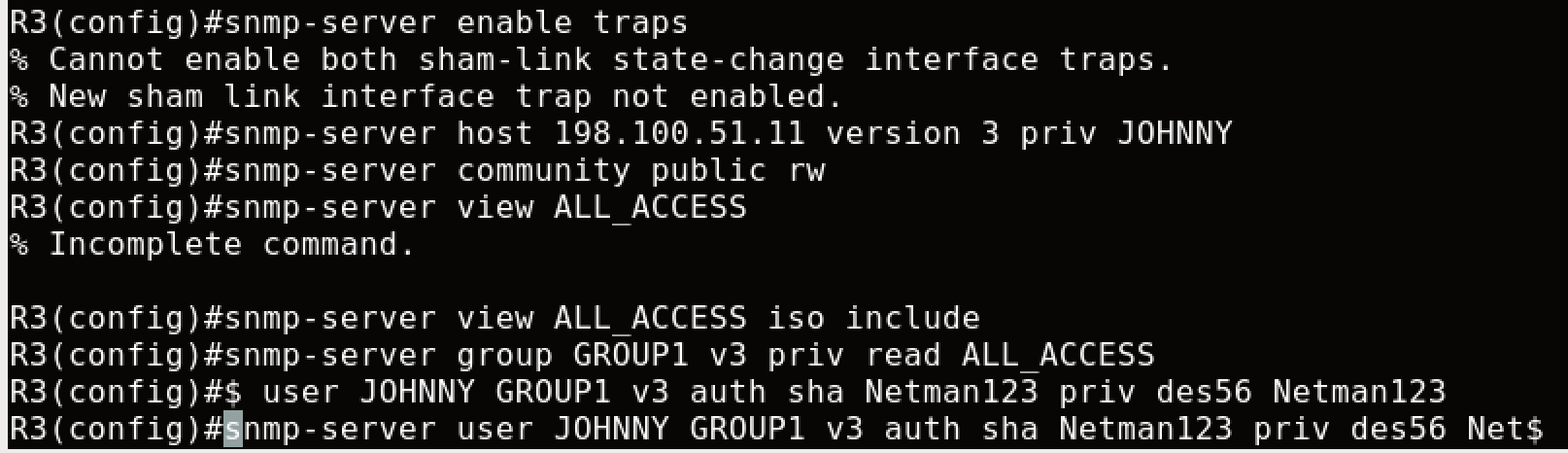
1. How did you configure SNMPv2 and v3 on routers R2 and R3? Provide running configuration screenshots (only portions relevant to SNMP). [**10 points**]

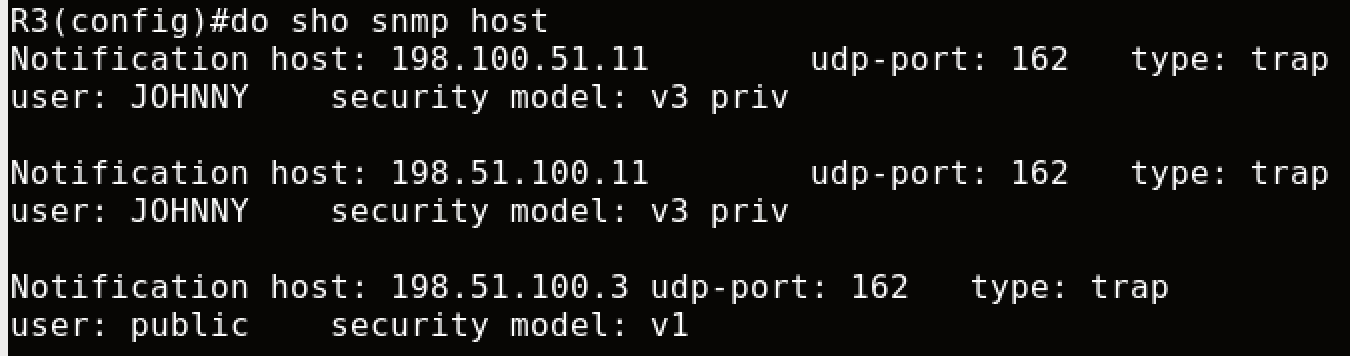
V2:





V3:





# Objective 2: SNMPGET and Dashboard

The list of OIDs that need to be fetched from the routers:

sysContact = 1.3.6.1.2.1.1.4.0

sysName = 1.3.6.1.2.1.1.5.0

sysLocation = 1.3.6.1.2.1.1.6.0

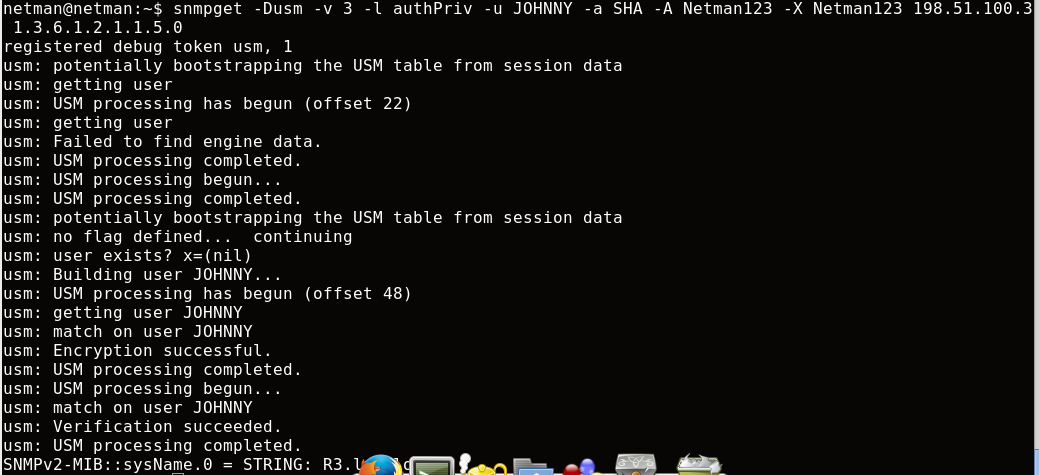
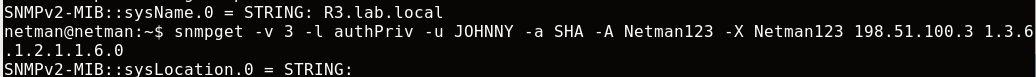
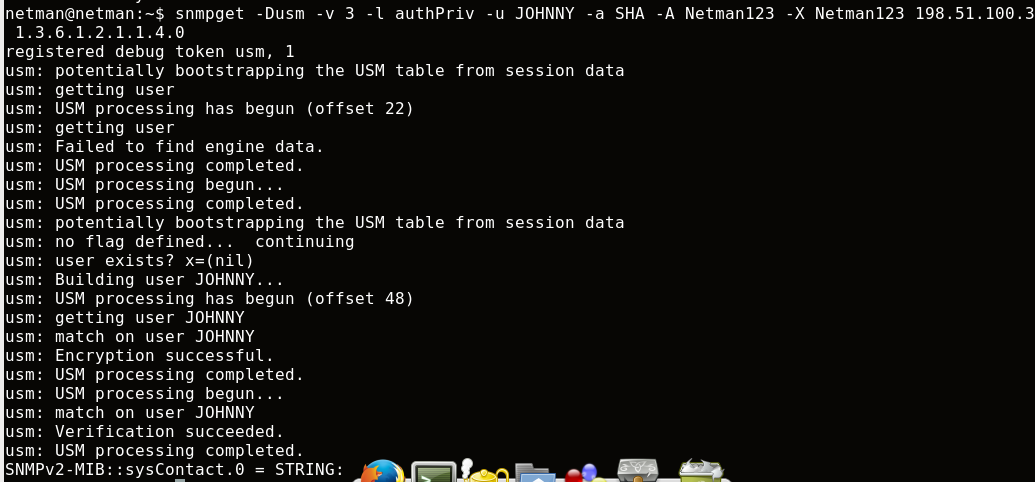
ifNumber = 1.3.6.1.2.1.2.1.0

sysUptime = 1.3.6.1.2.1.1.3.0

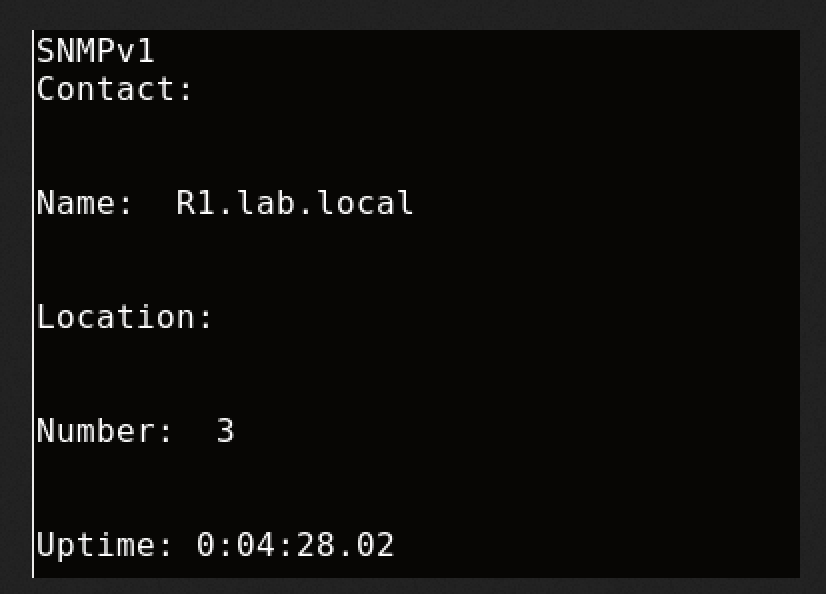
Sample command to run on terminal:

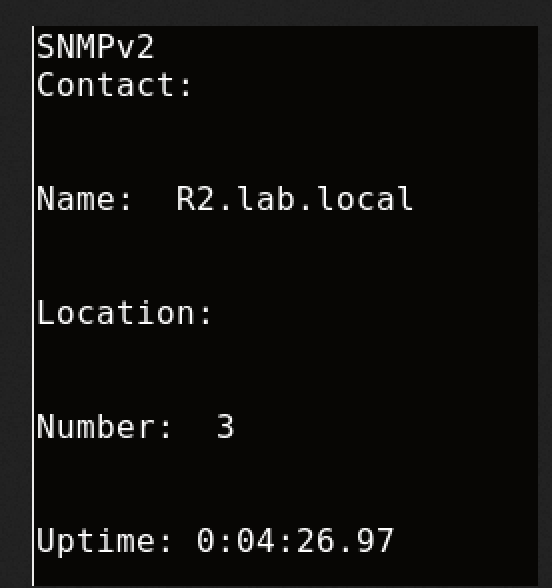
**snmpget -v 1 -c public 198.51.100.3 .1.3.6.1.2.1.1.4.0**

1. Enter the above SNMPGET commands for the OIDs mentioned for SNMP v1, v2, and v3. [**10 points**]



1. Create a dashboard to display the output from those commands using UNIX/Python. [**15 points**]







1. Use SNMPSET commands to modify Contact, Name, and Location to display varied output for each version: 1 and 2. [**10 points**]

**﻿snmpset -v 1 -c public 198.51.100.1 sysContact.0 = "Student Assistant"**

**﻿snmpset -v 1 -c public 198.51.100.1 sysName.0 = "Josh"**

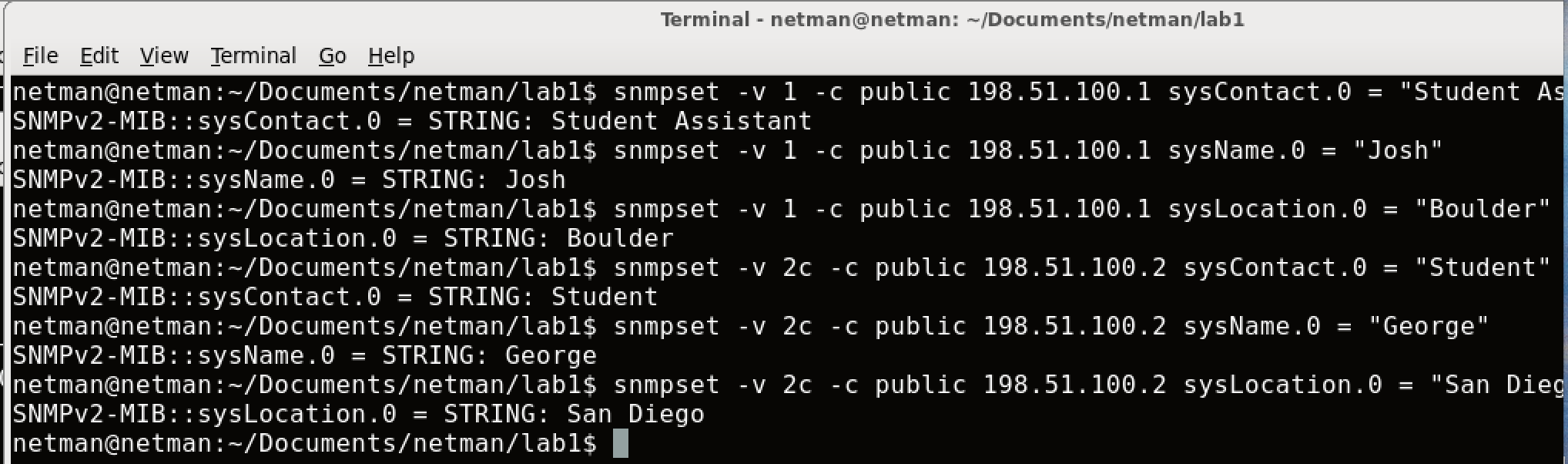
**﻿snmpset -v 1 -c public 198.51.100.1 sysLocation.0 = "Boulder"**

**﻿snmpset -v 2c -c public 198.51.100.2 sysContact.0 = "Student"**

**﻿snmpset -v 2c -c public 198.51.100.2 sysName.0 = "George"**

**﻿snmpset -v 2c -c public 198.51.100.2 sysLocation.0 = "San Diego"**

**﻿**

****

**Sample dashboard to be displayed using UNIX/Python:**

**SNMP v1**

Contact: Student Assistant

Name: Josh

Location: Boulder

Number: 2

Uptime: 0:54:20.47

**SNMP v2**

Contact: Student

Name: George

Location: San Diego

Number: 2

Uptime: 0:67:10.57

**SNMP v3 (any of the 2)**

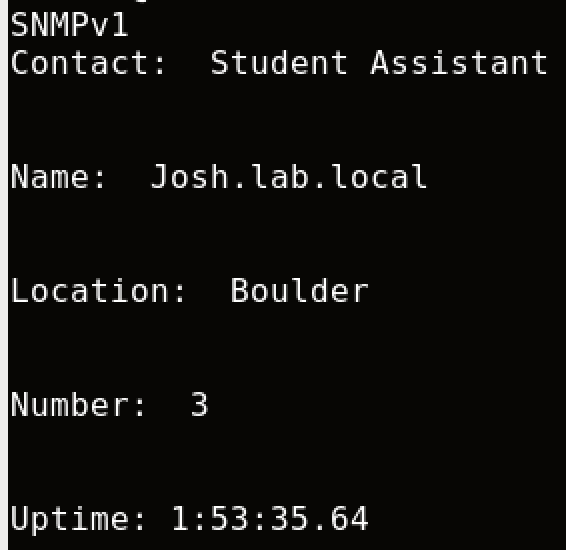
Contact: Professor

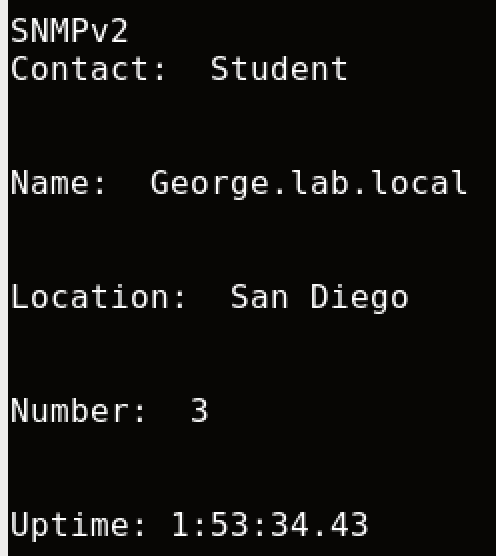
Name: Kelly

Location: Dallas

Number: 2

Uptime: 1:24:20.47





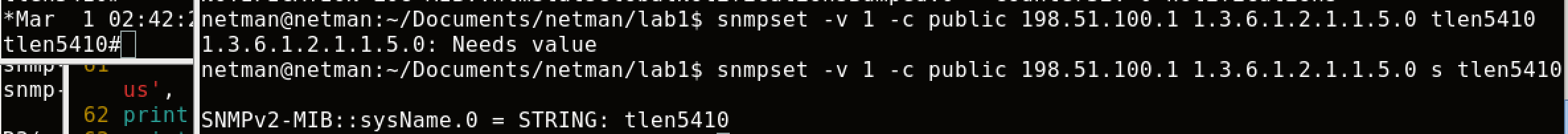


# Objective 3: SNMPSET Commands

**NOTE:** Must use SNMPSET commands to perform the below tasks on Router 1 in GNS3:

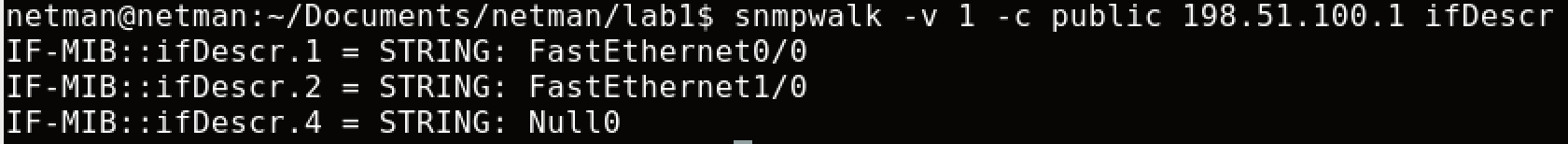
1. Change the hostname to “**tlen5410**” (provide a screenshot) [**10 points**]

Hostname = sysName

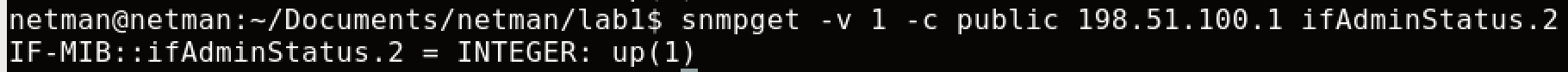


1. Change the interface status of the secondary interface (NOT THE MANAGEMENT INTERFACE) to “**Up**” (Assuming it’s up, if not change to “**Admin Down**”). Provide screenshots. [**10 points**]

First of all, I use the snmpwalk command to find the index of interfaces[3] on R1.



Then, I saw that ifDesct.2 is my interface. Thus, I appended it to the oid name of isAdminStatus.

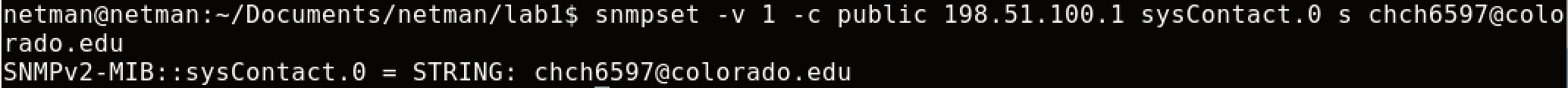


Then, set it to down status  


You can see the tlen5410’s interface f1/0 is down



1. Create a SNMP contact profile with the name (provide a screenshot): <**yourname@colorado.edu**> [**10 points**]

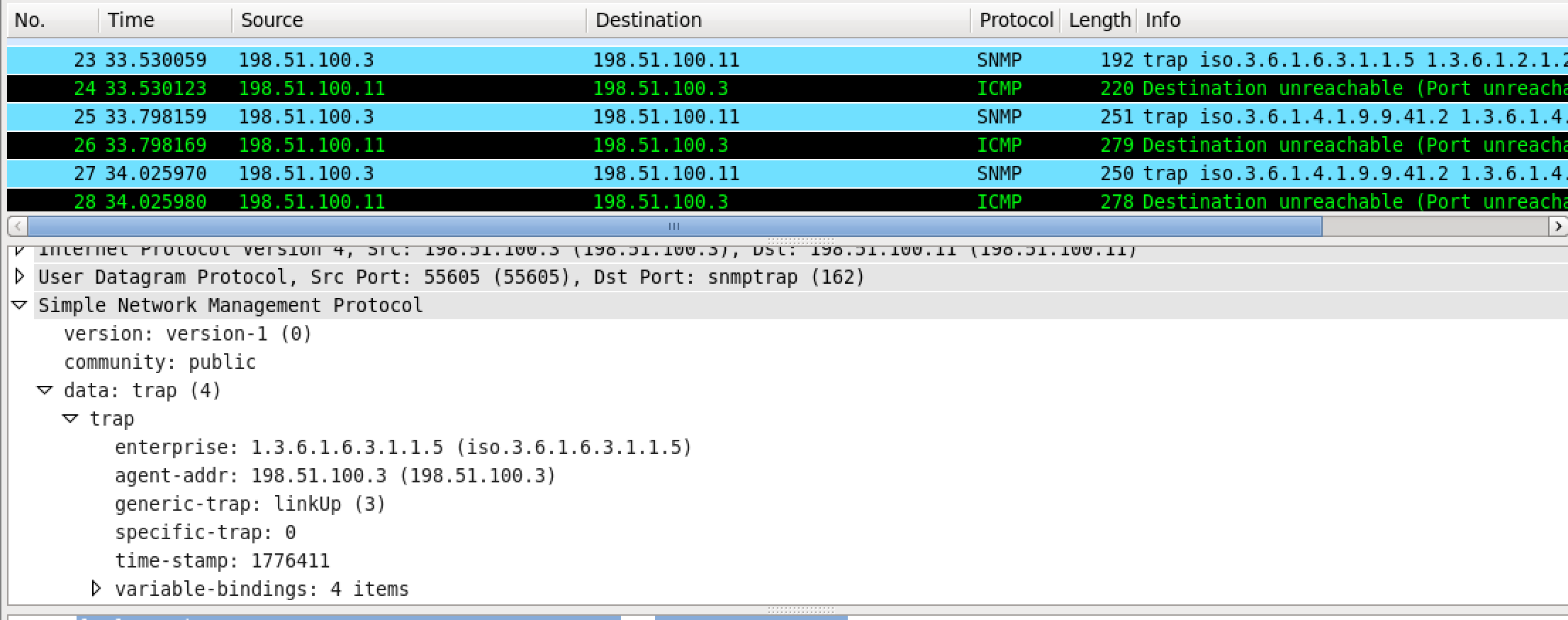


Objective 4: SNMP Traps and Wireshark/TCPDUMP

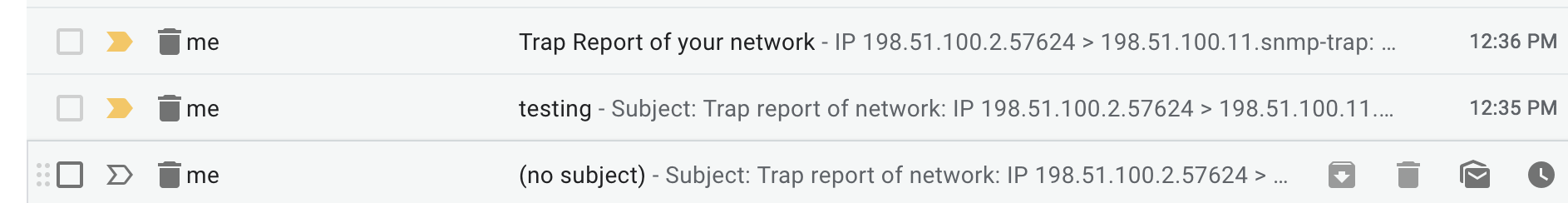
1. Start a new Wireshark capture on the tap0 interface of the VM. Apply a display filter to filter SNMP traffic.
2. Shutdown the interfaces on R2 and R3, and bring them up again. Do you observe different trap messages being exchanged between the SNMP agent and the manager (VM) in the packet capture? Provide relevant screenshots. [**10 points**]

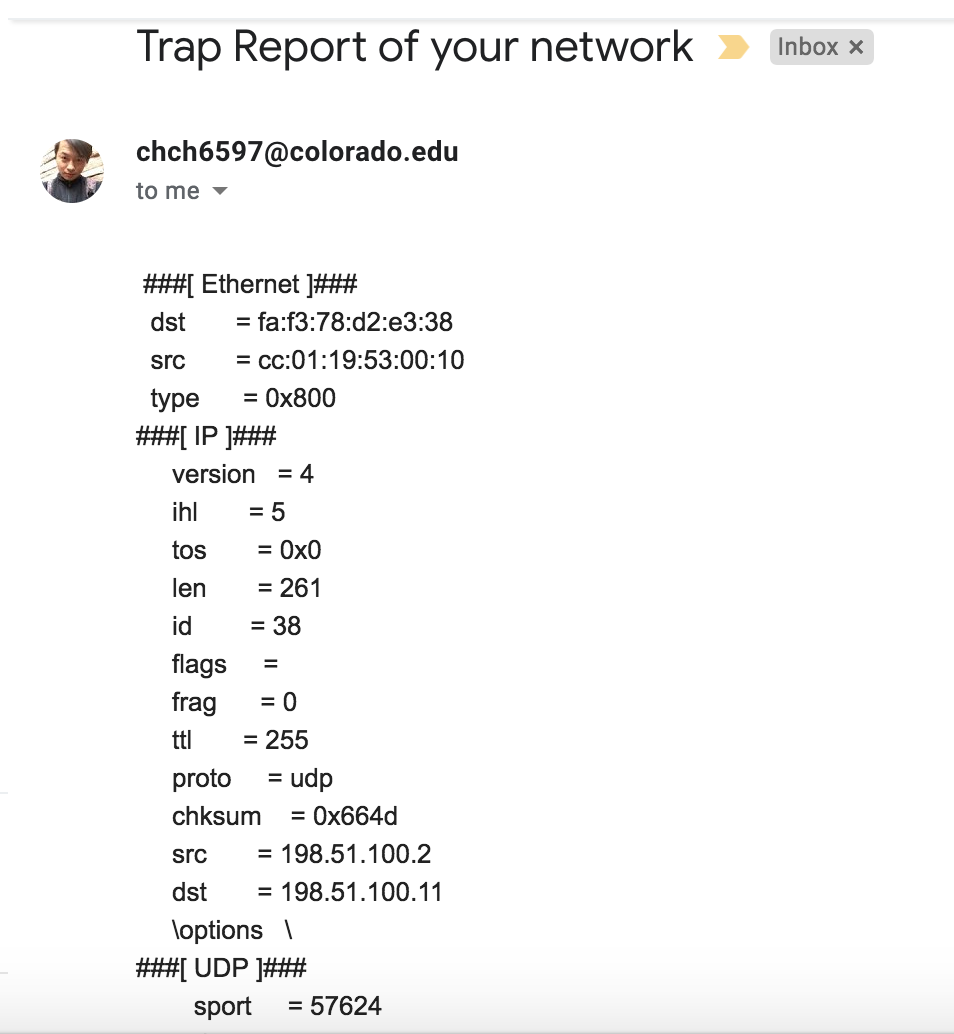
Yes, as my manager IP address is 198.51.100.11 and the agent IP address is 198.51.100.2, they are exchanging the packet in SNMP for v2 and v3.



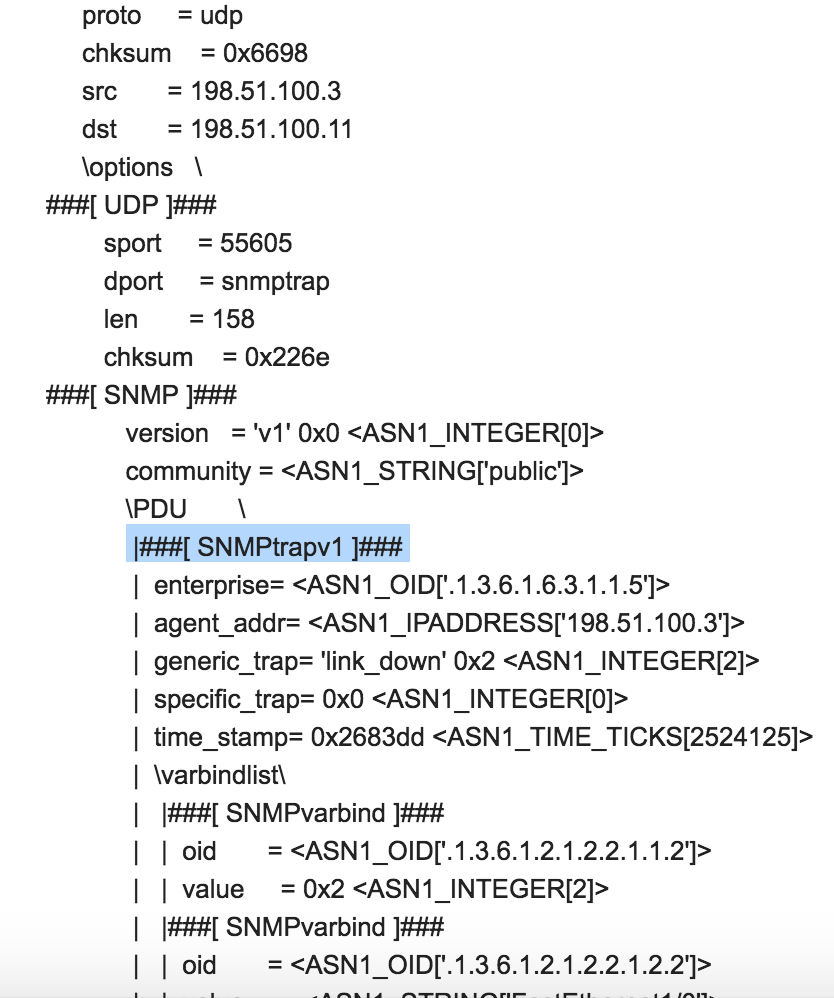
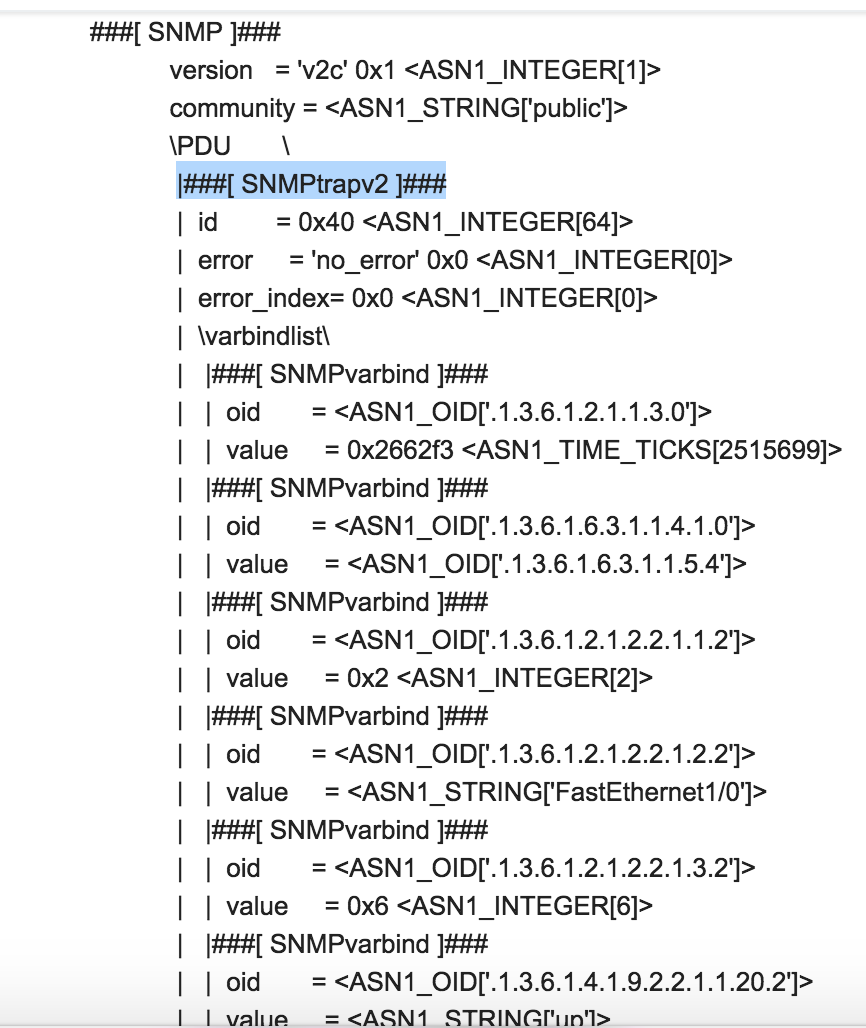


1. Start a capture using TCPDUMP. Bring down an interface on any of the routers (this should generate a trap). Store the output in a .pcap file. After stopping the TCPDUMP, create a Python script that will analyze and parse the .pcap file for a Trap. Then the Python script should generate an email, to your email id, with the contents of the Trap [<https://www.pythonforbeginners.com/google/sending-emails-using-google>]. Provide relevant screenshots and submit the code. [**20 points**]





1. What are the key differences you can observe between the trap messages for SNMPv2 and v3? Provide relevant screenshots highlighting the differences. [**10 Points**]



For SNMPv2, it is clear showing that it is SNMPtrapv2, but in version 3, it is showing SNMPtrapv1

Objective 5: Network Administration using SNMP

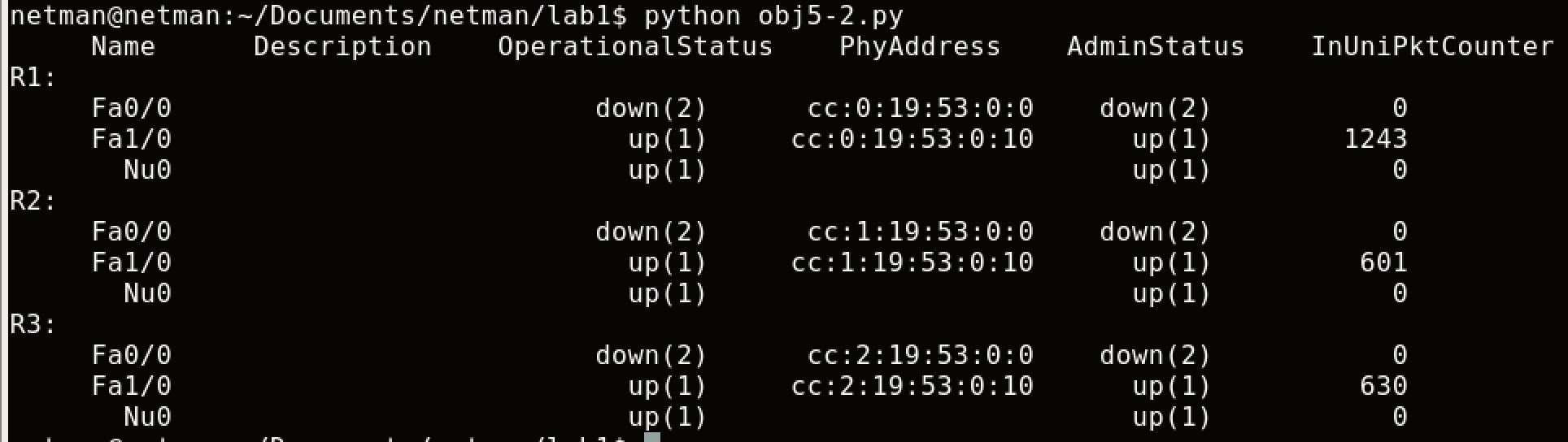
Imagine a Data Center or Service Provider network. You, being a principle network engineer, get a ticket for eBGP sessions going down on multiple routers. You start analyzing the output of all the possible “show” commands in BGP that you are aware of. However, all configurations and parameters look perfect and you scratch your head for a while trying to know the root cause of the issue. You run down to the data center/lab and check all the physical connections. On doing a “show ip interface brief” on all the affected routers, you see that some of the interfaces have been taken down administratively and the others show a Protocol down. Most networking problems reside at the lower levels and hence troubleshooting layer 1 is the first step of a bottom-up approach. The following objective will help you find an easier and faster way to check the layer 1 status before moving up the OSI model for troubleshooting.

1. Configure descriptions for the router interfaces for easier administration (e.g. Router(config-if)# description Management Interface).
2. Write a script in a language of your choice (e.g. UNIX/Python) to extract and display interface information from all the routers in the above topology using the following MIB objects (Hint: you can view entire MIB details using SNMPBULKWALK command). [**25 points**]

* ifName
* ifDescr
* ifOperStatus
* ifPhysAddress
* ifAdminStatus
* ifInUcastPkts

**Sample output to be displayed by the script:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Interface Name | Description | Operational Status | Physical Address | Admin Status | Incoming Unicast Packet Counter |
| R1 | Fa0/0 | Management Interface | Up | 00-03-47-92-9C-6F | Up | 100 |



1. Modify the above script to retrieve and display: interface IP address and network mask information. [**2.5 points**]



1. Implement both the scripts (TCPDUMP Trap obj 4.3 and extract interface info obj 5.2) using just one script. Also, ensure your script shall continuously monitor the interface status, display the interface information (as in obj 5.2) and parse the trap (as in obj 4.3). [**2.5 points**]

Not sure what to do!

Report Questions (5 points each)

1. Would you recommend using a management subnet for SNMP? Why/why not?

Yes, it will be easier to manage, and between manager and devices should be in the same subnet.

1. Why is a switch used in the network design in GNS3?

We need a device(agent) to collect variable that in the MIB about the devices and then send message back to manager.

1. Can you use a router instead? Why/why not?

Yes, the router can perform same functionality like switch, but router is smarter, people use it for routing more.

1. If you used a router, what would need to change (if anything).

No, there is no different

1. What command has to be entered on the router, to disable configuration changes to be made through SNMP?

No snmp-server enable traps

Network Discovery using NMAP

Objectives

* Learn the basic operations of network discovery using Nmap.
* Learn how to capture and analyze ICMP traffic.
* Learn how to capture and analyze port scanning traffic.
* Perform IP address spoofing.
* Gather OS information.
* Perform Scripting and Automation.

Summary

Nmap is a free open source tool that can be used for performing a variety of network scanning and security functions. To create a “map” of the network, Nmap sends specific packets to the target host (or hosts) and then analyzes the responses. Nmap can also be used to enumerate networks and avoid IDS through spoofing/stealth, please use this responsibly and follow the lab directions.

Nmap is available for download for many Linux distributions (There is also a version available for Windows). It also comes with a GUI (Zmap) that can be used as an alternative to the CLI. The functions of this lab will focus on ping sweeps (find hosts), port scanning (determine vulnerabilities/services), IP spoofing (avoiding detection by IDS), and gathering intelligence on a network.

Objective 1: Download and Install Nmap/Zmap on Your Machine

Follow the instructions from the Nmap website for your operating system:

<https://nmap.org/>

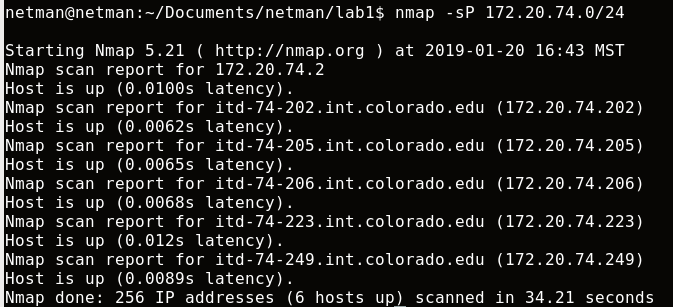
For the remainder of this lab, you can use **Nmap or Zmap**

Objective 2**:** Ping Sweeps and Port Scans

1. Perform a ping sweep for the following network (Note: this only works from CU

network or VPN; if unavailable use your home/private network): **172.20.74.0/24**

a. Provide a screenshot showing the command and the results [**5 points**]



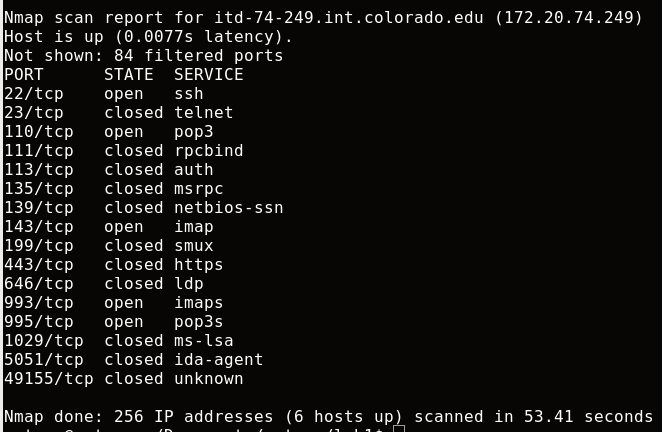
b. How many devices responded to the ping sweep? Provide information about how you can determine this. [**2.5 points**]

There are 6 hosts responded to the ping sweep because I use /24 network in command. You can see the last line Nmap done: 256 IP addresses (6 hosts up)

2. Choose a host that replied from the ping sweep; now perform a full scan on that host

a. Which well-knownports were open on this machine? Provide the screenshot. [**2.5 points**]

When I add -F(fast scan) option in command, it will give me common ports: Port: 22, 23, 443



b. Provide the command you would use to perform a “stealth”scan. [**2.5 points**]

sudo nmap -sS <host> [1]

Objective 3: IP Spoofing and OS Detection

1. Perform a full network scan on the /24 network (optional: use a spoofed IP

address (use target IP address from previous objective as the source))

a. Provide the command used [**2.5 points**]

sudo nmap -sP 172.20.74.0-255

Note: -s: src, -P: ping sweep, -255: /24

b. Explain the different “state”options for a Nmap port scan (i.e. open, filtered, closed, etc.) [**2.5 points**]

[1]Open state: Technically, this port if open port for attack, especially for attackers and pen-testers because this state is accepting TCP connections and UDP datagrams or SCTP associations for this port, but administrators try to close or protect this port.

Closed state: This port is still accessible but there is no application listening to it. It is still helpful in showing that a host is with an IP address like ping scanning. Technically, administrator might want to block this kind of port with firewall.

Filtered: nmap cannot determine if the port is open because this state filter prevents its probes from reaching the port. Because they provide only little information so these port frustrate attackers. Sometiems they respond with ICP error messages like type 3 code 13 (destination unreachable: communication administratively prohibited), but filters that simply drop probes without responding are far more common.

Unfiltered: If you use other scan types like window scan, SYN scan, or FIN scan, these can help resolve if the port is open because this state means that a port is accessible but nmap is not able to determine if it is open or closed. Only the ACK scan that is used to map firewall rulesets, classifies ports into this state.

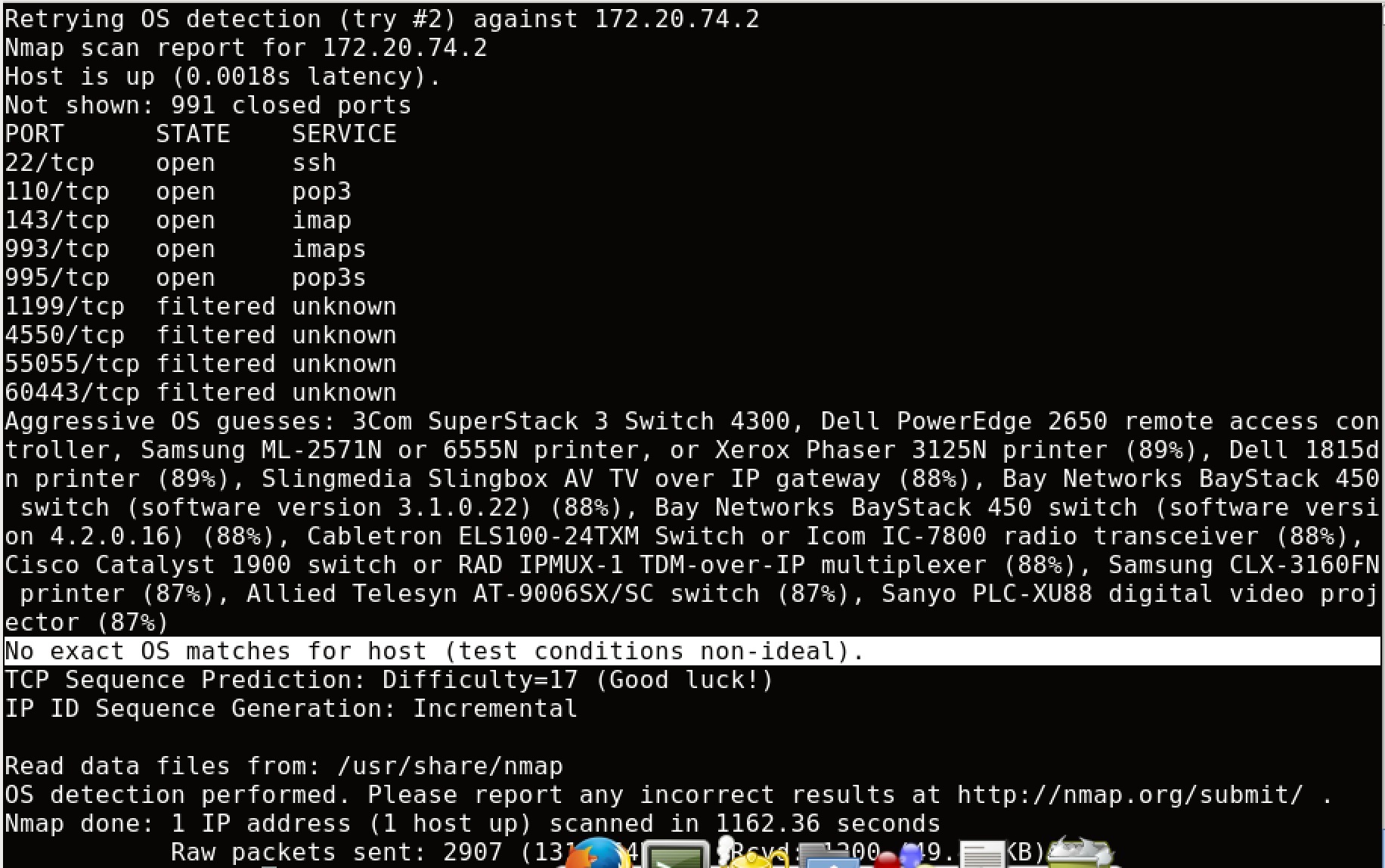
Open/filtered: When the scan types that open ports give nothing for response, nmap put ports in this sate when it is not able to determine if a port is open or filtered. This could happen because of a packet filter drop the probe, then nmap does not know if the port is open or filtered for sure.

Closed/filtered state: This state is used when nmap is not able to determine if a port is filtered or closed. This is only used for the IP ID idle scan

2. Provide screenshots of the Operating Systems running on each of these

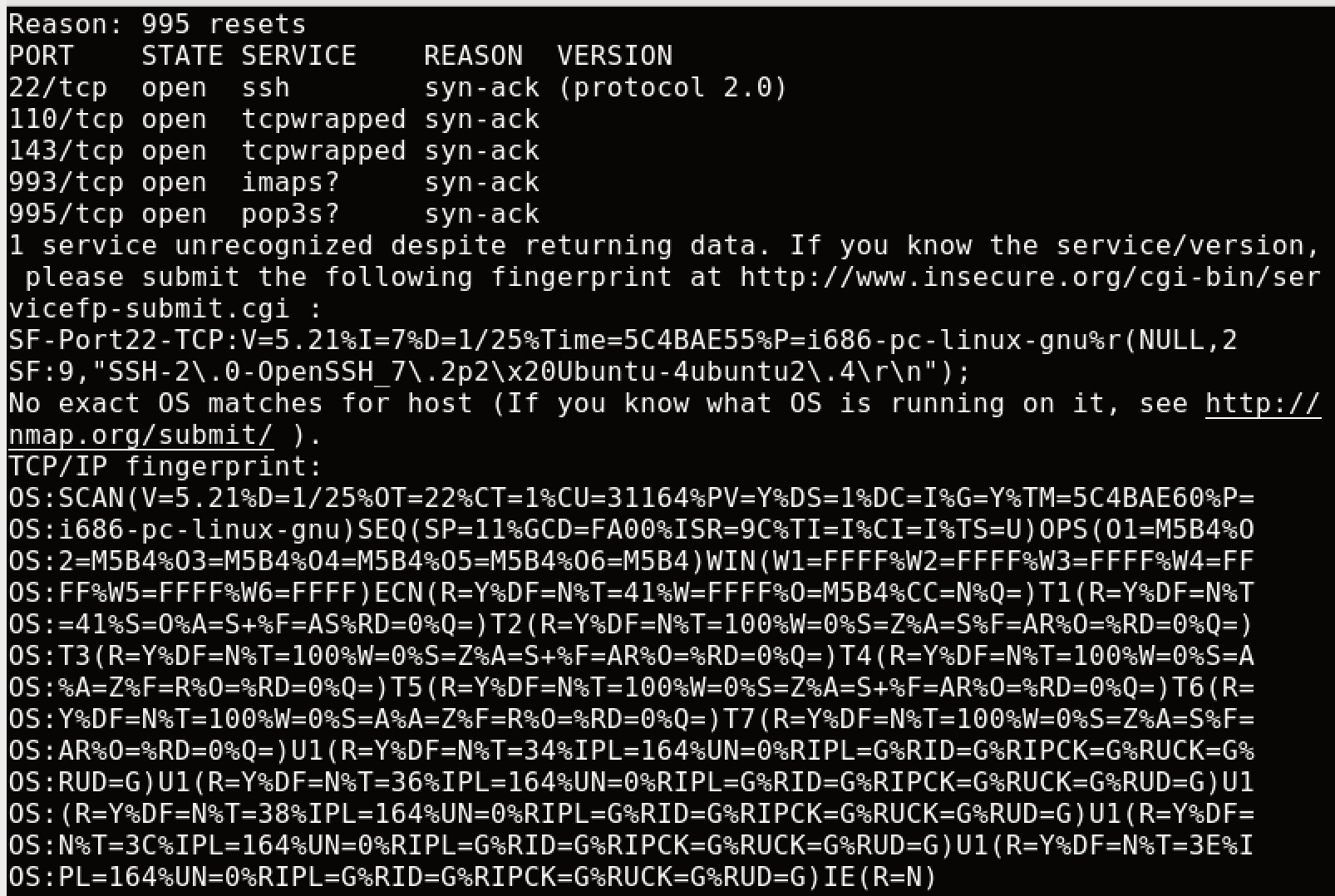
machines [**2.5 points**]

172.20.74.2:

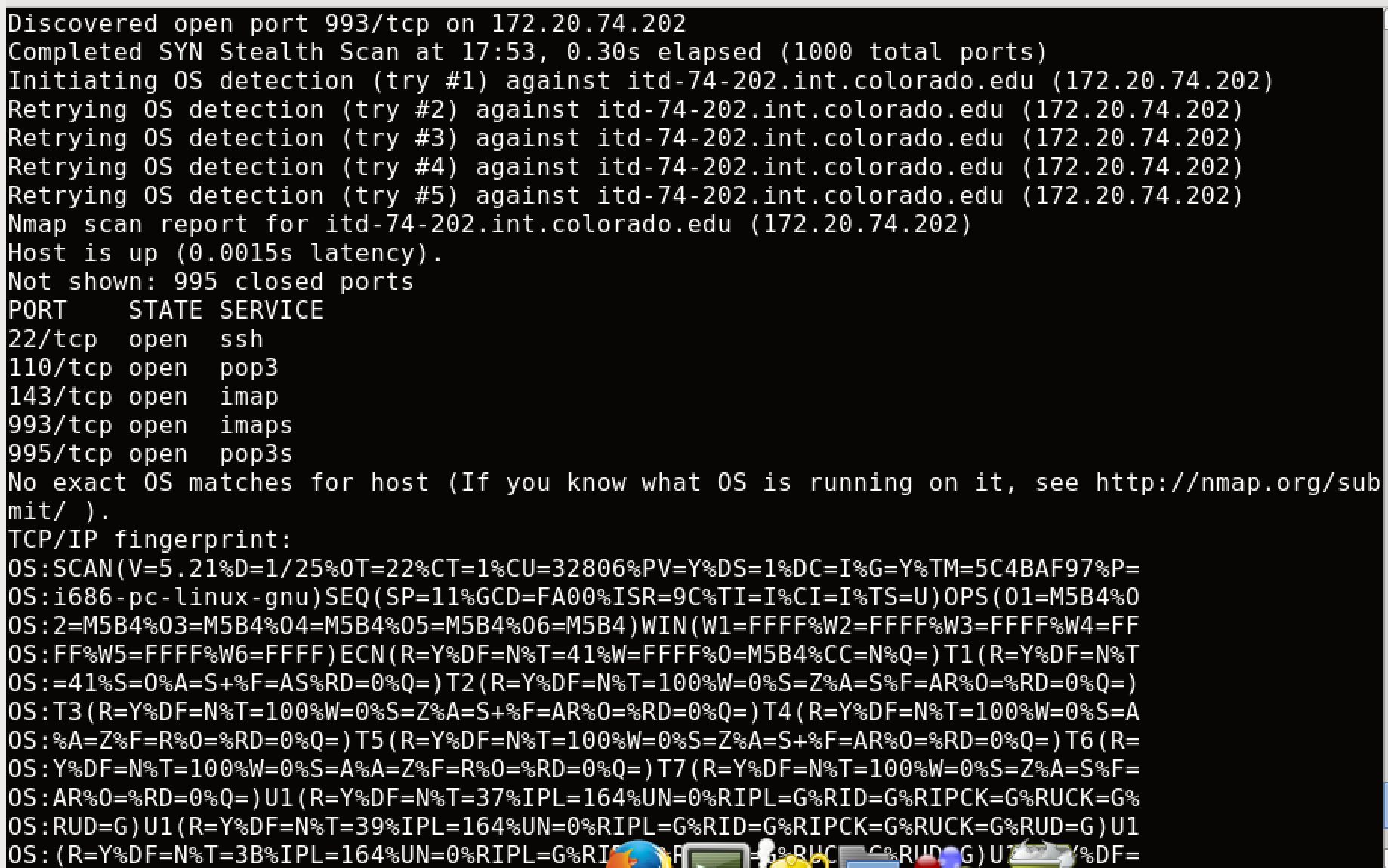


172.20.74.202 is failed to match operating system[4]

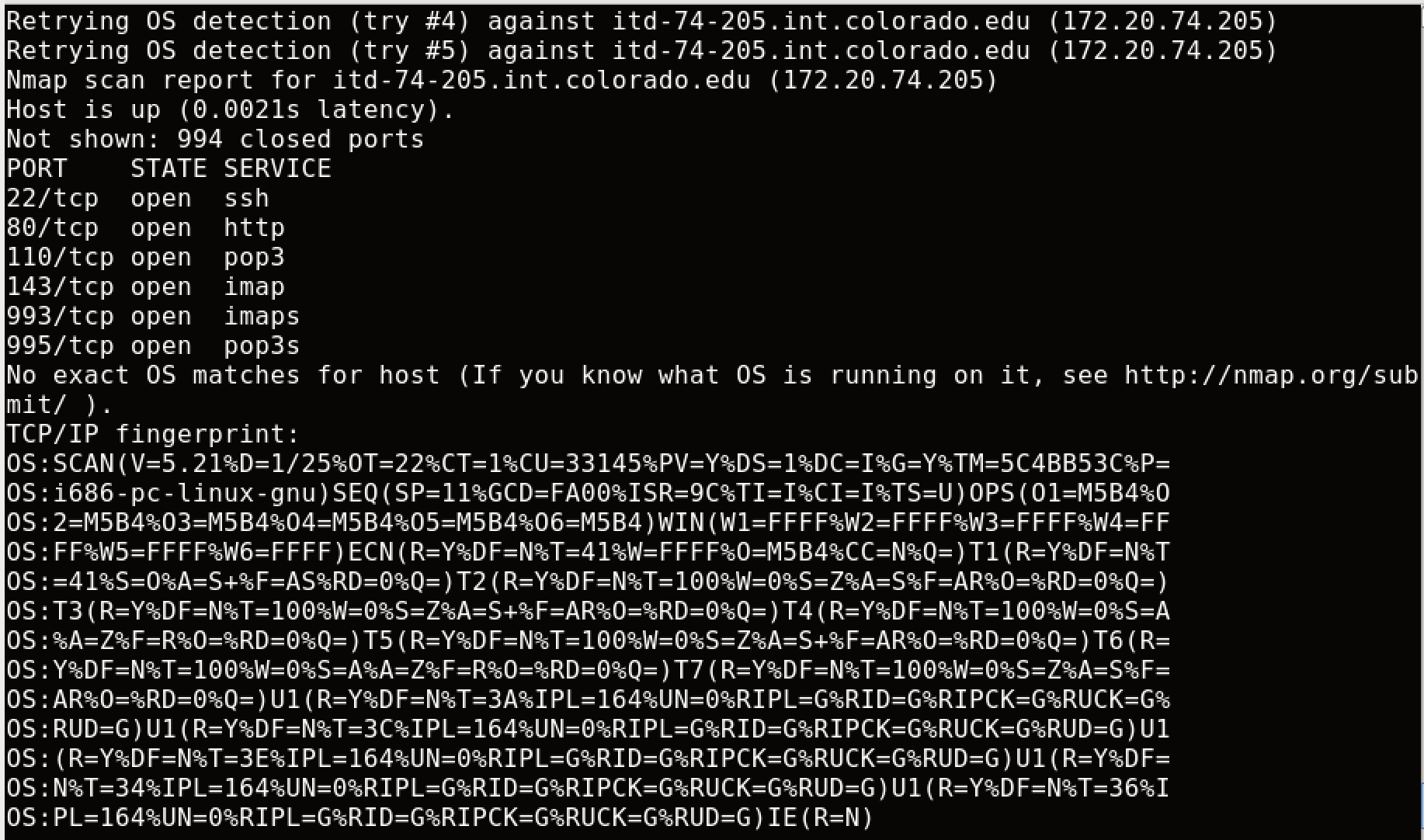
My command: sudo nmap -v -Pn -A 172.20.74.202



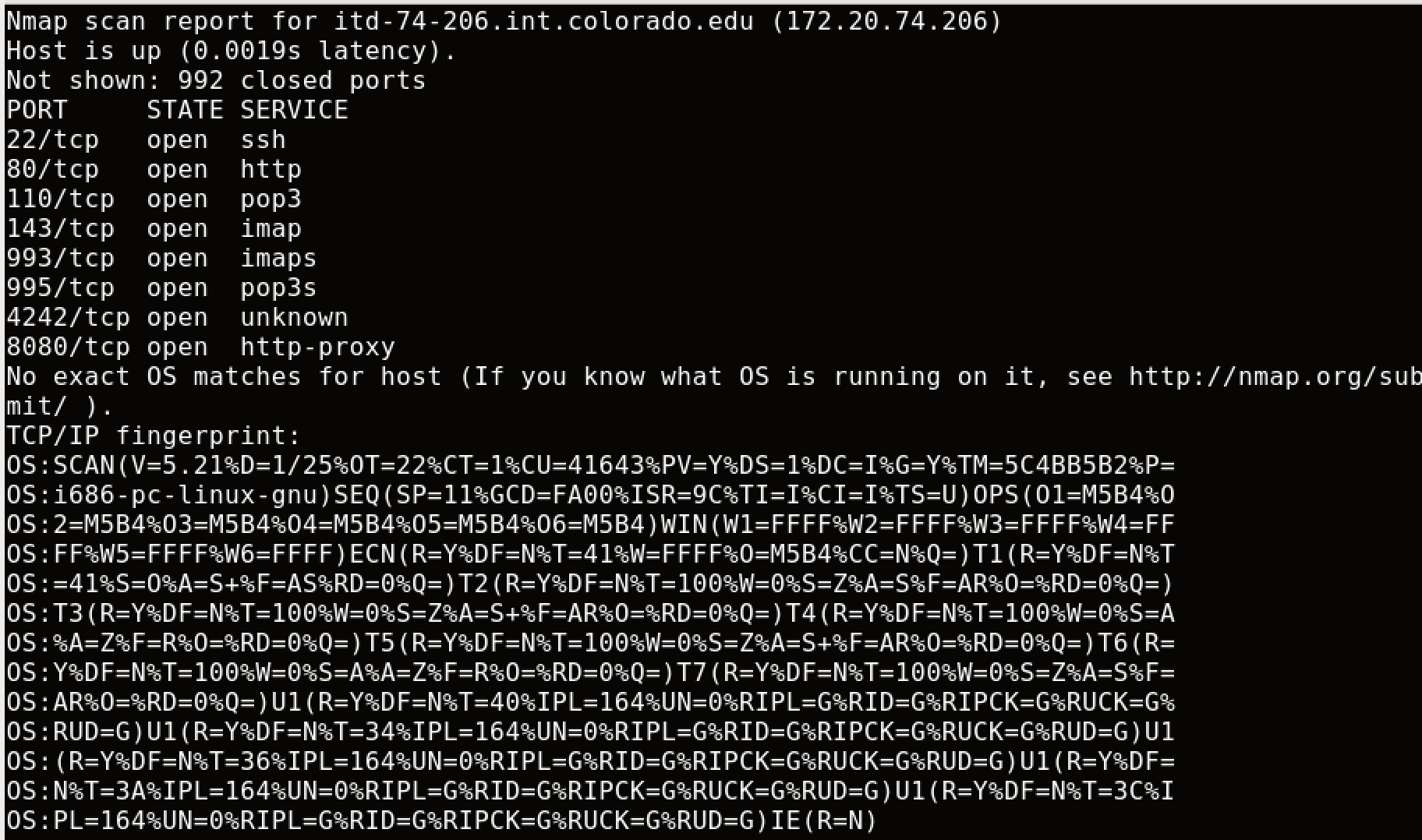
Or ‘sudo nmap -v -Pn -O 172.20.74.202



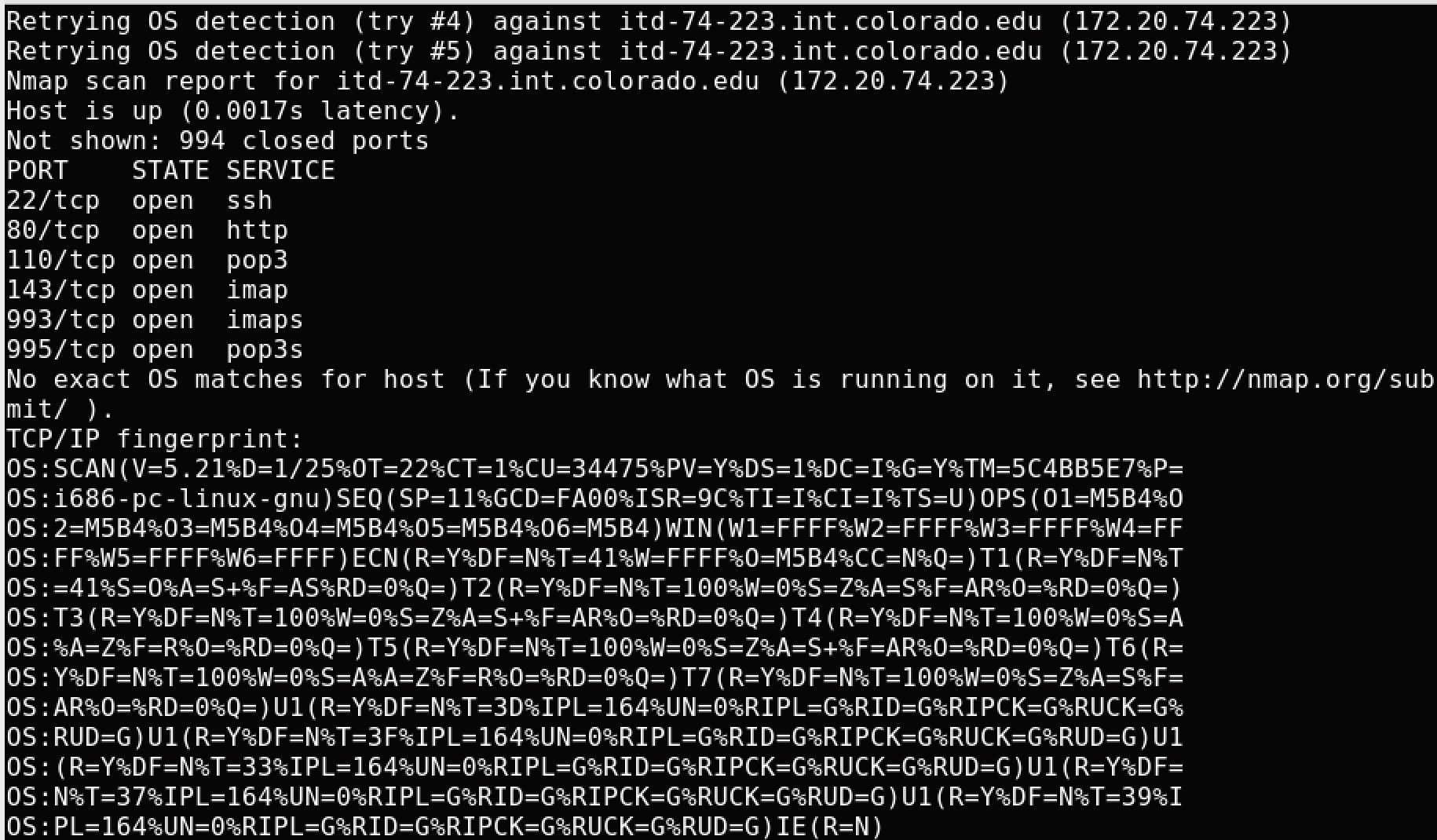
172.20.74.205 with ﻿sudo nmap -v -Pn -O 172.20.74.205



172.20.74.206 with ﻿sudo nmap -v -Pn -O 172.20.74.206:

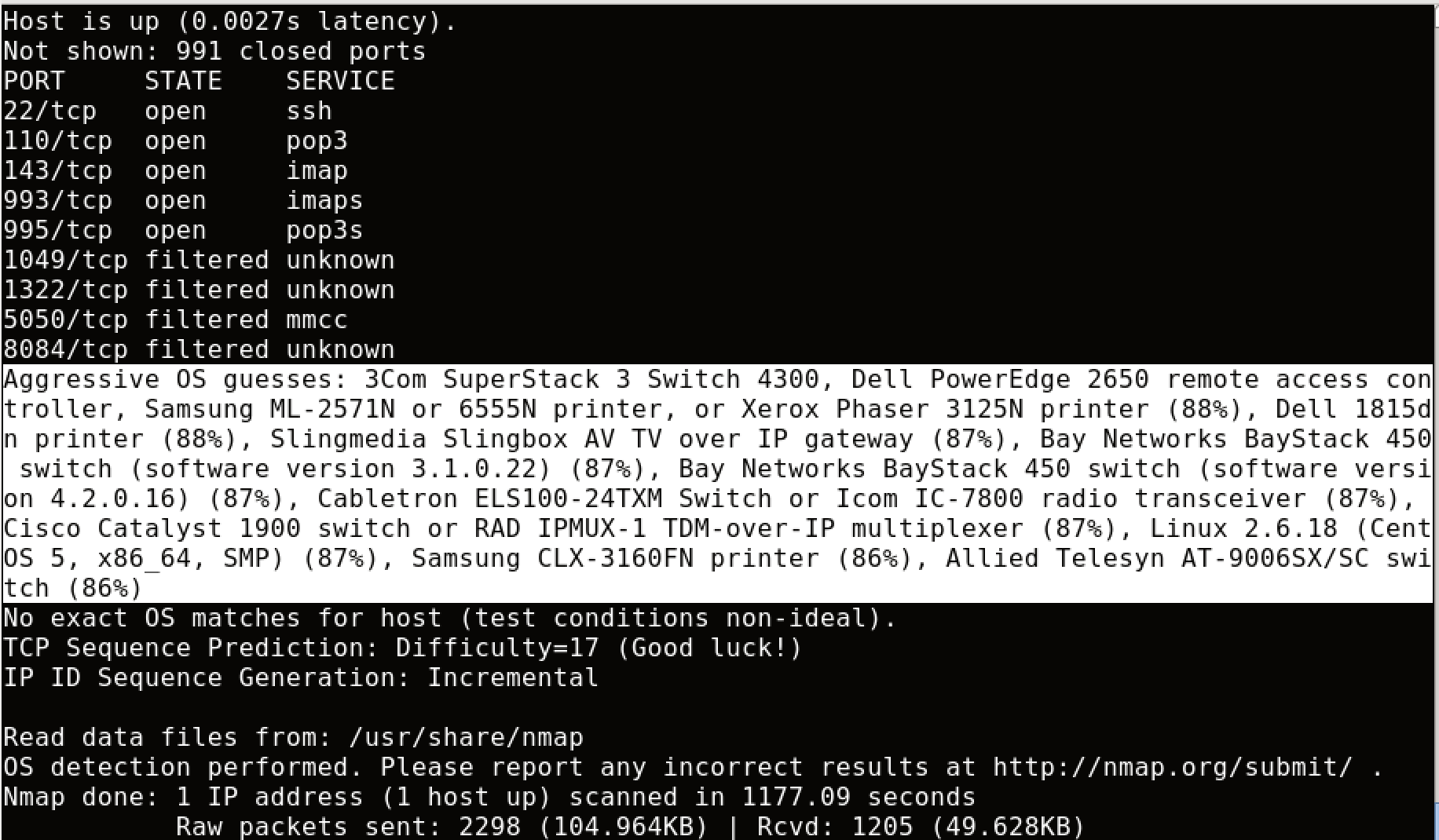


172.20.74.223 with ﻿sudo nmap -v -Pn -O 172.20.74.223:



172.20.74.249 with ﻿sudo nmap -v -Pn -O 172.20.74.249:

Finally, there is some guess from nmap for this host:



It’s Dell

Objective 4: Scripting and Automation

1. IP Address Mapping

a. If using the VM, install Nmap

**#sudo apt-get install nmap**

b. Run a ping sweep on the /24 network

c. Using **Bash or Python**, record the IP addresses into a **text/CSV** file d. Repeat the ping sweep after some time (~10 min.)

e. Compare the two files

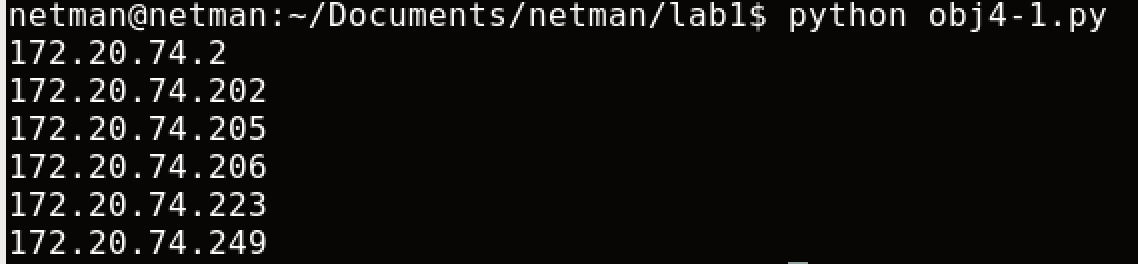
i. Were there any differences? If so, what is different? [**2 points**]

Yes, a little bit, sometimes, I can only scan up to 4 hosts instead of 6 hosts. But my current CSV file is 6 hosts because I accidentally ran the program again.

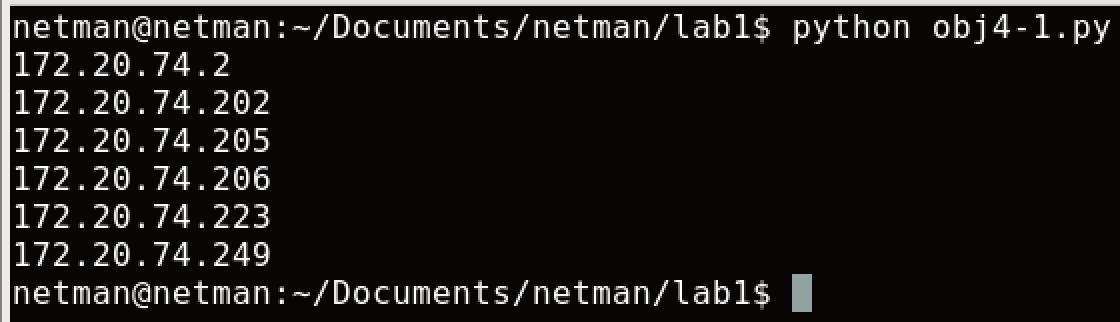
ii. Submit the scripts, files, procedures or screenshots of how you accomplished this [**10 points**]

I used python to do the command: ‘/usr/bin/nmap -n -sP 172.20.74.0/24’ and store the output into a list, and I use regex to match on ipv4 address format and save it to files.

CSV file:



Txt file:



f. As a network manager, list one thing that is useful and one thing could be detrimental with this information [**5 points**]

Useful: find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Detrimental: It might not be accurate every time. It could miss some important information.

2. Rogue Web Server (web servers ending with IP addresses .1-.10 are legitimate;

outside of that range are rogue)

a. Run a full network port scan to find open ports for **80, 443**, and **8080**

b. Submit the file of all web servers that are not in the range (i.e. rogue web server)

The rouge servers are:

172.20.74.206

172.20.74.223

172.20.74.205

172.20.74.206

i. How did you accomplish this? [**5 points**]

I did the command: ﻿ nmap 172.20.74.\* -p 80,8080,8088,443 -sV -vv [2]: This will run TCP SYN scan for the range of 172.20.74.0 through 172.20.74.254, but only the ports 80, 8080, 8088, 443 are scanned. If any open ports are identified, Nmap command will do a version scan (-sV) and the output from the scan will show all text at a verbosity level of two (-vv)

Report Questions

1. How can you set a decoy, to hide your source IP address using Nmap? [**2.5 points**]

To set decoy, add -D option in the nmap command[6], this is generally an effective technique to hide your ip address. However, this can still be traced back by Router path tracing.

2. List some ways Nmap can be used to trick a firewall. [**2.5 points**]

You can add -f option with the nmap command [5]:

This technique is for some firewall that is not properly configured. This provides fragmenting packets while scanning the ports, then avoid the firewalls inspection.

Ex:

Sudo nmap -f <target>

Another is Specify the specific MTU (Maximum Transmission Unit) by doing[5]:

Sudo nmap –mtu <mtu size> <target>

This could cause the confusion for firewall.

# Total Score = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ 202

References:

1. <https://nmap.org/bennieston-tutorial/>
2. https://www.professormesser.com/nmap/using-nmap-to-find-rogue-devices/
3. <http://www.whyismynetworkslow.com/?p=92>
4. <https://nmap.org/book/osdetect-unidentified.html>
5. <https://pentestlab.blog/2012/04/02/nmap-techniques-for-avoiding-firewalls/>
6. https://www.cyberciti.biz/tips/nmap-hide-ipaddress-with-decoy-ideal-scan.html