Lab - Linux Servers (Instructor Version)

**Instructor Note**: Red font color or gray highlights indicate text that appears in the instructor copy only.

# Objectives

In this lab, you will use the Linux command line to identify servers running on a given computer.

Part 1: Servers

Part 2: Using Telnet to Test TCP Services

# Recommended Equipment

* CyberOps Workstation virtual machine

**Instructor Note**: This lab can be done using the virtual machine created in a previous lab.

# Instructions

## Servers

Servers are essentially programs written to provide specific information upon request. Clients, which are also programs, reach out to the server, place the request, and wait for the server response. Many different client-server communication technologies can be used, with the most common being IP networks. This lab focuses on IP network-based servers and clients.

### Access the command line.

* + - 1. Log on to the CyberOps Workstation VM as the **analyst**,using the password **cyberops**. The account **analyst** is used as the example user account throughout this lab.
      2. To access the command line, click the **terminal** icon located in the Dock, at the bottom of VM screen. The terminal emulator opens.



### Display the services currently running.

Many different programs can be running on a given computer, especially a computer running a Linux operating system. Many programs run in the background so users may not immediately detect what programs are running on a given computer. In Linux, running programs are also called *processes*.

**Note**: The output of your **ps** command will differ because it will be based on the state of your CyberOps Workstation VM.

* + - 1. Use the **ps** command to display all the programs running in the background:

[analyst@secOps ~]$ **sudo ps –elf**

[sudo] password for analyst:

F S UID PID PPID C PRI NI ADDR SZ WCHAN STIME TTY TIME CMD

4 S root 1 0 0 80 0 - 2250 SyS\_ep Feb27 ? 00:00:00 /sbin/init

1 S root 2 0 0 80 0 - 0 kthrea Feb27 ? 00:00:00 [kthreadd]

1 S root 3 2 0 80 0 - 0 smpboo Feb27 ? 00:00:00 [ksoftirqd/0]

1 S root 5 2 0 60 -20 - 0 worker Feb27 ? 00:00:00 [kworker/0:0H]

1 S root 7 2 0 80 0 - 0 rcu\_gp Feb27 ? 00:00:00 [rcu\_preempt]

1 S root 8 2 0 80 0 - 0 rcu\_gp Feb27 ? 00:00:00 [rcu\_sched]

1 S root 9 2 0 80 0 - 0 rcu\_gp Feb27 ? 00:00:00 [rcu\_bh]

1 S root 10 2 0 -40 - - 0 smpboo Feb27 ? 00:00:00 [migration/0]

1 S root 11 2 0 60 -20 - 0 rescue Feb27 ? 00:00:00 [lru-add-drain]

5 S root 12 2 0 -40 - - 0 smpboo Feb27 ? 00:00:00 [watchdog/0]

1 S root 13 2 0 80 0 - 0 smpboo Feb27 ? 00:00:00 [cpuhp/0]

5 S root 14 2 0 80 0 - 0 devtmp Feb27 ? 00:00:00 [kdevtmpfs]

1 S root 15 2 0 60 -20 - 0 rescue Feb27 ? 00:00:00 [netns]

1 S root 16 2 0 80 0 - 0 watchd Feb27 ? 00:00:00 [khungtaskd]

1 S root 17 2 0 80 0 - 0 oom\_re Feb27 ? 00:00:00 [oom\_reaper]

<some output omitted>

#### Question:

Why was it necessary to run **ps** as root (prefacing the command with **sudo**)?

Type your answers here.

Some processes do not belong to the analyst user and may not be displayed if ps was executed as analyst, which is a regular user account.

* + - 1. In Linux, programs can also call other programs. The **ps** command can also be used to display such process hierarchy. Use **–ejH** options to display the currently running process tree after starting the nginx webserver with elevated privileges.

**Note**: The process information for the nginx service is highlighted. Your PID values will be different.

[analyst@secOps ~]$ **sudo /usr/sbin/nginx**

[analyst@secOps ~]$ **sudo ps –ejH**

[sudo] password for analyst:

PID PGID SID TTY TIME CMD

1 1 1 ? 00:00:00 systemd

167 167 167 ? 00:00:01 systemd-journal

193 193 193 ? 00:00:00 systemd-udevd

209 209 209 ? 00:00:00 rsyslogd

210 210 210 ? 00:01:41 java

212 212 212 ? 00:00:01 ovsdb-server

213 213 213 ? 00:00:00 start\_pox.sh

224 213 213 ? 00:01:18 python2.7

214 214 214 ? 00:00:00 systemd-logind

216 216 216 ? 00:00:01 dbus-daemon

221 221 221 ? 00:00:05 filebeat

239 239 239 ? 00:00:05 VBoxService

287 287 287 ? 00:00:00 ovs-vswitchd

382 382 382 ? 00:00:00 dhcpcd

387 387 387 ? 00:00:00 lightdm

410 410 410 tty7 00:00:10 Xorg

460 387 387 ? 00:00:00 lightdm

492 492 492 ? 00:00:00 sh

503 492 492 ? 00:00:00 xfce4-session

513 492 492 ? 00:00:00 xfwm4

517 492 492 ? 00:00:00 Thunar

1592 492 492 ? 00:00:00 thunar-volman

519 492 492 ? 00:00:00 xfce4-panel

554 492 492 ? 00:00:00 panel-6-systray

559 492 492 ? 00:00:00 panel-2-actions

523 492 492 ? 00:00:01 xfdesktop

530 492 492 ? 00:00:00 polkit-gnome-au

395 395 395 ? 00:00:00 nginx

396 395 395 ? 00:00:00 nginx

408 384 384 ? 00:01:58 java

414 414 414 ? 00:00:00 accounts-daemon

418 418 418 ? 00:00:00 polkitd

<some output omitted>

#### Question

How is the process hierarchy represented by **ps**?

**Type your answers here.**

Through indentation.

* + - 1. As mentioned before, servers are essentially programs, often started by the system itself at boot time. The task performed by a server is called a *service.* In such fashion, a web server provides web services.

The **netstat** command is a great tool to help identify the network servers running on a computer. The power of **netstat** lies on its ability to display network connections.

**Note**: Your output maybe different depending on the number of open network connections on your VM.

In the terminal window, type **netstat**.

[analyst@secOps ~]$ **netstat**

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost.localdo:48746 localhost.local:wap-wsp ESTABLISHED

tcp 0 0 localhost.localdo:48748 localhost.local:wap-wsp ESTABLISHED

tcp6 0 0 localhost.local:wap-wsp localhost.localdo:48748 ESTABLISHED

tcp6 0 0 localhost.local:wap-wsp localhost.localdo:48746 ESTABLISHED

tcp6 0 0 localhost.local:wap-wsp localhost.localdo:48744 ESTABLISHED

tcp6 0 0 localhost.localdo:48744 localhost.local:wap-wsp ESTABLISHED

Active UNIX domain sockets (w/o servers)

Proto RefCnt Flags Type State I-Node Path

unix 3 [ ] DGRAM 8472 /run/systemd/notify

unix 2 [ ] DGRAM 8474 /run/systemd/cgroups-agent<some output omitted>

As seen above, **netstat** returns lots of information when used without options. Many options can be used to filter and format the output of **netstat**, making it more useful.

* + - 1. Use **netstat** with the **–tunap** options to adjust the output of **netstat**.Notice that **netstat** allows multiple options to be grouped together under the same “**-**“ sign.

The information for the nginx server is highlighted.

[analyst@secOps ~]$ **sudo netstat -tunap**

[sudo] password for analyst:

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:6633 0.0.0.0:\* LISTEN 257/python2.7

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 395/nginx: master

tcp 0 0 0.0.0.0:21 0.0.0.0:\* LISTEN 279/vsftpd

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 277/sshd: /usr/bin

tcp6 0 0 :::22 :::\* LISTEN 277/sshd: /usr/bin

udp 0 0 192.168.1.15:68 0.0.0.0:\* 237/systemd-network

#### Question:

What is the meaning of the **–t**, **-u**, **–n**, **–a** and **–p** options in **netstat**? (use **man netstat** to answer)

Type your answers here.

-a: shows both listen and non-listening sockets. -n: use numeric output (no DNS, service port or username resolution), -p: show the PID of the connection owner process. -t: shows TCP connections. –u: shows UDP connections

Is the order of the options important to **netstat**?

Type your answer here.

No, the option order is irrelevant.

Clients will connect to a port and, using the correct protocol, request information from a server. The **netstat** output above displays a number of services that are currently listening on specific ports. Interesting columns are:

* 1. The first column shows the Layer 4 protocol in use (UDP or TCP, in this case).
  2. The third column uses the **<ADDRESS:PORT>** format to display the local IP address and port on which a specific server is reachable. The IP address 0.0.0.0 signifies that the server is currently listening on all IP addresses configured in the computer.
  3. The fourth column uses the same socket format **<ADDRESS:PORT>** to display the address and port of the device on the remote end of the connection. 0.0.0.0:\* means that no remote device is currently utilizing the connection.
  4. The fifth column displays the state of the connection.
  5. The sixth column displays the process ID (PID) of the process responsible for the connection. It also displays a short name associated to the process.

#### Question:

Based on the **netstat** output shown in item (d), what is the Layer 4 protocol, connection status, and PID of the process running on port 80?

Type your answers here.

TCP, LISTEN and 395.

While port numbers are just a convention, can you guess what kind of service is running on port 80 TCP?

Type your answers here.

This is probably a web server.

* + - 1. Sometimes it is useful to cross the information provided by **netstat** with **ps**. Based on the output of item (d), it is known that a process with **PID 395** is bound to TCP port 80. Port 395 is used in this example. Use **ps** and **grep** to list all lines of the **ps** output that contain **PID 395**. Replace 395 with the PID number for your particular running instance of nginx:

[analyst@secOps ~]$ **sudo ps -elf | grep 395**

[sudo] password for analyst:

1 S root 395 1 0 80 0 - 1829 19:33 ? 00:00:00 nginx: master process /usr/bin/nginx

5 S http 396 395 0 80 0 - 1866 19:33 ? 00:00:00 nginx: worker process

0 S analyst 3789 1872 0 80 0 - 1190 19:53 pts/0 00:00:00 grep 395

In the output above, the **ps** command is piped through the **grep** command to filter for only the lines containing the number 395. The result is three lines with text wrapping.

The first line shows a process owned by the **root** user (third column), started by another process with PID 1 (fifth column), at 19:33 (twelfth column)

The second line shows a process with PID 396, owned by the **http** user, started by process 395, at 19:33.

The third line shows a process owned by the **analyst** user, with PID 3789, started by a process with PID 1872, as the **grep 395** command.

#### Question:

The process PID 395 is **nginx**. How could that be concluded from the output above?

Type your answers here.

Based on the last column of line 1, the output shows nginx command line.

What is **nginx**? What is its function? (Use google to learn about nginx)

Type your answers here.

nginx is a lightweight webserver. A quick google search is extremely helpful on finding information about unidentified processes.

The second line shows that process 396 is owned by a user named http and has process number 395 as its parent process. What does that mean? Is this common behavior?

Type your answers here.

It means that nginx started process 396 under the http username. This is normal as nginx runs itself for every client that connects to port 80 TCP.

Why is the last line showing grep 395?

Type your answers here.

Because the grep 395 was used to filter the ps output, when the output was compiled, grep 395 was still running and therefore, it appeared in the list.

## Using Telnet to Test TCP Services

Telnet is a simple remote shell application. Telnet is considered insecure because it does not provide encryption. Administrators who choose to use Telnet to remotely manage network devices and servers will expose login credentials to that server, as Telnet will transmit session data in clear text. While Telnet is not recommended as a remote shell application, it can be very useful for quickly testing or gathering information about TCP services.

The Telnet protocol operates on port 23 using TCP by default. The **telnet** client however, allows for a different port to be specified. By changing the port and connecting to a server, the **telnet** client allows for a network analyst to quickly assess the nature of a specific server by communicating directly to it.

**Note**: It is strongly recommended that **ssh** be used as remote shell application instead of **telnet**.

* + - 1. In Part 1, **nginx** was found to be running and assigned to port 80 TCP. Although a quick internet search revealed that **nginx** is a lightweight web server, how would an analyst be sure of that? What if an attacker changed the name of a malware program to **nginx**, just to make it look like the popular webserver? Use **telnet** to connect to the local host on port 80 TCP:

[analyst@secOps ~]$ **telnet 127.0.0.1 80**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

* + - 1. Press a few letters on the keyboard. Any key will work. After a few keys are pressed, press ENTER. Below is the full output, including the Telnet connection establishment and the random keys pressed (fdsafsdaf, this case):

fdsafsdaf

HTTP/1.1 400 Bad Request

Server: nginx/1.16.1

Date: Tue, 28 Apr 2020 20:09:37 GMT

Content-Type: text/html

Content-Length: 173

Connection: close

<html>

<head><title>400 Bad Request</title></head>

<body bgcolor="white">

<center><h1>400 Bad Request</h1></center>

<hr><center>nginx/1.16.1</center>

</body>

</html>

Connection closed by foreign host.

Thanks to the Telnet protocol, a clear text TCP connection was established, by the Telnet client, directly to the nginx server, listening on 127.0.0.1 port 80 TCP. This connection allows us to send data directly to the server. Because nginx is a web server, it does not understand the sequence of random letters sent to it and returns an error in the format of a web page.

Why was the error sent as a web page?

Type your answers here.

Nginx is a web server and as such, only speaks the HTTP protocol.

While the server reported an error and terminated the connection, we were able to learn a lot. We learned that:

* + - * 1. The **nginx** with PID 395 is in fact a web server.
        2. The version of **nginx** is1.16.1.
        3. The network stack of our CyberOps Workstation VM is fully functional all the way to Layer 7.

Not all services are equal. Some services are designed to accept unformatted data and will not terminate if garbage is entered via keyboard. Below is an example of such a service:

* + - 1. Looking at the **netstat** output presented earlier, it is possible to see a process attached to port 22. Use Telnet to connect to it.

Port 22 TCP is assigned to SSH service. SSH allows an administrator to connect to a remote computer securely.

Below is the output:

[analyst@secOps ~]$ **telnet 127.0.0.1 22**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

SSH-2.0-OpenSSH\_8.2

sdfjlskj

Invalid SSH identification string.

Connection closed by foreign host.

Use Telnet to connect to port 68. What happens? Explain.

Type your answers here.

Unable to connect because the connection is refused. Telnet is a TCP-based protocol and will not be able to connect to UDP ports.

# Reflection Questions

* 1. What are the advantages of using netstat?

Type your answers here.

Netstat allows for an analyst to display all the connections currently present on a computer. Source and destination addresses, ports, and process IDs can also be displayed, providing a quick overview of all connections present on a computer.

* 1. What are the advantages of using Telnet? Is it safe?

Type your answers here.

Yes, as long it is not used as a remote shell. It is perfectly safe to quickly test or gather information about a given network service.

End of document