Networks, Communications & Cyber Defence

Practical Session
TCP

Peter Norris
University of Warwick
October 2021

Contents

Starting assumptions

1: You have used Netkit **lstart** and can read the associated **lab.conf** file.

2: You know how to shut down virtual machines using **lhalt** and **lcrash**.

3: You have saved captured network traffic from a virtual machine using **tcpdump**.

4: You have viewed a packet capture file in the real Linux host using **wireshark**.

5: You are prepared to make accurate notes of what you do.

6: You will complete unfinished activities before the next timetabled lab session.

7: You will resolve what you do not understand by conducting your own careful (and ethically sound) experimentation and / or further reading.

TCP activities

Intended outcomes

8: Can pass data from a TCP client to a TCP listener in Linux using the **netcat** (aka **nc**) instruction.

9: Can analyse a captured pcap file using wireshark and explain how a TCP connection is established, used for reliable bi-directional data transfer and closed down.

10: Can adapt marginally incorrect instructions and / or Netkit configurations. (again; **there will be some deliberate errors**.)

Launching the Netkit lab

11: Open a Linux terminal, make the **~/nklabs/** directory your current directory:

```
12: cd nklabs/
13: pwd
```

14: As ever, use the **man** command to find out what a particular command does, however, note that **netcat** in the real host and virtual machines are different versions.

15: Copy the zipped archive **lab08.tar.gz** and save it in the **~/nklabs/** directory. This contains the configuration information for around 15 virtual machines split across two directories: **lab08a** directory and **lab08b** directory.

16: The overall arrangement of the virtual machines into subnets is almost identical to the lab06 and lab07 arrangement. The lab **lab08a** contains the **W, X** and part of the **D** LAN. The lab **lab08b** contains the **Y** and **Z** subnets and the remainder of the **D** LAN.

17: Extract the the contents of **lab08.tar.gz** into the **~/nklabs** directory. Assuming your current directory is **~/nklabs** , this can be achieved by:

```
18: tar -xvcf lab08.tar.zg
```

19: Place yourself in the newly created **lab08a** directory and list its contents:

```
20: cd lab08
21: ls
```

22: Look in the **lab.conf** file. Confirm that it corresponds to the diagram of the network you have made last week in your notes and the one below.

23: With your current directory as the **~/nklabs/lab08a** directory (which contains the **lab.conf** file), launch the seven virtual machines using the instruction:

```
24: lstart
```

25: Look carefully at the virtual machines as they start. In particular record any errors you see.

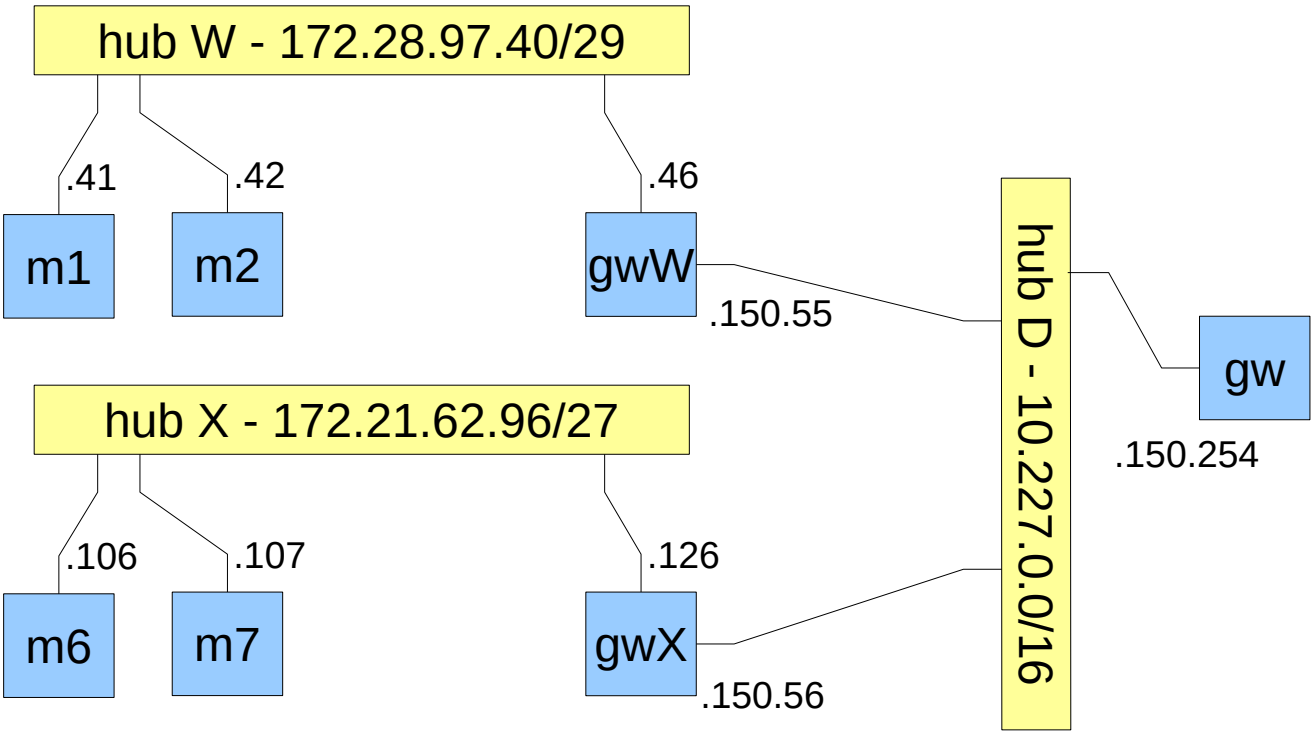
Look at the configuration

26: Once all machines are up, use **tcpdump** on machine **gwW** to store captured traffic in the file **gwW-d1.pcap** in the **lab08** directory you created earlier. The instructions will be similar to the one below.

```
27: tcpdump -s0 -i eth0 -w /hostlab/gwW-d1.pcap
```

28: Now on both **m1** and **m2**, start screen and initiate three screens sessions, (read the initial messages, then press whatever is suggested to get to the shell prompt).

```
29: screen
30: ctrl-a n
31: ctrl-a n
```



32: On both **m1** and **m2**, check the state of the machines' network connections using netstat. We will set it to update the output every second using the watch command (what is the default time interval?).

```
33: Watch -n1 netstat -ant4
```

34: On **m1** start a netcat tcp listener on port 80 (what is this in hex?)

```
35: nc -l -p 80
```

36: Now generate some TCP traffic. From machine **m2** , connect to the port 80 tcp listener on **m1** and send a few words in each direction.:

```
37: netcat ???? 80
```

38: On **m1** start another netcat tcp listener on port 80 in the unused screen. Then go back to the netstat “watch” screen and record what you have found.

```
39: nc -l -p 80
```

40: Go back to **m2** . First try to connect to port 3306 (this should fail – why?), then send the content of /etc/services via this new port 80 connection to **m1**.

```
41: dog /etc/services | netcat 17228.97.41 80 -q5
```

42: Create another netcat tcp connection, this time between machine **m2** and machine **m6**. Make **m6** be the listener on port 123 and **m2** the client on port 5555. Record exactly how you did this. (What is the usual service on port 123?)

43: Stop the tcpdump on **gwW**, then view it in wireshark. Make sure you do not have name resolution enabled for any layer. Find out how to view precisely one tcp stream (hint – highlight one packet in a tcp stream, then see what is below the *Analyse* menu header).

44: Find a FIN packet and make sure you can find the FIN bit set in the flags field. Do the same for SYN, ACK and RST.

45: Ensure you know the difference between absolute and relative sequence numbers in the wireshark display. Ensure you can display either.

46: Ensure you can find the port numbers used in any tcp stream.

Extension activity - transfer files and directories

47: In the UDP lab, you tried to move a file across the network using UDP. Try the same thing using TCP.

48: Try and achieve a similar effect for directories using the tar command. Tar can convert a directory tree structure into a linear byte stream. Originally this was to store directory structure and content onto a tape archive. Tar can also do the reverse, converting a correctly formatted byte stream back into a directory structure. The -c switch is used to create a new archive from files and directories. The -x switch is used to extract files and directories from a tape archive.

49: A set of files can be found on all machines under **/var/tart-stuff/**. Confirm the structure you find is consistent with the content of **shared.startup**.

50: Choose which direction you want to send things (to the listener or from the listener). Choose where you want to untar things (**/tmp** is a good choice if you are unsure).

51: Look at the man pages of nc and tar to achieve the effect you desire. As usual, have a tcpdump running somewhere on the path between the client and the listener to capture the traffic.

52: The commands you are looking for are along the lines of:

```
53: tar -czf - …. | nc …  # on the machine sending the tarball
54: nc … | tar -xzf - … # on the machine accepting the tarball
```

55: Note carefully the specific commands you used. How were you able to end the connection as soon as the transfer of the tarball was complete?

Extension activity – start a web server

56: Leave the first set of machines running which you started from the **lab08a** directory ie hubs **W** and **X** and some of hub **D.**

57: Debian provides multiple desktops aka workspaces. Select a new workspace (quickly switch between them using *ctrl alt right-arrow* or *ctrl alt left-arrow*) and in it open a terminal with the current directory set to **~/nklabs/lab08b**.

58: Start the netkit lab in the **lab08b** directory using **lstart**. This lab will also use hub **D** to create the overall configuration shown below, plus a web server and two Domain Name Servers (**www**, **dns1** and **dns2**) on hub **D** which are not shown on the diagram on the next page. Note that in this lab, addressing and routing is achieved using the **ip** instruction rather than **ifconfig** and **route**.

59: Start a packet capture on **gw**, then on **m1** do a browse using lynx.

```
60: nslookup m2.cyber.test
```

61: Stop the packet capture and explain what is happening in the UDP packets. Try typing *udp* into the wireshark filter - explain what happens.

Extension activity – emulate the lecture

62: Find the packet capture file which was created in the lecture and try and emulate it as closely as you can in the lab. Think about time, ip addresses and tcp port numbers (ie sockets). What about controlling initial sequence numbers? How close can you get it? Record accurately how you did this.

Extension activity – scan the network with nmap

63: Nmap is the network scanning tool of choice to discover both host and ports that are up and open. On the **gwW** machine, start tcpdump capturing traffic on eth0 and saving it to your ~/nklabs/lab08 directory.

64: Start the apache web server (look in www.startup for hints on how to do this) and other netcat tcp listeners on several ports on **m1** and **m2**. Look in /etc/services and choose port numbers that look common. Check with netstat that all is as expected.

65: On the **gw** machine, launch a network scan against the W LAN using the instruction:

```
66: nmap -sS 172.28.97.40/29
```

67: Stop the packet capture on **gwW** and analyse it in wireshark. Explain what is happening. Rationlise what is reported to **gw** to what you see in the captured traffic.

68: In any convenient terminal, execute the following and ensure you know what it is doing as ever, *man seq* is a good idea):

```
69: seq 10 5 60
```

70: On **m1**, start lots of listeners using the script:

```
71: for i in $(seq 10 5 200)
72: do
73:   nc -l -p $i &
74: done
```

75: Leave the m1 terminal with netstat being watched so you can see all the connections on the screen.

```
76: watch -n1 netstat -art4
```

77: Start another packet capture on **gwW** and launch another network scan from **gw**, this time only aim it at **m1** using the instruction

```
78: nmap -sT -p 1-100 -r 172.28.97.41
```

79: Stop the packet capture on **gwW** and analyse it in wireshark. Explain what is happening. In particular:
What does -sT do rather than -sS?
What does -p 1-100 do?
What does -r do?

80: Experiment with nmap only on networks where you have explicit permission from the network administrator. Be especially careful with all nmap instructions. Careless fingers can cause you to scan large chunks of the Internet. **This is likely to get you into trouble**.

References

81: Lyon, G (aka Fyodor); Nmap Security Scanner; https://nmap.org/

82: Postel, J; Trasmission Control Protocol; IETF RFC793; https://tools.ietf.org/html/rfc793