Networks, Communications & Cyber Defence

Practical Session
DHCP
ARP

Peter Norris
University of Warwick
September 2021

Contents

Acknowledgements

1: Various students have contributed to the evolution of these labs over the years. Particular thanks goes to Josh Hawking and his team for migrating Netkit to a modern kernel and Luke Spademan for migrating from net-tools to iproute2 commands.

Starting assumptions

2: You have used Netkit **lstart** to launch a coordinated group of several virtual machines. Specifically, you are aware of how this group is coordinated via the **lab.conf** and **lab.dep** files.

3: You know how to shut down virtual machines using **lhalt** or **lcrash**.

4: You have used **ifconfig** and **ping** within the virtual machines.

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

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

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

8: You will complete unfinished activities.

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

DHCP activities

Intended outcomes

10: Can configure a DHCP server in Linux via **dnsmasq**.

11: Can explain the main features of the DHCP protocol (discover, offer, request, acknowledge).

12: Can analyse the operation of the DHCP protocol in a pcap file using wireshark.

Launching the Netkit lab

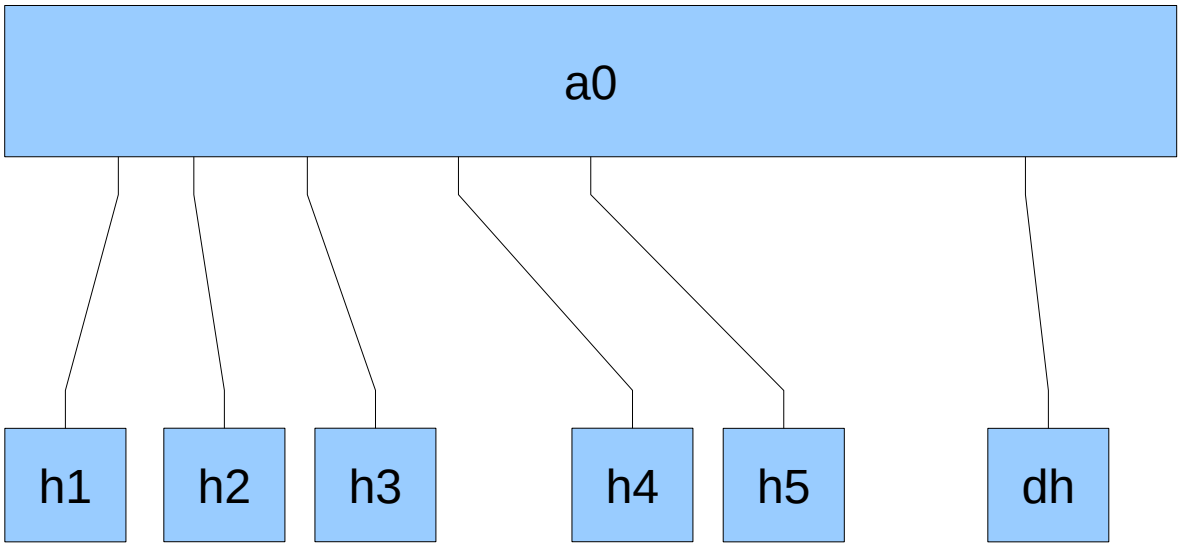
13: Open a Linux terminal, make the ~**/nklabs/** directory your current directory:

```
14: cd ~/nklabs/
15: pwd
```

16: As ever, use the **man** command to find out what a particular command does

17: Copy the zipped archive **lab04.tar.gz** and save it in the **~/nlabs/** directory. This contains the configuration information for seven virtual machines in the **lab04** directory.

18: This represents the configuration of a switch (**a0**), five hosts (**h1**, **h2**, **h3, h4, h5**) and a DHCP server (**dh**). The switch (**a0**) will **not** have a network (IP) address. Each of the five hosts and the DHCP server will get a network (IP) address one way or another. We have given the switch (**a0**) eight ports, connected six of them to **h1**, **h2**, **h3, h4, h5,** and **dh**, and left the two unused. Connectivity is as the diagram below (draw this in your own notes to annotate as the lab progresses):



19: Extract the the contents of **lab04.tar.gz** into the **~/nlabs** directory. Assuming your current directory is **~/nlabs** , this can be achieved by:

```
20: tar -xvzf lab04.tar.gz
```

21: Place yourself in the newly created **lab04** directory and list its contents:

```
22: cd lab04
23: ls
```

24: You will see seven directories (some empty, some with content), one for each virtual machine. Identify which virtual machines have been given a **/etc/network/interfaces** file. Look at the man page and note anything that particularly catches your attention:

```
25: man interfaces
```

26: Look in the **lab.conf** file. Confirm that it corresponds to the diagram of the network you have made in your notes.

27: Look at the seven ***.startup** files, one for each machine. Identify which of these *directly* set the IP address via more modern **ip** or old-school **ifconfig** and which use **ifup** to utilise the settings in **/etc/network/interfaces**. Note the MAC address and IP address of each network interface card (NIC) on your diagram.

28: The file **a0.startup** defines the configuration of the bridge (aka switch) in an almost identical way to the week 3 *switch* lab.

29: With your current directory as the **~/nlabs/lab04** directory (which contains the **lab.conf** file), launch the seven virtual machines using the instruction:

```
30: lstart
```

31: You should see four virtual machines start up in quick succession Look in **lab.dep** and explain the order in which they start. In particular, identify why **h4** and **h5** seem to start before **h1, h2** and **h3**. Arrange the windows on the screen so they are consistent with the network diagram you have created in your notes.

32: Once all machines are up, dump all traffic passing through the switch (**a0**) for later analysis with wireshark using:

```
33: tcpdump -s0 -i sw0 -w /hostlab/a0-sw0-dump01.pcap
```

34: This file will persist after you log out because we have used **/hostlab** .

Look at the configuration

35: On all machines except the switch (**a0**), note the MAC address and IP address that each machine has for its eth0 NIC. Note precisely where each came from.

```
36: ip addr show dev eth0
```

37: Confirm your understanding of precisely where **h1** and **h3** got their MAC / IP addresses from. Make changes on the host file system using **gedit** so each has a different MAC and gets a different IP address. Predict what will happen (write it down). Stop and restart **h1, h3** and **dh** and confirm you are right.

```
38: lcrash h1 h3 dh
39: lstart h1 h3 dh
```

40: On **h1, h2** and **h3** virtual machines, look at the DNS information in their files **/etc/resolv.conf** Note what you find.

```
41: cat /etc/resolv.conf
```

42: Use your favourite search engine to find out how to drive the text editor **vim**. **I** for insert, **esc, :q** to quit and **:wq** to write changes and quit will get you started. vim and nano are installed editors in the Netkit virtual machines.

43: Change the settings on the running **dh** virtual machine so the DNS domain is changed to **nccd.cyber.test** and add a new preferred nameserver with an IP address of **208.67.222.222** For the changes to take effect you will need to stop and restart dhcp server on the **dh** machine. Red failure messages indicate you messed something up. (Look in **/var/log/daemon.log** to see what might be wrong.)

```
44: vim /etc/dnsmasq.local.conf
45: /etc/init.d/dnsmasq restart
```

46: You can either wait up to five minutes until the leases run out or you can force a client (**h3** for example) to make a new DHCP request

```
47: dhclient -v eth0
```

48: Look in **/var/log/daemon.log** on the dh machine to see its log of allocations. Look in **/var/lib/misc/dhcpd.leases** to see the details of the leases that have been issued. Anything you particularly notice?

```
49: cat /var/log/daemon.log
50: cat /var/lib/misc/dnsmasq.leases
```

51: Look at the route information on **h3**.

```
52: ip route
```

53: Change the settings on the dhcp server's **/etc/dnsmasq.local.conf** file so the default gateway is 192.168.97.1. Restart the dhcp server (you have already done this once so you know how to do it). Get **h3** to issue another dhcp client request and confirm the gateway device (look for a "G" under flags to find the gateway) has changed.

54: In the switch (a0), use ctrl-C to stop the packet capture and look carefully at the traffic. Associate what you find with the various activities so far. In particular, note the frame numbers in the capture file that associate with your activities in the various hosts. Also ensure that you expand the "Bootstrap Protocol" section in the middle panel to display the various DHCP options being sent. Confirm these change when you changed the DHCP setting (changing the Domain Name for example). Also confirm that what you see for the different option numbers corresponds to section 9 of RFC 2132 and that the overall message and protocol conforms to section 2 of RFC 2131.

ARP activities

Intended outcomes

55: Can explain how ARP enables a host to associate a MAC address with an IP address.

56: Can analyse ARP traffic in a pcap file using wireshark.

57: Can configure ARP parameters on a Linux host.

Generating ARP traffic

58: Using the machines from the DHCP activity, dump all traffic passing through the switch (**a0**) for later analysis with wireshark using:

```
59: tcpdump -s0 -i sw0 -w /hostlab/a0-sw0-dump02.pcap
```

60: As with the dump of DHCP traffic, this file will persist after you log out because we have used **/hostlab** (which maps to the directory containing **lab.conf**)

61: Look in the running **h4** virtual machine's ARP cache to see what it already knows. We will do this two ways: the old fashioned way using **arp** and the more modern way using **ip neigh**. We use the **-n** option to prevent IP addresses being resolved to host names.

```
62: arp -n
63: ip neigh show
```

64: On **h4**, quickly ping the IP address of each machine in turn, together with one non-existent address. Since we are pinging an IP address, ARP is going to have to discover the corresponding MAC address so it can send an ethernet frame to the correct NIC.

```
65: ping -c1 192.168.97.41
66: ping -c1 192.168.97.42
```

```
67: ping -c1 192.168.97.101
68: ping -c1 192.168.97.5
69: ping -c1 192.168.97.88
```

70: Look again at what **h4** knows about MAC address / IP address associations. Write these down.

```
71: arp -n
72: ip neigh show
```

73: Repeat this for **h3** and then for **h5**. Record what you find before and after pinging the other machines (remember the IP addresses will not all be exactly the same). Identify any discrepancies between what you found on the three machines. Look in their associated *.startup files and ensure you can explain why they differ.

74: Stop the traffic dump in a0 and open the file in wireshark. Ensure you can associate the frames that you see in wireshark with your ping activities

DHCP extension activities

75: Try and capture traffic corresponding to a DHCP release activity.

76: Try configuring two address pools: one for known MAC addresses and one for unknown MAC addresses.

77: Try overloading eth0 on the dh machine with extra IP addresses and see if you can configure the DHCP server to issue addresses on more than one subnet.

```
78: ifconfig eth0:1 172.20.20.20/27
79: ip addr add 10.11.12.13/28 dev eth0
```

ARP extension activities

80: Ettercap has variously (not) been pre-built into Netkit. It should be present in Netkit-ng from release 1.1.4. If you find it missing, a seriously non-trivial exercise for the student is therefore to install ettercap into a netkit machine. This is hard. Do not be disappointed if you do not manage it. Ettercap is a particularly aggressive Man-In-The-Middle (MITM) tool. **Do not use it on real networks. You have been warned**.

81: Start tcpdump to capture all traffic on the switch **a0**, then use ettercap on **h2** to poison the arp caches of **h1** and **h3**, thereby intercepting traffic passing between them. Look at the arp cache of all machines before, during and after the attack.

82: The instruction you are after is something along the lines of

```
83: ettercap -T -M arp /192.168.97.41/  /192.168.97.101/
```

84: Before the first slash, you put MAC addresses you are interested in (blank means any MAC address). After the second slash, you port the ports you are interested in (blank means any ports). The command we have above therefore provides MITM functionality between the two IP addresses, regardless of MAC address and port number.

85: On **h1** use netcat to set up a tcp listener on port 1234

```
86: netcat -l -p 1234
```

87: On **h3** connect to the listener that h1 set up

```
88: netcat 192.168.97.41 1234
```

89: Then just type in **h1** or **h3** (enter will cause the data to be sent - this is low budget instant messaging).

90: Ctrl-C will end the netcat sessions. Hitting q in the ettercap window will end the poisoning session. Then stop the packet capture and explain what happened.

References

[RFC0951] Croft, W. and J. Gilmore, "Bootstrap Protocol", RFC 951, September 1985. http://www.ietf.org/rfc/rfc0951.txt

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997. http://www.ietf.org/rfc/rfc2131.txt

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, November 1982. http://www.ietf.org/rfc/rfc0826.txt

Hawking, J et al, Netkit-jh, https://github.com/netkit-jh/netkit-jh-build/releases [accessed 25th Oct 2021]