网络空间安全实训实验报告

纪盛谦 57118218 2021-7-5

Task1.1：

1.1A:

在 docker 上构造并发送报文

```
>>> a.dst = '10.9.0.1'
>>> b = ICMP()
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
>>> a.dst = '10.9.0.5'
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
>>>
```

sniffing 程序抓取报文如下

```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoof
ing Lab/Labsetup/volumes# python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:bb:2f:e7:e8
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x66c9
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
```

可以看到，在 VM 上可以抓取到符合筛选条件的报文
如果不适用 root 权限则程序无法进行，因为没有相应的权限

```
[07/04/21]seed@VM:~/.../volumes$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 8, in <module>
    pkt = sniff(iface='br-20296f7b74dd',filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in
 sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in
_run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, i
n __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(typ
e))  # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

1.1B:

针对不同 filter，本实验中设置了不同的响应函数，分别输出 type1、type2、type3 以及报文内容，代码如下：

root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab/Labsetup/volumes

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


def print_pkt1(pkt):
    print("type1")
    pkt.show()

def print_pkt2(pkt):
    print("type2")
    pkt.show()

def print_pkt3(pkt):
    print("type3")
    pkt.show()



pkt1 = sniff(iface='br-20296f7b74dd',filter='icmp', prn=print_pkt1)
pkt2 = sniff(iface='br-20296f7b74dd',filter='tcp and src host 10.9.0.1 and dst port 23', prn=print_pkt2)
pkt3 = sniff(iface='br-20296f7b74dd',filter='net 128.230.0.0 mask 255.255.0.0', prn=print_pkt3)


~
~
```

在进行 type1 筛选时，构造报文：

```
>>> from scapy.all import *
>>> a = IP()
>>> a.dst = '10.9.0.1'
>>> b = ICMP()
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
```

Sniffing 检测结果：

```
root@VM:/volumes# python3 sniffer.py
type1
###[ Ethernet ]###
  dst       = 02:42:bb:2f:e7:e8
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
```

在进行 type2 筛选时，注释掉 pkt1 行的代码，构造响应报文：

```
>>> a = IP()
>>> a.dst = '10.9.0.1'
>>> b = TCP()
>>> b.dport = 23
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
```

Sniffing 检测结果：

```
root@VM:/volumes# python3 sniffer.py
type2
###[ Ethernet ]###
  dst       = 02:42:bb:2f:e7:e8
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x66b8
     src       = 10.9.0.5
```

在进行 type3 筛选时，注释掉 pkt1、pkt2 行的代码，构造响应报文:

```
>>> a = IP()
>>> a.src = '128.230.1.1'
>>> a.dst = '10.9.0.1'
>>> send(a)
.
Sent 1 packets.
```

Sniffing 检测结果：

```
root@VM:/volumes# python3 sniffer.py
type3
###[ Ethernet ]###
  dst       = 02:42:bb:2f:e7:e8
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 20
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = hopopt
     chksum    = 0xeef8
     src       = 128.230.1.1
     dst       = 10.9.0.1
```

Task1.2：

Spoofing 代码如下

```
>>> from scapy.all import *
>>> a = IP()
>>> a.src = '10.9.0.2'
>>> b = ICMP()
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
>>>
```

Task1.3：

Traceroute 代码如下：

```
from scapy.all import *


def traceroute(dst_ip):
    flag = 0
    for i in range(30):
        a = IP()
        a.dst = dst_ip
        a.ttl = i
        b = ICMP()
        re = sr1(a/b)
        re_ip = re.getlayer(IP).src

        print('%2d  %15s'%(i,re_ip))


        if re_ip == dst_ip:
            flag = 1
            break
    if flag == 0:
        print('Not Found')
    else:
        print('Traceroute over')

traceroute('10.9.0.5')
~
~
```

经过一跳即可到达目的 IP，运行截图如下：

```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing
 and Spoofing Lab/Labsetup/volumes# python3 Traceroute.py
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
 0         10.9.0.5
Traceroute over
```

Task1.4：

Sniffing_then_Spoofing 代码如下：

```
from scapy.all import *

def spoof(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("src",pkt[IP].src)
        print("dst",pkt[IP].dst)
        a = IP()
        a.src = pkt[IP].dst
        a.dst = pkt[IP].src
        a.ihl = pkt[IP].ihl
        b = ICMP()
        b.type = 0
        b.id = pkt[ICMP].id
        b.seq = pkt[ICMP].seq
        c = pkt[Raw].load
        p = a/b/c
        send(p,verbose=0)


pkt = sniff(filter='icmp', prn=spoof)
```

对 1.2.3.4 进行 ping，可以 ping 通，因为报文经过攻击者，攻击者检测到 ping 的 ICMP 报文后返回了一个报文，从而使 Host 误认为可以 ping 通。运行结果如下图：

```
root@0b0e28f4ec98:/# ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Net Unreachable
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=56.7 ms
From 10.9.0.1 icmp_seq=2 Destination Net Unreachable
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=16.9 ms
From 10.9.0.1 icmp_seq=3 Destination Net Unreachable
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=22.5 ms
From 10.9.0.1 icmp_seq=4 Destination Net Unreachable
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=16.8 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=22.8 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=21.0 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=23.9 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=22.2 ms
```

对 10.9.0.99 进行 ping，不可以 ping 通，因为这个 IP 在内网内，不必经过攻击者，因而不会接受到 spoofing 的报文，而且这个 IP 在内网中不存在的，因此 ping 不通。运行结果如下图：

```
root@0b0e28f4ec98:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
From 10.9.0.5 icmp_seq=4 Destination Host Unreachable
From 10.9.0.5 icmp_seq=5 Destination Host Unreachable
From 10.9.0.5 icmp_seq=6 Destination Host Unreachable
From 10.9.0.5 icmp_seq=7 Destination Host Unreachable
From 10.9.0.5 icmp_seq=8 Destination Host Unreachable
From 10.9.0.5 icmp_seq=9 Destination Host Unreachable
From 10.9.0.5 icmp_seq=10 Destination Host Unreachable
```

对 8.8.8.8 进行 ping，可以 ping 通，因为报文出内网时会通过攻击者，而攻击者检测到 ping 的 ICMP 报文后返回了一个报文，从而使 Host 误认为可以 ping 通。运行结果如下图：

```
root@0b0e28f4ec98:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Net Unreachable
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=54.4 ms
From 10.9.0.1 icmp_seq=2 Destination Net Unreachable
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=22.6 ms
From 10.9.0.1 icmp_seq=3 Destination Net Unreachable
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=17.7 ms
```