

## 第一次实验

57118139 顾宸玮

### Task 1.1: Sniffing Packets

A. In the above program, for each captured packet, the callback function `print_pkt()` will be invoked; this function will print out some of the information about the packet. Run the program with the root privilege and demonstrate that you can indeed capture packets. After that, run the program again, but without using the root privilege; describe and explain your observations.

Python 程序如下，经过使用 `ifconfig | grep br` 的命令，发现 hash 值为

b558b6d968f1

```
1#!/usr/bin/env python3
2from scapy.all import *
3def print_pkt(pkt):
4    pkt.show()
5pkt = sniff(iface='br-
b558b6d968f1', filter='icmp', prn=print_pkt)
```

在 Ping 10.9.0.5 后，发现程序输出如下：

```
root@VM:/home/seed/Desktop# ./mycode.py
####[ Ethernet ]####
  dst      = 02:42:0a:09:00:05
  src      = 02:42:06:d0:77:35
  type     = IPv4
####[ IP ]####
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 59570
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x3ddf
  src      = 10.9.0.1
  dst      = 10.9.0.5
  \options \
####[ ICMP ]####
```



number 23.

Python 代码如下，send 一个 tcp 包就可以输出信息

```
1#!/usr/bin/python3
2from scapy.all import *
3def print_pkt(pkt):
4    pkt.show()
5pkt=sniff(iface='br-b558b6d968f1',filter='tcp and
    src host 10.9.0.1 and dst port 23',prn=print_pkt)
```

Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

Python 代码如下，send 一个符合网段的包就可以输出结果

```
1#!/usr/bin/python3
2from scapy.all import *
3def print_pkt(pkt):
4    pkt.show()
5pkt=sniff(iface='br-b558b6d968f1',filter='net
    128.230.0.0/16',prn=print_pkt)
```

## Task 1.2: Spoofing ICMP Packets

Please make any necessary change to the sample code, and then demonstrate that you can spoof an ICMP echo request packet with an arbitrary source IP address.

按照实验手册做如下操作：

```
[07/05/21]seed@VM:~/Desktop$ sudo su
root@VM:/home/seed/Desktop# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more
>>> from scapy.all import *
>>> a=IP()
>>> a.dst='10.0.9.5'
>>> b=ICMP()
>>> p=a/b
>>> send(p)
.
Sent 1 packets.
>>>
```

```
>>> ls(a)
version      : BitField (4 bits)      = 4      (4)
ihl          : BitField (4 bits)      = None    (None)
tos          : XByteField              = 0      (0)
len          : ShortField              = None    (None)
id           : ShortField              = 1      (1)
flags        : FlagsField (3 bits)     = <Flag 0 (>) (<Flag 0 (>))
frag         : BitField (13 bits)      = 0      (0)
ttl          : ByteField               = 64     (64)
proto        : ByteEnumField           = 0      (0)
chksum       : XShortField             = None    (None)
src          : SourceIPField           = '0.0.0.0' (None)
dst          : DestIPField             = '10.0.9.5' (None)
options      : PacketListField         = []     ([])
>>>
```

首先进行窥探，发送一个有特定源目的地址的 ICMP 包

```
Open  [ ]  spoofing.py  Save  [ ]  [ ]  [ ]
~ / Desktop

1#!/usr/bin/python3
2from scapy.all import *
3ip=IP()
4ip.dst='10.9.0.5'
5ip.src='1.2.3.4'
6icmp=ICMP()
7pkt=ip/icmp
8send(pkt)

root@VM:/home/seed/Desktop# ./spoofing.py
Sent 1 packets.
```

因为在上述操作中执行了两次，在下面嗅探的时候就收到了三个包，分别是发送的两个包和一次回应的包

```
root@VM:/home/seed/Desktop# tcpdump -w /tmp/packets -v icmp
tcpdump: listening on br-b558b6d968f1, link-type EN10MB (Ethernet), capture size 262144 bytes
Got 3
```

### Task 1.3: Traceroute

**The objective of this task is to use Scapy to estimate the distance, in terms of number of routers, between your VM and a selected destination. This is basically what is implemented by the traceroute tool. In this task, we will write our own tool. The idea is quite straightforward: just send an packet (any type) to the destination, with its Time-To-Live (TTL) field set to 1 first. This packet will be dropped by the first router, which will send us an ICMP error message, telling us that the time-to-live has exceeded. That is how we get the IP address of the first**

router. We then increase our TTL field to 2, send out another packet, and get the IP address of the second router. We will repeat this procedure until our packet finally reach the destination. It should be noted that this experiment only gets an estimated result, because in theory, not all these packets take the same route (but in practice, they may within a short period of time). The code in the following shows one round in the procedure.

每次增加 TTL 的代码如下：

```
#!/usr/bin/python3
from scapy.all import *
import sys
argulist=sys.argv
if len(argulist)!=2:
    print("Usage: sudo ./trace.py [hostname]")
    exit()
MAX_TTL=255
dsthostname=argulist[1];
dstIP=socket.gethostbyname(argulist[1]);
print("trace.py to "+dsthostname+" (" +dstIP+"),255 hops max");
ip=IP()
ip.dst=dstIP
ip.ttl=1
icmp=ICMP()
while ip.ttl<=MAX_TTL:
    reply=sr1(ip/icmp,verbose=0,timeout=2)
    if(reply==None):
        print(str(ip.ttl)+"\t* * *")
        ip.ttl+=1
        continue
    print(str(ip.ttl)+"\t"+reply.src)
    if(reply.src==dstIP):
        break
    ip.ttl+=1
```

进行测试后，在后面测试了百度的地址，可以发现 trace 结果如下：

```

[07/05/21]seed@VM:~/Desktop$ sudo su
root@VM:/home/seed/Desktop# ./trace.py www.baidu.com
trace.py to www.baidu.com (117.185.17.144),255 hops ma
x
1      172.20.10.1
2      * * *
3      10.136.112.29
4      * * *
5      218.206.124.153
6      221.183.47.69
7      * * *
8      221.183.51.210
9      117.185.123.54
10     * * *
11     117.185.17.144
root@VM:/home/seed/Desktop# █

```

然后尝试 trace1.2.3.4，结果发现到了第 8 次就后面没有结果了，可能是在这一跳时包被路由丢弃了

```

^Z
[5]+  Stopped                  sudo ./sni
spo.py
[07/06/21]seed@VM:~/Desktop$ sudo ./trac
e.py 1.2.3.4
trace.py to 1.2.3.4 (1.2.3.4),255 hops m
ax
1      172.20.10.1
2      * * *
3      10.136.121.154
4      * * *
5      * * *
6      183.207.30.137
7      111.24.6.33
8      111.24.16.202
9      * * *
10     * * *
11     * * *
12     * * *

```

## Task 1.4: Sniffing and-then Spoofing

**In your experiment, you should ping the following three IP addresses from the user container. Report your observation and explain the results.**

代码如下：

```

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

```



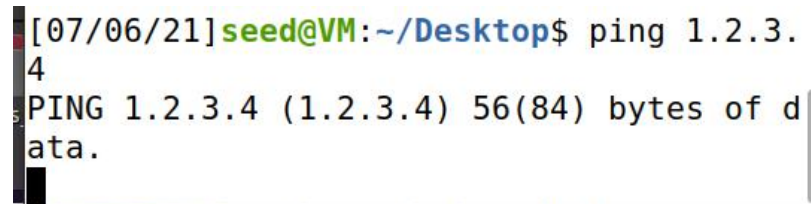
```

def spoof_pkt(pkt):
    if pkt[ICMP].type!=8:
        return
    ip=IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)
    icmp=ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
    data=pkt[Raw].load
    newpkt=ip/icmp/data
    send(newpkt,verbose=0)
    print("send spoofed packet\n")
while(1):
    pkt=sniff(filter='icmp',prn=spoof_pkt)

```

一旦捕获了一个 echo 请求包，程序应该立即 spoof 一个 ICMP echo 应答包，将第一个包的源设置为 spoof 包的目的地，并将第一个包的目的地设置为 spoof 包的源。

1.在未运行代码时发现无法 ping 通 1.2.3.4

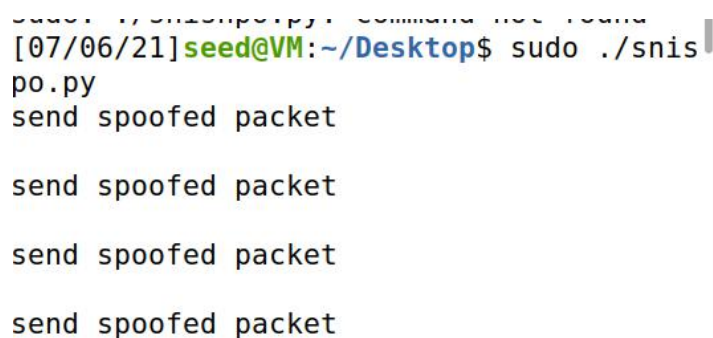


```

[07/06/21] seed@VM:~/Desktop$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.

```

在运行代码后，可以发现代码发送了数据包并且可以 ping 通 1.2.3.4



```

[07/06/21] seed@VM:~/Desktop$ sudo ./snis
po.py
send spoofed packet

send spoofed packet

send spoofed packet

send spoofed packet

```

```
[07/06/21]seed@VM:~/Desktop$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=18.8 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=20.1 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=22.9 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=16.5 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=16.2 ms
```

2.无法 ping 通 10.9.0.99, 即使运行了代码以后也无法 ping 通 10.9.0.99, 并且代码也没有发送数据包

```
[07/06/21]seed@VM:~/Desktop$ ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Host Unreachable
From 10.9.0.1 icmp_seq=2 Destination Host Unreachable
```

3.在未运行程序时, ping 8.8.8.8 是可以通的, 但是在运行代码后, 会出现 (DUP!),

```
[07/06/21]seed@VM:~/Desktop$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=109 time=103 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=101 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=93.6 ms
```

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.	[4]+ Stopped sudo ./spo.py
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=17.2 ms	[07/06/21]seed@VM:~/Desktop\$ sudo ./spo.py
64 bytes from 8.8.8.8: icmp_seq=1 ttl=109 time=108 ms (DUP!)	send spoofed packet
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=20.3 ms	send spoofed packet
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=84.8 ms (DUP!)	send spoofed packet
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=24.3 ms	^Z
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=102 ms (DUP!)	[5]+ Stopped sudo ./spo.py
	[07/06/21]seed@VM:~/Desktop\$



## 实验总结

本次实验中，我了解到了如何欺骗和嗅探一个数据包，实验过程中也遇到了很多问题，比如 python 使用的熟练以及 scapy 原理的不熟悉，但是最终还是学习到了很多新知识。