

The report of lab 1

57118117 湛雨阳

Task 1.1A.

Code:

(1) icmp 嗅探程序

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(iface='br-c16868942eaf', filter='icmp', prn=print_pkt)
```

Result:

(1) 使用 root 用户特权运行 sniff 脚本结果如下：（正常运行）

```
###[ Ethernet ]###
  dst      = 02:42:0a:09:00:05
  src      = 02:42:61:a3:e6:73
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 23597
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0xca64
  src      = 10.9.0.1
  dst      = 10.9.0.5
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x44d0
  id       = 0x6
  seq      = 0x9
###[ Raw ]###
  load     = '\x0b.\xe3\x00\x00\x00\x00\x01\xbf\x04\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567'
```

```

####[ Ethernet ]###
  dst      = 02:42:61:a3:e6:73
  src      = 02:42:0a:09:00:05
  type     = IPv4
####[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 10407
  flags    =
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x3deb
  src      = 10.9.0.5
  dst      = 10.9.0.1
  \options \
####[ ICMP ]###
  type     = echo-reply
  code     = 0
  chksum   = 0x4cd0
  id       = 0x6
  seq      = 0x9
####[ Raw ]###
  load     = '\x0b.\xe3'\x00\x00\x00\x00\x01\xbf\x04\x00\x00\x00\x00\x
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,
-./01234567'

```

(2) 切换到 seed 用户后执行脚本结果如下：(报错)

```

root@VM:/volumes# su seed
seed@VM:/volumes$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 7, in <module>
    pkt = sniff(iface='br-cl6868942eaf', 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, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)) # 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

```

Answer to the questions:

在切换至 seed 用户，不使用 root 用户的权限运行 sniffer.py 时，显示 operation not permitted，根据报错可以了解到，报错中提到的函数需要更高级别的权限才能执行。

Task 1.1B.

Code:

1、仅捕获 ICMP 报文代码如下（与 Task1.1A 一致）

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(iface='br-c16868942eaf', filter='icmp', prn=print_pkt)
```

2、捕获来自特定 IP 地址的，且目的端口为 23 的 TCP 报文：

(1) sniff 程序

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(iface='br-c16868942eaf', filter='tcp and src host 10.0.2.15 and dst port 23',
    prn=print_pkt)
```

(2) 包发送程序

```
1from scapy.all import *
2
3ip = IP()
4ip.src = '10.9.0.1'
5ip.dst = '10.9.0.5'
6tcp = TCP()
7tcp.dport = 23
8p = ip/tcp
9send(p)
```

3、捕获发送到指定子网 128.230.0.0/16 中的数据包：

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(iface='br-c16868942eaf', filter='dst net 128.230.0.0/16', prn=print_pkt)
```

Result:

1、仅捕获 ICMP 报文结果如下（与 Task1.1A 一致）:

```
###[ Ethernet ]###
  dst      = 02:42:0a:09:00:05
  src      = 02:42:61:a3:e6:73
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 23597
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0xca64
  src      = 10.9.0.1
  dst      = 10.9.0.5
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x44d0
  id       = 0x6
  seq      = 0x9
###[ Raw ]###
  load     = '\x0b.\xe3'\x00\x00\x00\x00\x01\xbf\x04\x00\x00\x00\x00\x
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,
-./01234567'
```

```
###[ Ethernet ]###
  dst      = 02:42:61:a3:e6:73
  src      = 02:42:0a:09:00:05
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 10407
  flags    =
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x3deb
  src      = 10.9.0.5
  dst      = 10.9.0.1
  \options \
###[ ICMP ]###
  type     = echo-reply
  code     = 0
  chksum   = 0x4cd0
  id       = 0x6
  seq      = 0x9
###[ Raw ]###
  load     = '\x0b.\xe3'\x00\x00\x00\x00\x01\xbf\x04\x00\x00\x00\x00\x
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,
-./01234567'
```

2、捕获来自特定 IP 地址的，且目的端口为 23 的 TCP 报文结果如下：

```
###[ Ethernet ]###
  dst      = 02:42:0a:09:00:05
  src      = 02:42:61:a3:e6:73
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 40
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = tcp
  chksum   = 0x66b8
  src      = 10.9.0.1
  dst      = 10.9.0.5
  \options \
###[ TCP ]###
  sport    = ftp_data
  dport    = telnet
  seq      = 0
  ack      = 0
  dataofs  = 5
  reserved = 0
  flags    = S
  window   = 8192
  chksum   = 0x7ba0
  urgptr   = 0
  options  = []
```

3、捕获发送到指定子网 128.230.0.0/16 中的数据包结果如下：

```
###[ Ethernet ]###
  dst      = 02:42:1c:42:19:90
  src      = 02:42:0a:09:00:05
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 56653
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0xd266
  src      = 10.9.0.5
  dst      = 128.230.0.1
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x4af
  id       = 0x10
  seq      = 0x1
###[ Raw ]###
  load     = '\xcdF\xe3'\x00\x00\x00\x00\x81\xc5\x02\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567'
```

Task 1.2.

Code:

设置虚假地址为 192.168.1.1，构造包代码如下：

```
1 from scapy.all import *
2
3 ip = IP()
4 ip.src = '192.168.1.1'
5 ip.dst = '10.9.0.5'
6 icmp = ICMP()
7 p = ip/icmp
8 send(p)
```

Result:

使用 ICMP 嗅探程序查看更改包结果，可以看到源地址已被更改。

```
###[ Ethernet ]###
  dst      = 02:42:0a:09:00:05
  src      = 02:42:1c:42:19:90
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 28
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = icmp
  checksum = 0xaf29
  src      = 192.168.1.1
  dst      = 10.9.0.5
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  checksum = 0xf7ff
  id       = 0x0
  seq      = 0x0
```

Task 1.3.

Code:

设定 ttl 从 1 到 20，访问 baidu.com，代码如下：

```
1 from scapy.all import *
2
3 ans,unans=sr(IP(dst='www.baidu.com', ttl=(1,20))/TCP(flags=0x2))
4 for snd,rcv in ans:
5     print(snd.ttl, rcv.src, isinstance(rcv.payload, TCP))
```

Result:

脚本运行结果如下，可以看到经过 5 跳后成功访问目标网站：

```
root@VM:/volumes# python3 sender.py
Begin emission:
Finished sending 20 packets.
.*****.....^C
Received 38 packets, got 14 answers, remaining 6 packets
1 192.168.43.1 False
2 58.240.96.10 False
3 182.61.216.0 False
4 221.6.2.173 False
5 112.80.248.76 True
6 112.80.248.76 True
7 112.80.248.76 True
8 112.80.248.76 True
9 112.80.248.76 True
10 112.80.248.76 True
11 112.80.248.76 True
12 112.80.248.76 True
13 112.80.248.76 True
14 112.80.248.76 True
```

Task 1.4.

Code:

获取并构造包过程的代码:

```
1 from scapy.all import *
2
3 def sniff_sproof(pkt):
4     if pkt[ICMP].type == 8:
5         ip = IP(src=pkt[IP].dst,dst=pkt[IP].src)
6         icmp = ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
7         data = pkt[Raw].load
8         p = ip/icmp/data
9         send(p)
10 pkt = sniff(iface='br-429f5989ca75',filter='icmp',prn=sniff_sproof)
11
```

Result:

1、ping 1.2.3.4 结果如下,可以发现本应该不存在所以 ping 不通的地址可以成功 ping 通:

```
root@8ecd56eebb63:/# 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=68.2 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=16.4 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=17.9 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=17.0 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=25.1 ms
```

2、ping 10.9.0.99 结果如下,该地址在 LAN 上不可达:

```
root@8ecd56eebb63:/# 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
```


3、ping 8.8.8.8 该 IP 地址正常存在，可以 ping 通，且有冗余包产生：

```
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=110 time=49.5 ms  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=78.4 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=16.8 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=48.6 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=14.7 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=49.4 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=17.2 ms  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=110 time=50.4 ms (DUP!)
```

如图，出现了 duplicated 数据包提示，说明正常返回的包和伪造包混在一起。

Answer to questions:

Ping 不存在地址 1.2.3.4 时，attack 机能够捕获到 icmp-echo-request 请求包，并根据其包内容伪造出 reply 包，因此可以让本应该 ping 不通的地址显示成 ping 通。

对于内网中不存在的地址 10.9.0.5，无论在开启欺骗或是不开启的情况下，都无法 ping 通。

而对于本来就存在的地址 8.8.8.8，通过上面所贴的代码得到的结果同样是 ping 通。但会出现 DUP! 情况，说明正常 ping 返回的包和脚本伪造的包都被接收了，出现了冗余。