**lab[1]-report**

57118114 蔡欣明

**Task 1.1: Sniffing Packets**

**Task 1.1A**

利用ifconfig可知，主机的IP地址为10.9.0.1，docker的IP地址为10.9.0.5。创建文件sniffer.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

def

print\_pkt

(

pkt

):

pkt

.

show

()

pkt

=

sniff

(

iface

=

'br-a5fdad393ebc'

,

filter

=

'icmp'

,

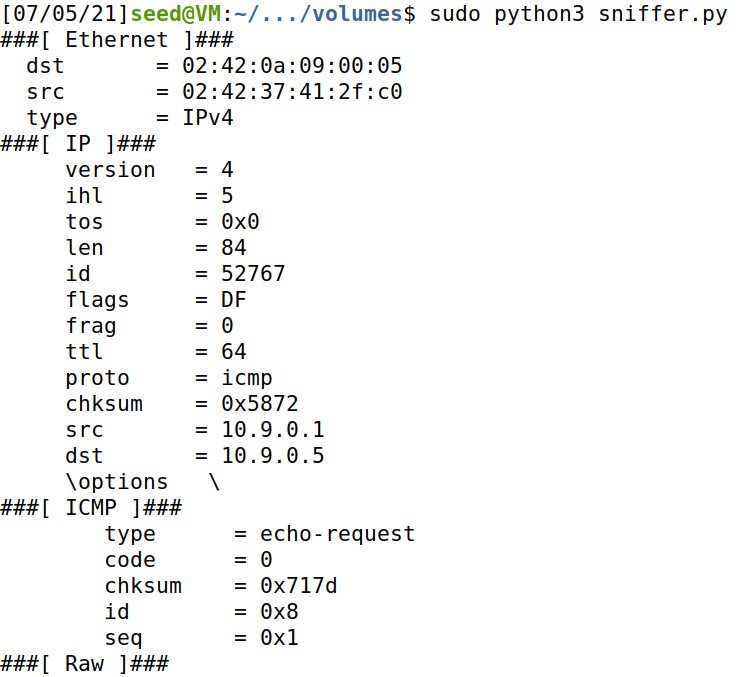
prn

=

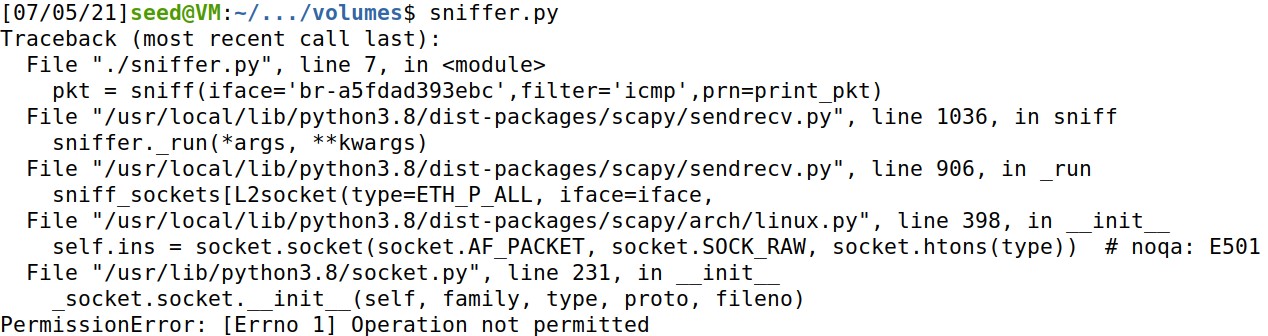
print\_pkt

)

利用root权限运行该程序后，在主机中ping连接docker的IP地址，得到结果如下，成功嗅探到icmp报文。



如果不使用root权限运行该程序，则会出现Operation not permitted的错误，原因是socket的调用需要更高的权限。



**Task 1.1B** 利用ifconfig可知，主机的IP地址为10.9.0.1，docker的IP地址为10.9.0.5。

创建文件sniffer.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

def

print\_pkt

(

pkt

):

pkt

.

show

()

pkt

=

sniff

(

iface

=

'br-08f60a398414'

,

filter

=

'icmp'

,

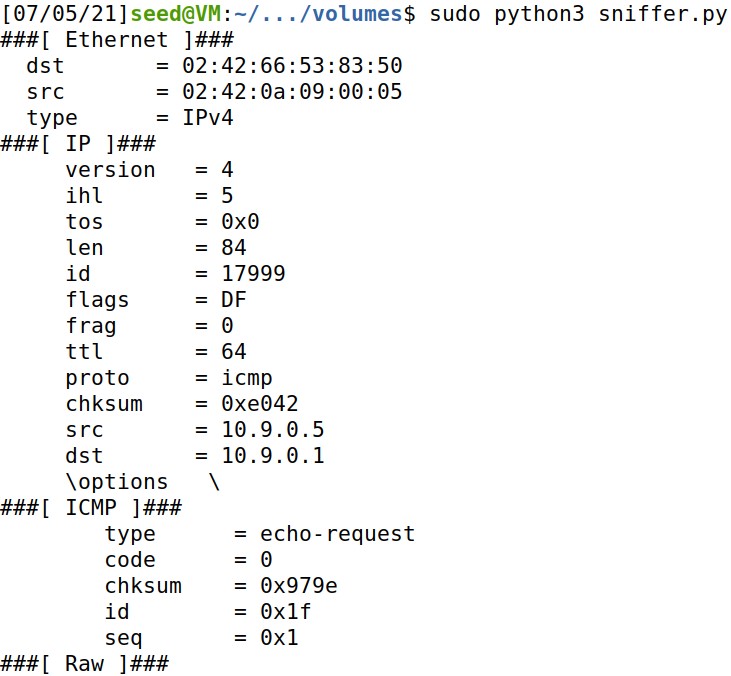
prn

=

print\_pkt

)

利用root权限运行该程序后，在docker中ping连接主机的IP地址，得到结果如下，成功嗅探到icmp报文。



创建文件tcp\_sniffer.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

def

print\_pkt

(

pkt

):

pkt

.

show

()

pkt

=

sniff

(

iface

=

"br-08f60a398414"

,

filter

=

'tcp and src host 10.9.0.5 and dst

port 23'

,

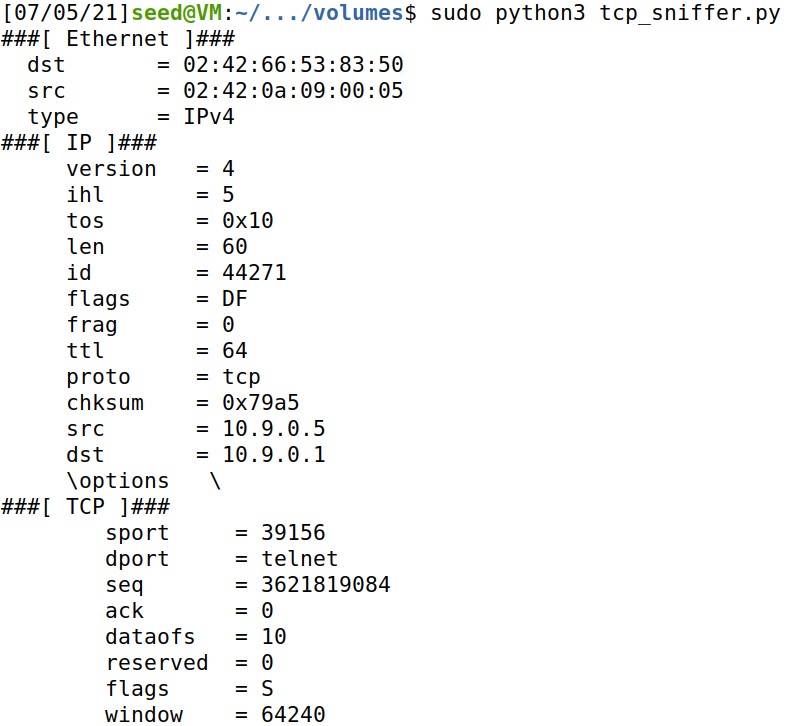
prn

=

print\_pkt

)

利用root权限运行该程序后，在docker中telnet连接主机的IP地址，得到结果如下，成功嗅探到满足条件的IP报文。



创建文件subnet\_sniffer.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

def

print\_pkt

(

pkt

):

pkt

.

show

()

pkt

=

sniff

(

iface

=

'br-08f60a398414'

,

filter

=

'dst net 10.8.0.0/24'

,

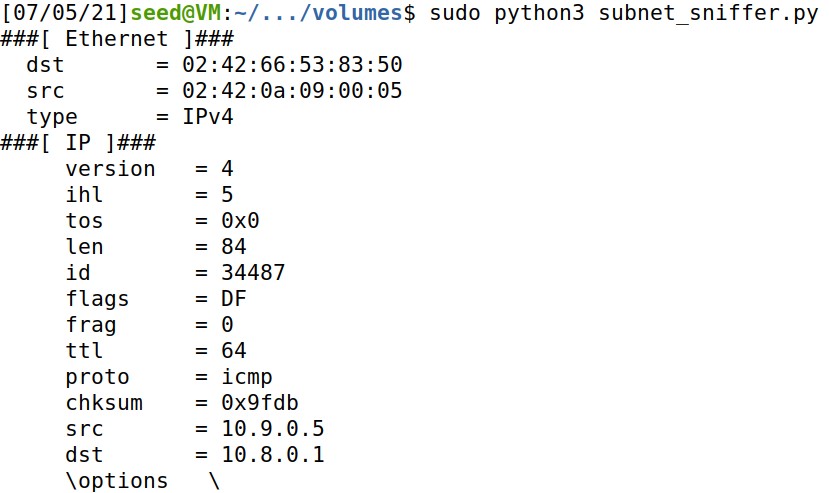
prn

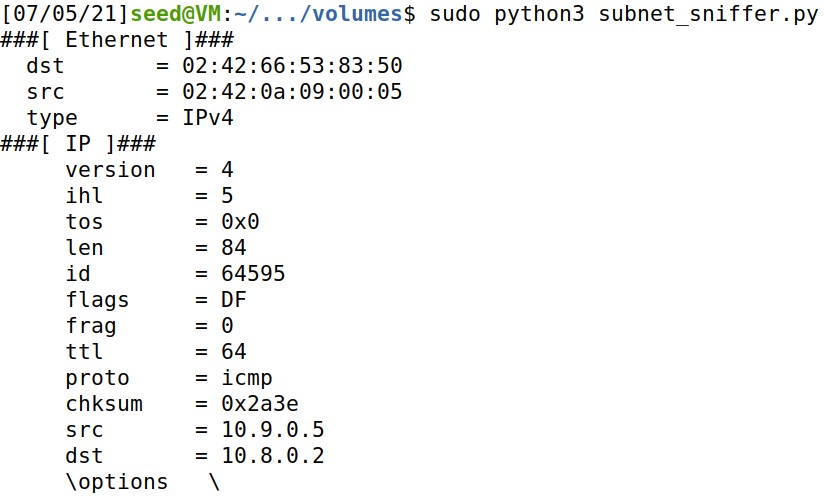
=

print\_pkt

)

利用root权限运行该程序后，在docker中ping连接子网的IP地址，得到结果如下，成功嗅探到满足条件的IP报文。





**Task 1.2: Spoofing ICMP Packets**

创建文件icmp\_spoof.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

a

=

IP

()

a

.

src

=

'10.8.0.1'

a

.

dst

=

'10.9.0.5'

b

=

ICMP

()

p

=

a

/

b

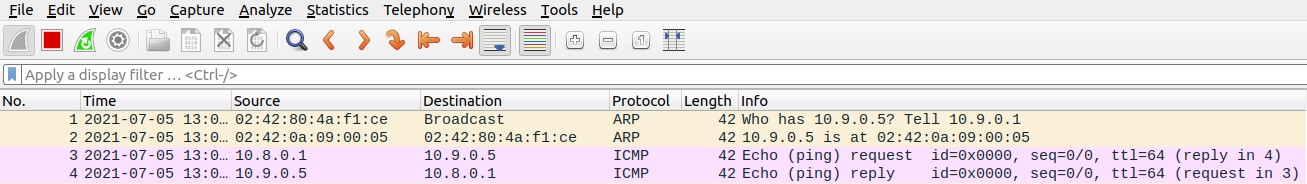
send

(

p

)

利用root权限运行该程序后，在wireshark中进行抓包，得到结果如下，成功伪造icmp报文。



**Task 1.3: Traceroute**

创建文件traceroute.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

res

=

''

for

i

in

range

(

1

,

30

):

pkt

=

IP

(

dst

=

'36.152.44.96'

,

ttl

=

i

)

/

ICMP

()

trace\_reply

=

sr1

(

pkt

,

timeout

=

1

)

if

trace\_reply

is

None

:

print

(

'Timeout'

)

else

:

src

=

trace\_reply

[

'IP'

].

src

ans

=

str

(

i

)

+

':'

+

src

print

(

ans

)

res

+=

ans

+

'\n'

if

src

==

'36.152.44.96'

:

print

(

res

)

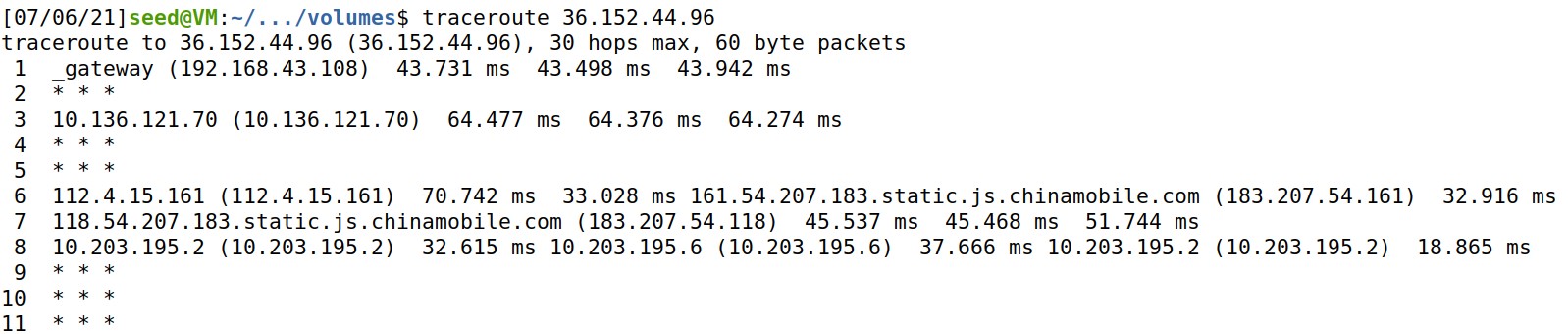
exit

(

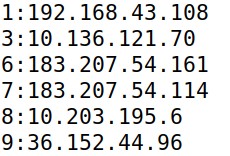
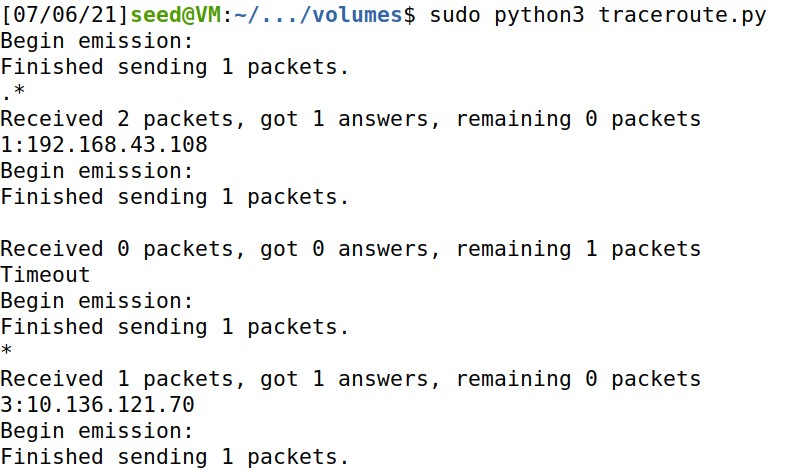
0

)

利用traceroute可知，报文经过的路径如下。



利用root权限运行该程序后，得到的结果与traceroute的结果基本一致，成功实现功能。



**Task 1.4: Sniffing and-then Spoofing** 创建文件sniff\_spoof.py，代码如下。

#!/usr/bin/env python3

from

scapy

.

all

import

\*

def

spoof\_pkt

(

pkt

):

a

=

IP

()

a

.

src

=

pkt

[

IP

].

dst

a

.

dst

=

pkt

[

IP

].

src

a

.

ihl

=

pkt

[

IP

].

ihl

b

=

ICMP

()

b

.

type

=

"echo-reply"

b

.

code

=

0

b

.

id

=

pkt

[

ICMP

].

id

b

.

seq

=

pkt

[

ICMP

].

seq

data

=

pkt

[

Raw

].

load

p

=

a

/

b

/

data

send

(

p

)

pkt

=

sniff

(

iface

=

'br-a1ccba9d103e'

,

filter

=

'icmp[icmptype] == icmp-echo'

,

prn

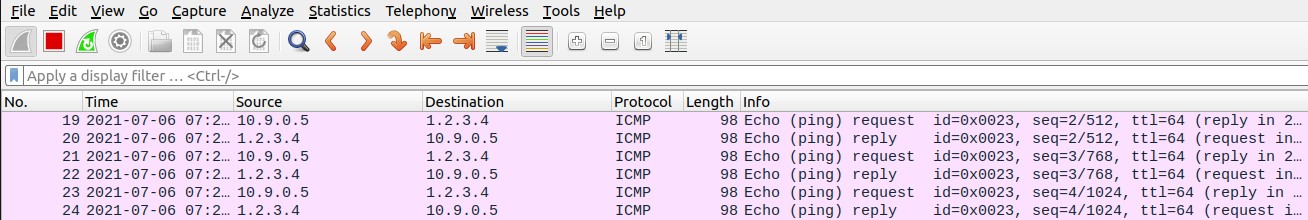
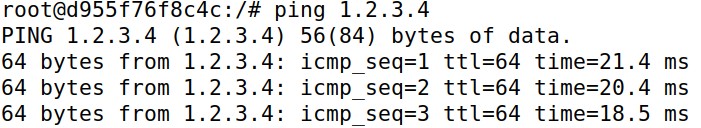
=

spoof\_pkt

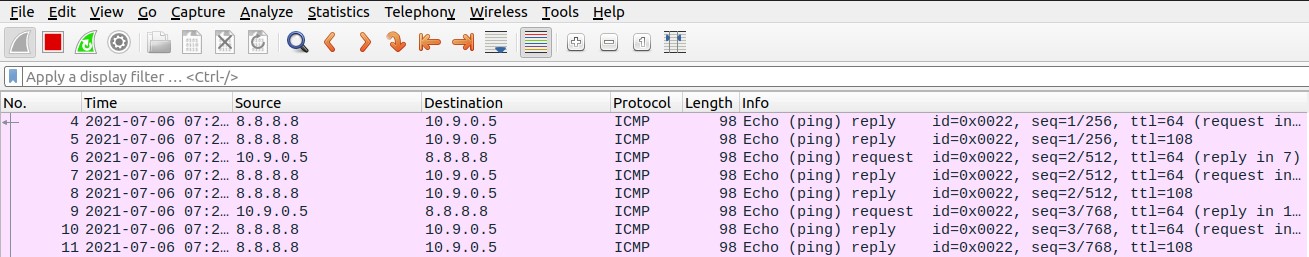
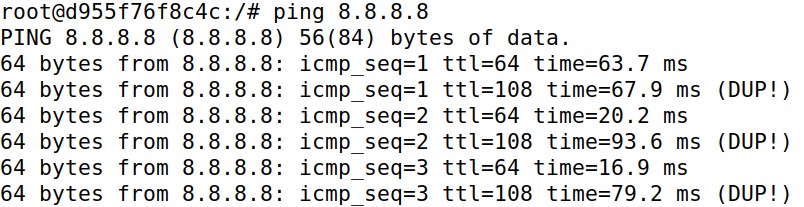
)

利用root权限运行该程序后，在docker中ping连接不同的IP地址，得到结果如下。

在ping连接1.2.3.4时，可知成功嗅探并伪造发送ICMP报文。由于该地址现实中并不存在，所以无法收到真正的ICMP报文，只能收到伪造的ICMP报文。



在ping连接8.8.8.8时，可知成功嗅探并伪造发送ICMP报文。由于该地址现实中存在，所以不仅收到真正的ICMP报文，而且也收到伪造的ICMP报文。



在ping连接10.9.0.99时，可知没有成功嗅探并伪造发送ICMP报文。由于该地址与主体处于同一个局域网内，利用ARP协议广播来寻找，但又因为该地址现实中不存在，所以不会产生ICMP报文。

