**实验1：Packet Snifﬁng and Spooﬁng Lab**

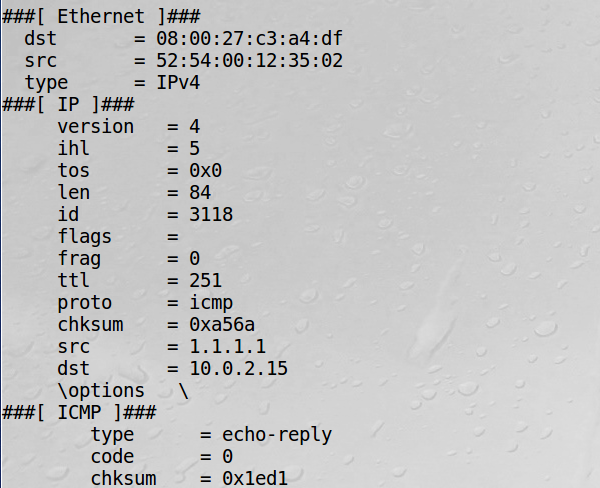
**Task 1: Using Tools to Sniff and Spoof Packets**

## Task 1.1：Sniffing Packets

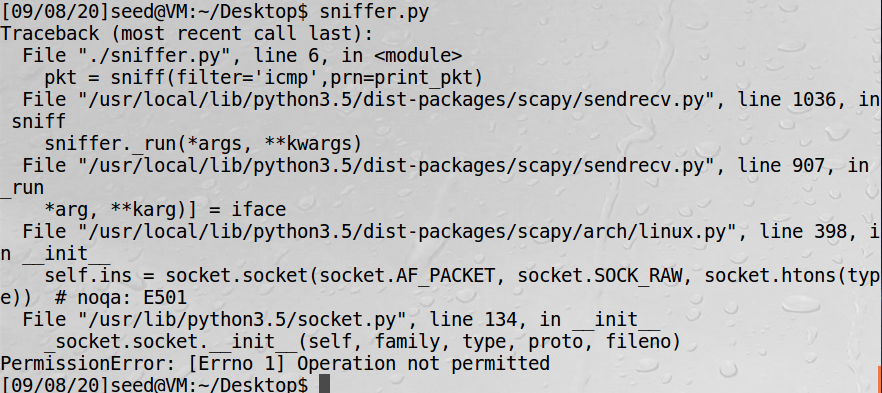
以root权限运行sniffer.py，因为未接收到任何数据包，程序会阻塞



打开一个新的控制台，使用ping命令，ping 1.1.1.1，在原控制台下会输出所捕获包的信息

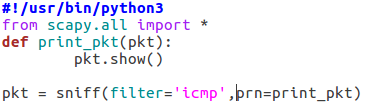


当不以root权限运行sniffer.py时，会出现操作不被允许的问题

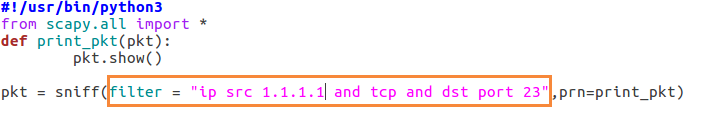


使用BPF对包进行过滤：

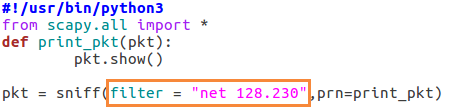
① 只捕获 ICMP 包



② 捕获来自特定IP，以及目的端口是23的任何TCP包

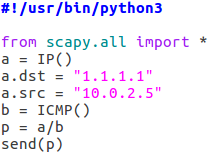


③ 捕获来自或到达特定子网的数据包。可以选择任何子网，例如128.230.0.0/16; 不应选择VM所连接的子网

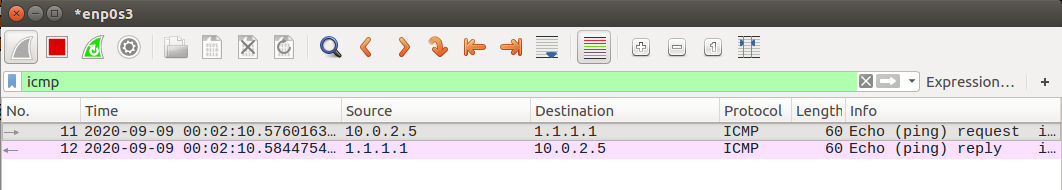


## Task 1.2：Spoofing ICMP Packets

在10.0.2.4的VM上运行如下代码，假装自己是10.0.2.5，向1.1.1.1发送ICMP请求报文



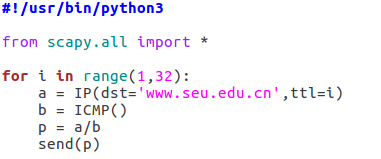
在10.0.2.5上用wireshark抓包，抓到了源地址为10.0.2.5的请求，以及由1.1.1.1发往10.0.2.5的响应，说明伪造成功



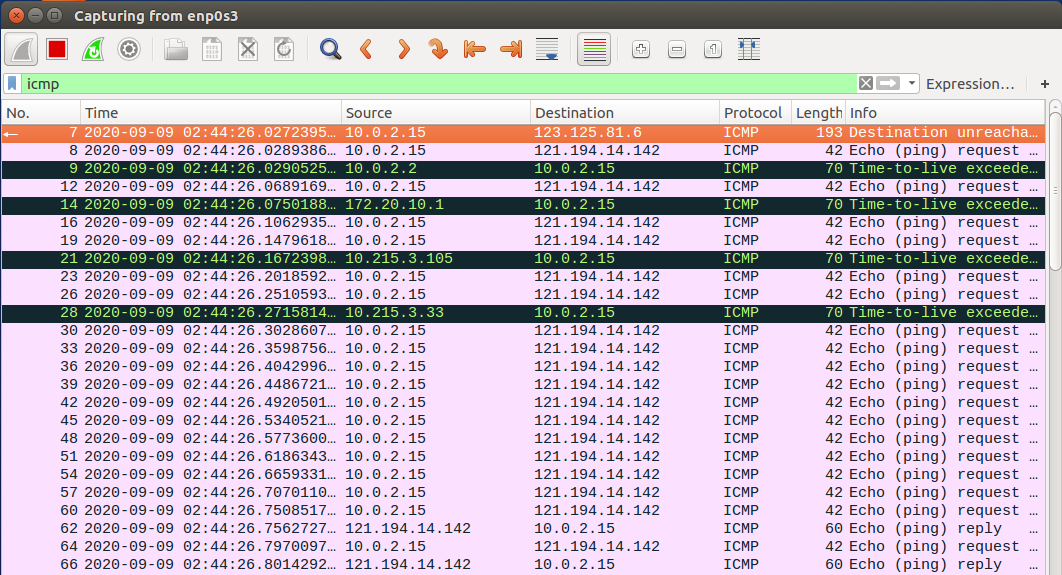
## Task 1.3：Traceroute

使用Scapy来估计VM与目标主机之间的距离（路由数量）

代码如下：

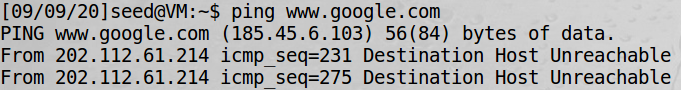


通过wireshark抓包的结果如下，可以看出中间会经过四个路由器

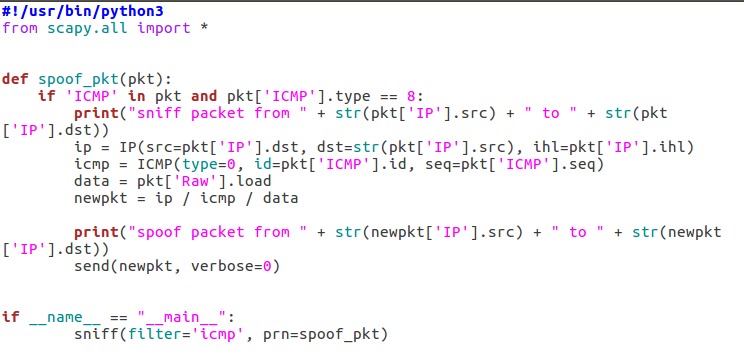


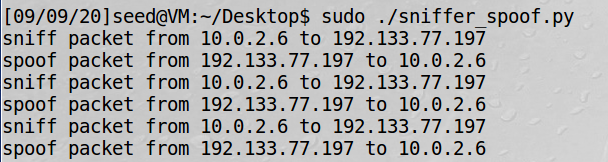
## Task 1.4：Sniffing and-then Spoofing

在10.0.2.6的VM上ping www.google.com是ping不通的

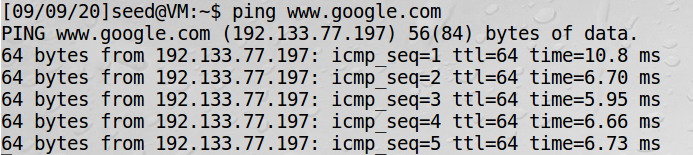


在10.0.2.5上运行如下代码





然后再在10.0.2.6上ping www.google.com，显示“可以”ping通



**实验2：ARP Cache Poisoning Attack Lab**

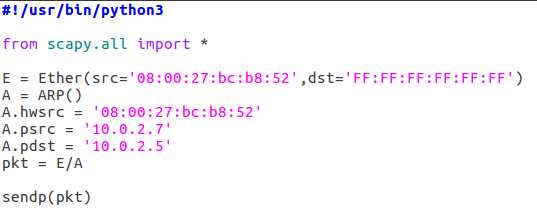
**Task 1: ARP Cache Poising**

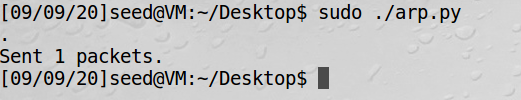
通过ARP 缓存中毒进行中间人攻击

|  |  |  |
| --- | --- | --- |
| M | 10.0.2.6 | 08:00:27:bc:b8:52 |
| A | 10.0.2.5 | 08:00:27:b1:3a:70 |
| B | 10.0.2.7 | 08:00:27:19:43:7d |

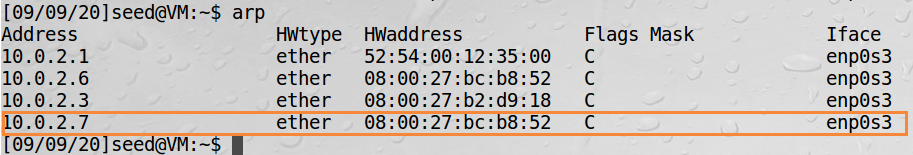
**Task 1A** (using ARP request). 在M上构造一个ARP请求包并发送给A，检查M的MAC地址在A的ARP缓存中是否映射到B的IP地址：

M上的请求包构造如下：



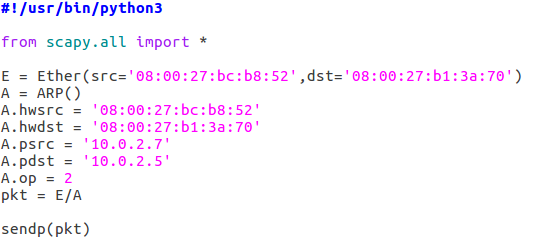


M发送的arp请求包中的源mac地址为M的mac地址，源ip地址为B的ip地址，目的ip地址为A的ip地址。A在收到M发出的arp请求后，会将请求中的ip地址与mac地址的对应关系写入A的arp缓存，因此可实现欺骗。



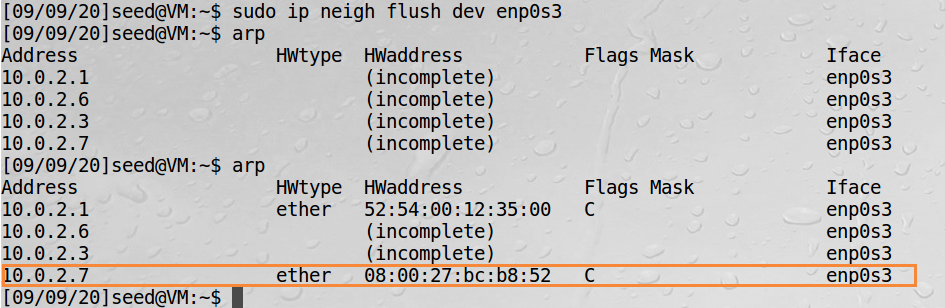
**Task 1B** (using ARP reply). 在主机M上构造一个ARP响应包并发送给A，检查M的MAC地址在A的ARP缓存中是否映射到B的IP地址：

主机M上的相应包构造如下：



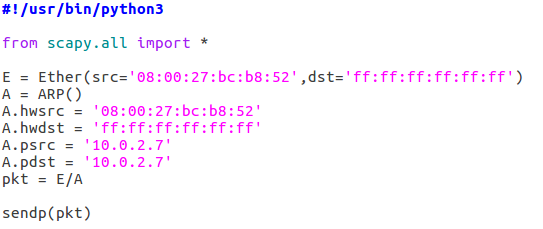
M发送的响应包中目的ip和目的mac地址都是A的，源mac地址为M的，源ip为B的，

从下图可以看出攻击成功

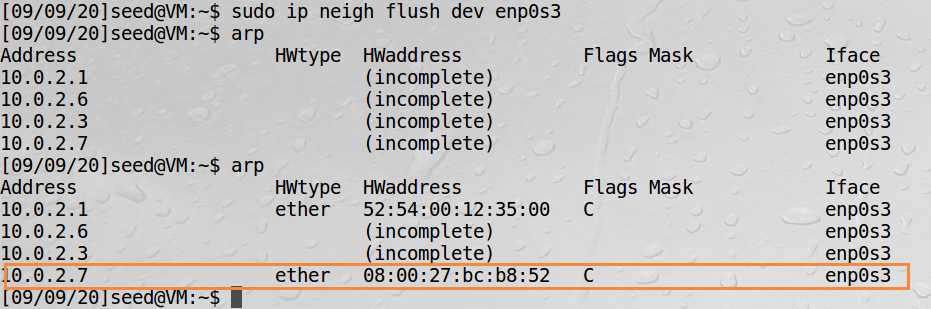


**Task 1C** (using ARP gratuitous message).

主机M上的相应包构造如下：



源ip和目的ip地址均为M要伪造的B的ip地址，源硬件地址为M的ip地址，目的硬件地址为广播地址，下图可以看出攻击成功

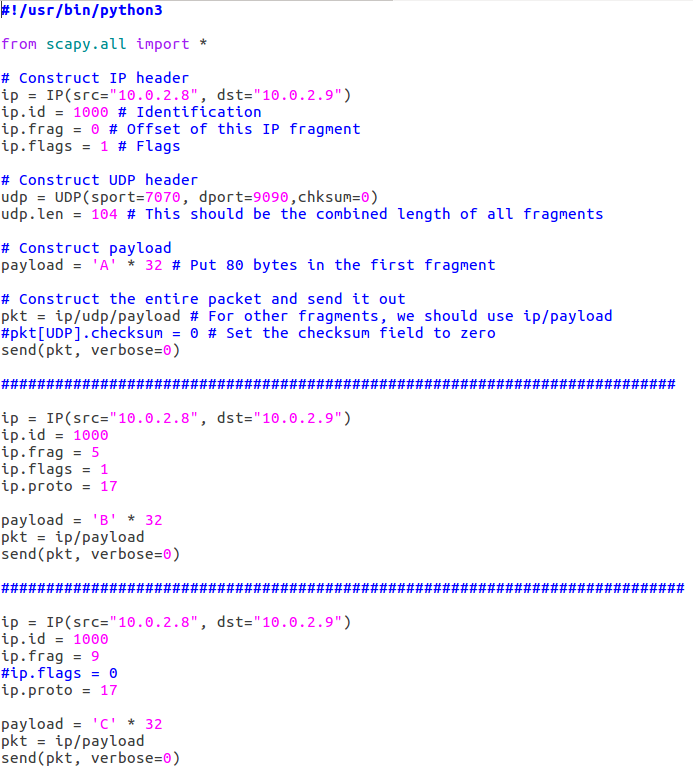


**实验3：IP/ICMP Attacks Lab**

**Task 1: IP Fragmentation**

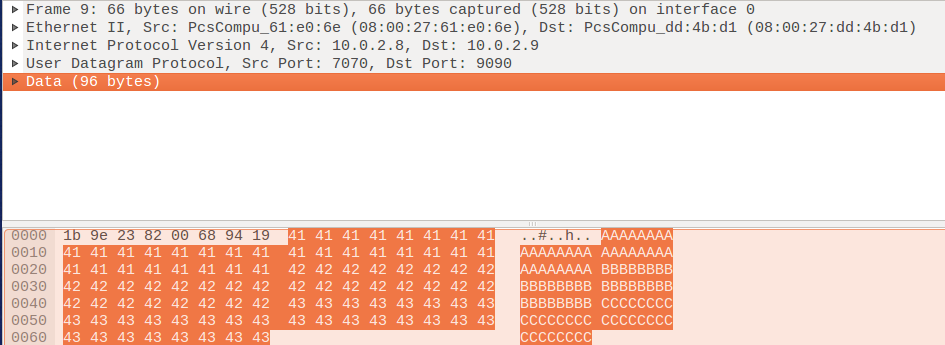
## Task 1.a：Conducting IP Fragmentation

三个数据包的构造如下图所示：

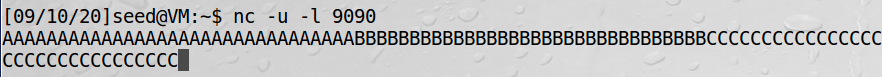


在10.0.2.8上运行上面的代码，在10.0.2.9进行观察，发现wireshark上得到了完整的数据包



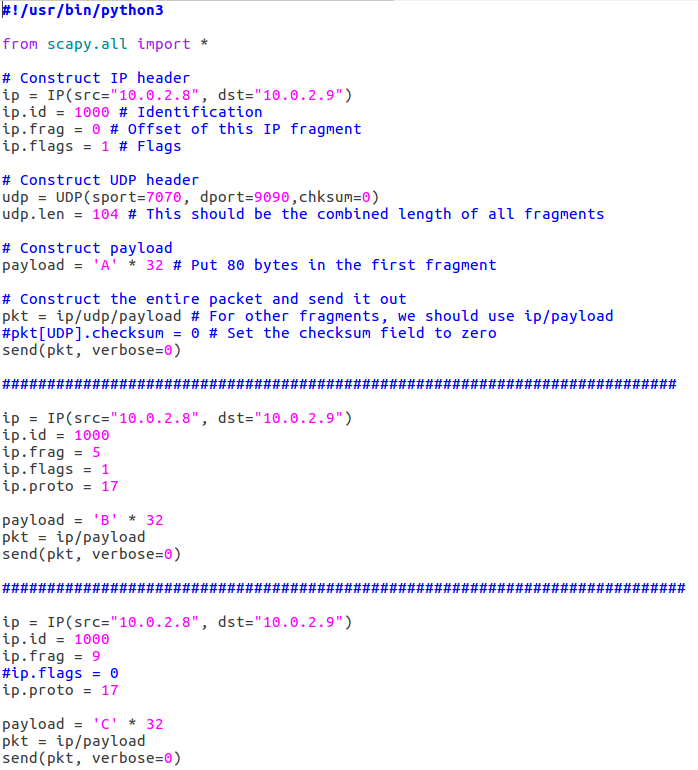


终端为

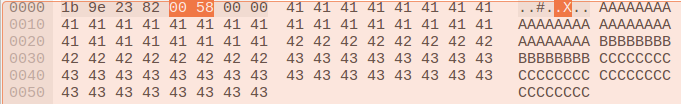


## Task 1.b：IP Fragments with Overlapping Contents

① 第一个分片的末尾和第二个分片的开始有16字节的重叠

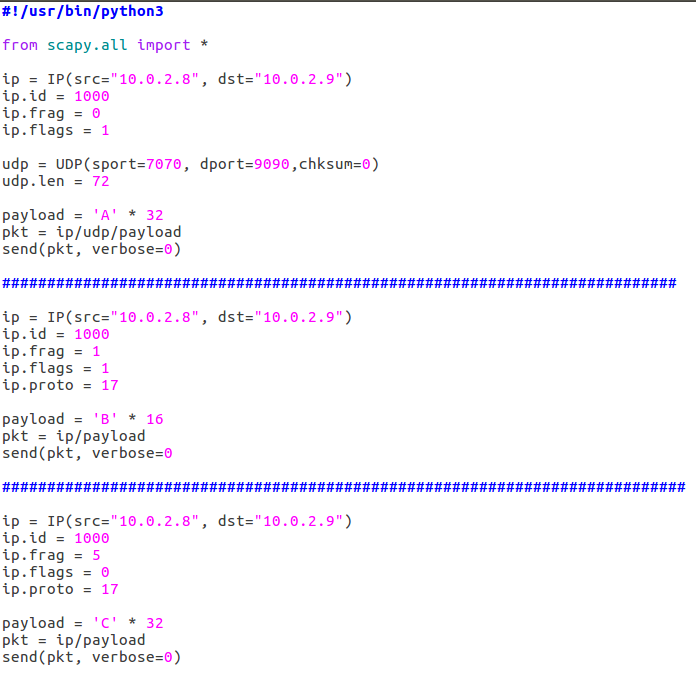


先发送第一个分片再发送第二个分片的结果：重叠的部分显示的是第一个分片的内容

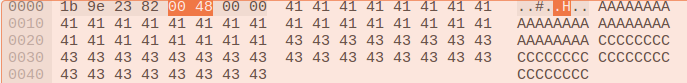


先发送第二个再发送第一个分片的结果与上一种情况相同

② 第二个分片完全被第一个分片包含



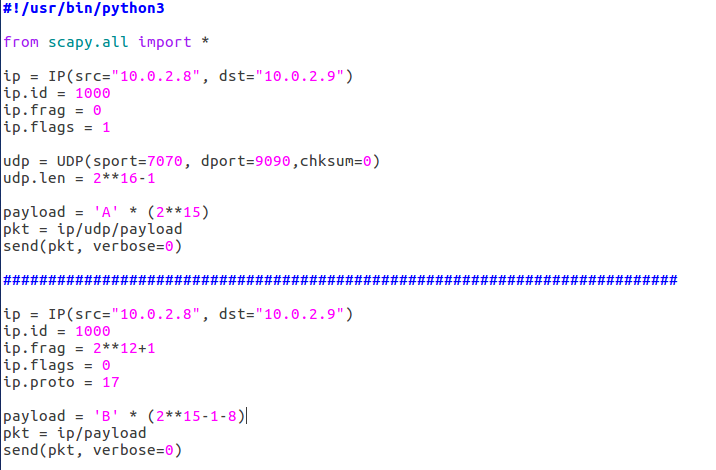
先发送第一个分片再发送第二个分片的结果：重叠的部分显示的是第一个分片的内容



先发送第二个再发送第一个分片的结果与上一种情况相同

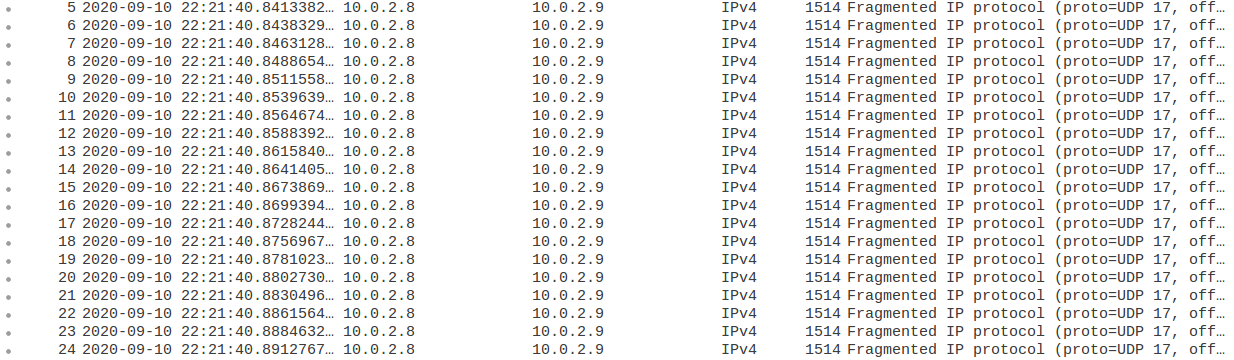
## Task 1.c：Sending a Super-Large Packet

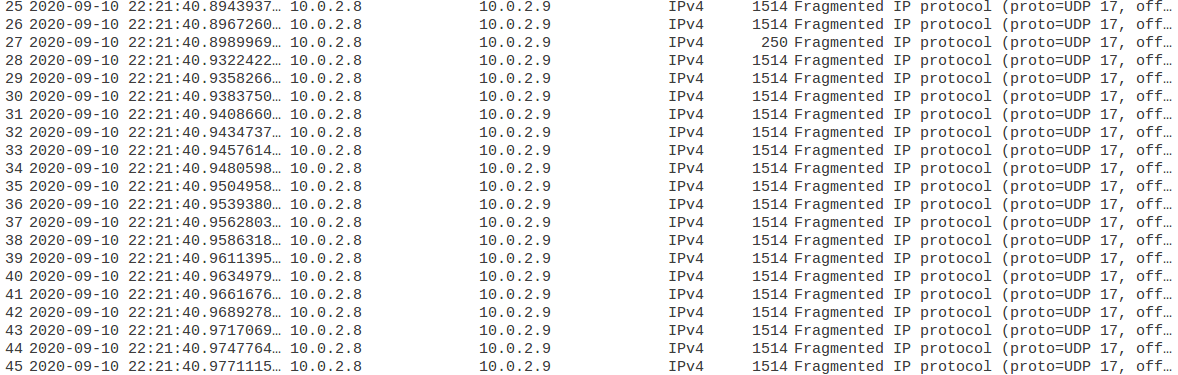
报文构造如下图

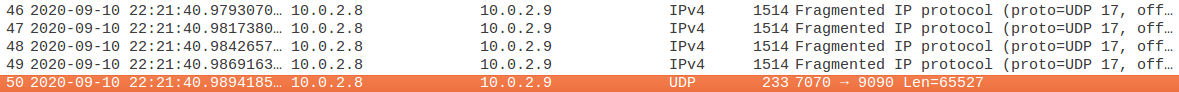


第一个分片ip负载部分的长度为8+2^15（udp头部+2^15数据），第二个分片ip负载部分的长度为2^15-1-8（数据），总的ip报文长度为2^16-1+20，超出最大长度2^16-1

在10.0.2.8发送后，在10.0.2.9抓取报文





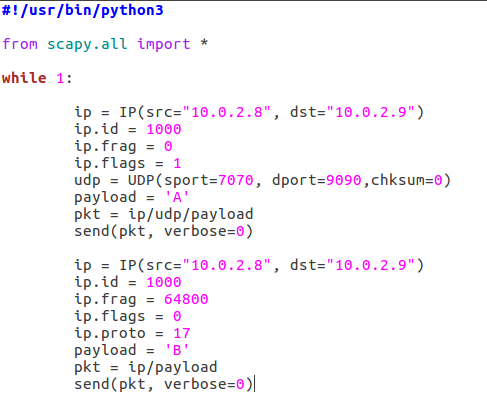


我们成功发送了65527字节的数据



## Task 1.d：Sending Incomplete IP Packet

攻击代码如下



构造两个分片，第一个分片的偏移为0；第二个分片的偏移是64800。为了重组，目标主机会分配64k的缓存，可通过不断发包消耗目标内存。可以明显感觉到被攻击的主机变卡了。