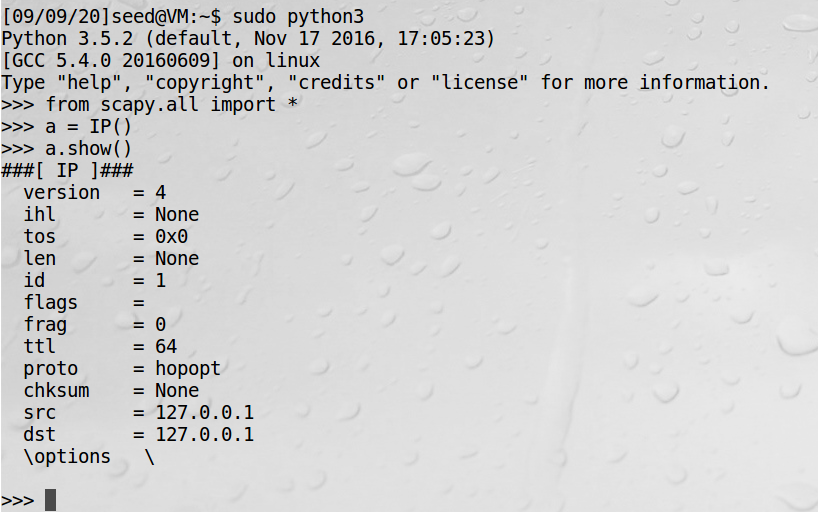
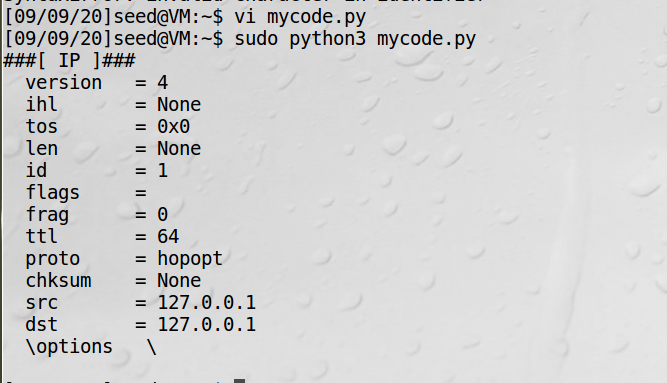
**Packet Sniffifing and Spoofifing Lab实验报告**

**57117207 高晓悦**

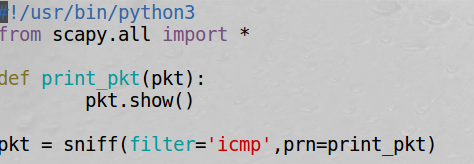
**Lab Task Set 1: Using Tools to Sniff and Spoof Packets**

1. 安装scapy
2. 测试：



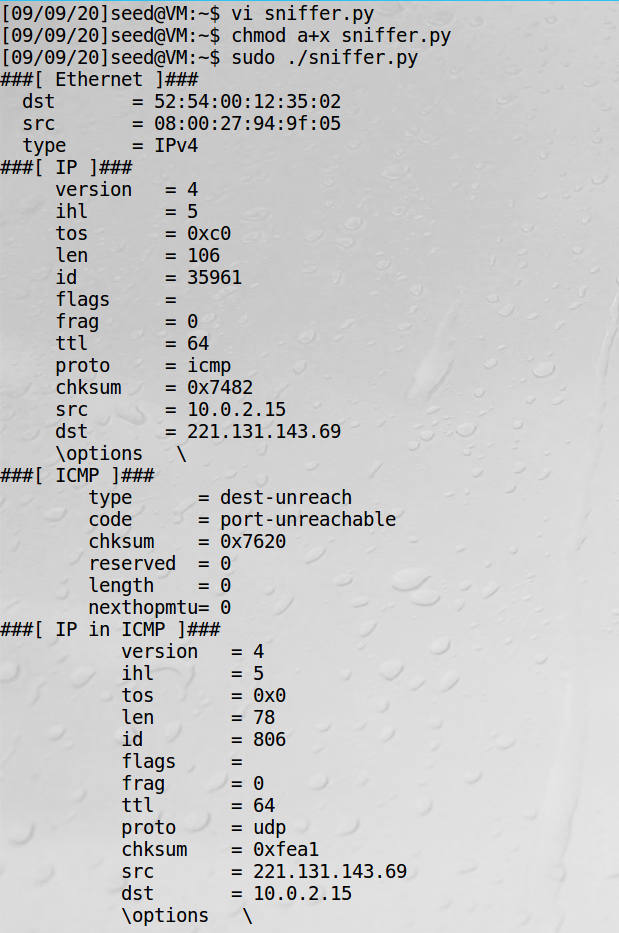
**Task 1.1: Sniffifing Packets**

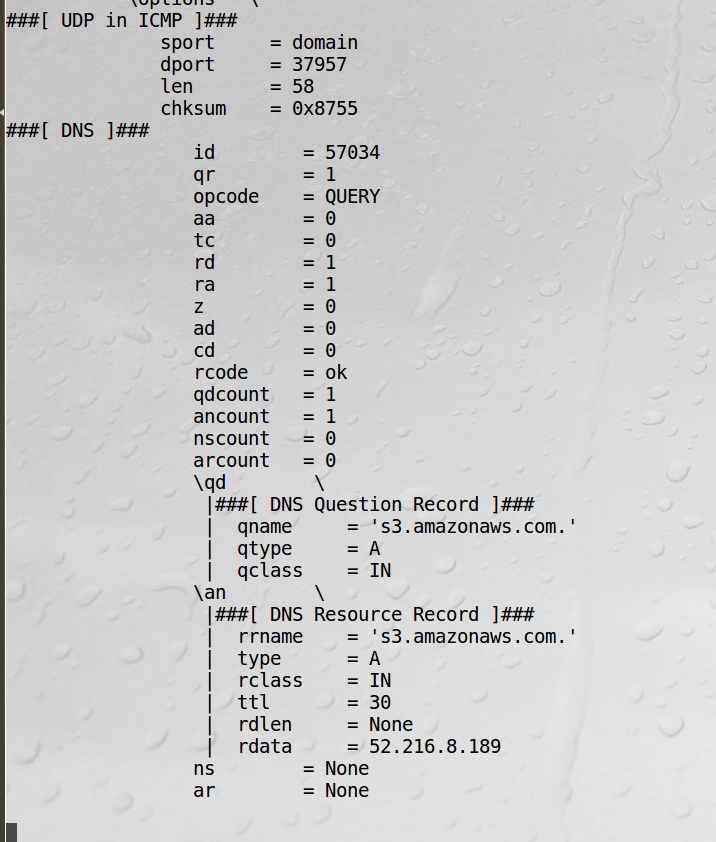
1. 编写sniffer.py程序来监听ICMP包：

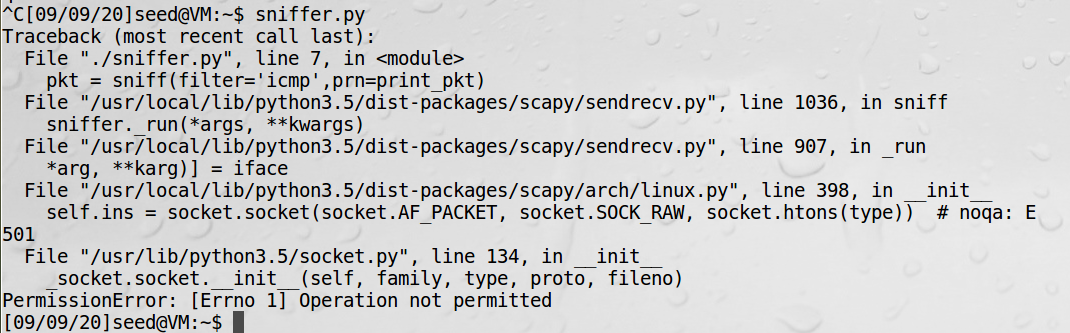


**Task 1.1A：**

1、root权限下运行结果：

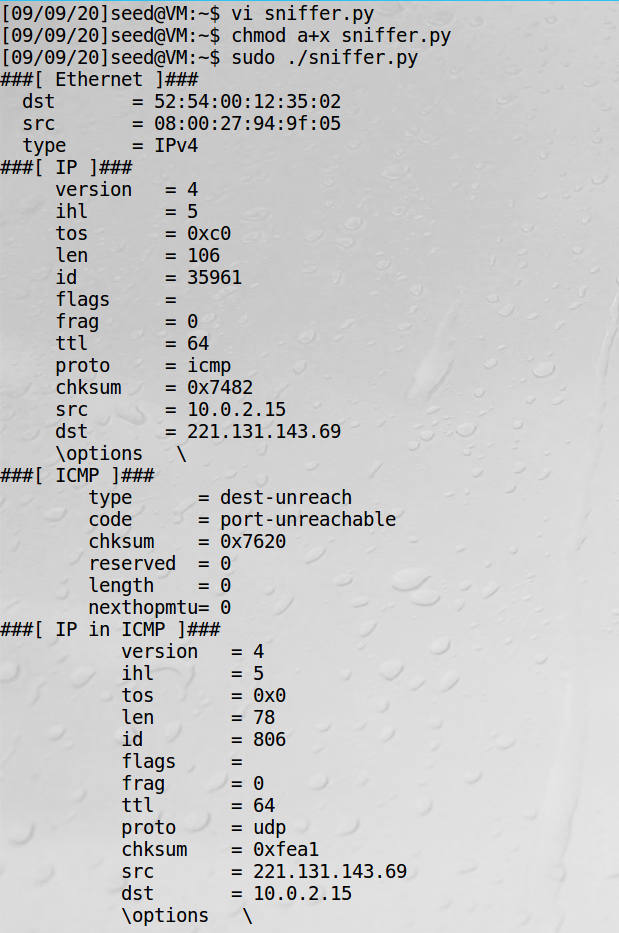


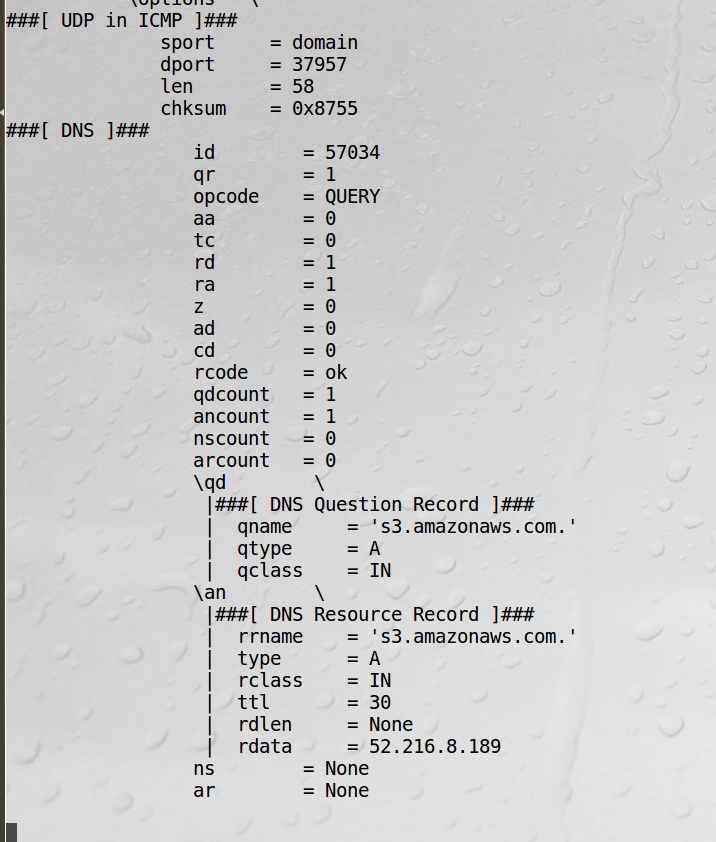


2、普通权限下运行，报错，错误为Operation not permitted，表示普通用户的权限不足以进行监听：

**Task 1.1B：**

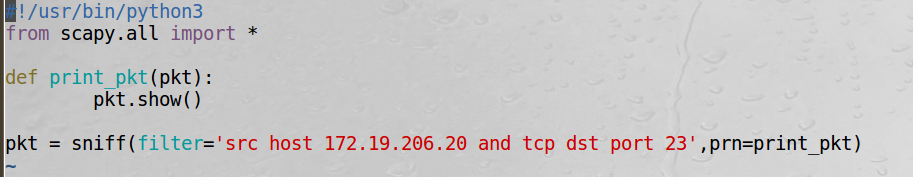
1、监听ICMP报文，结果同上：



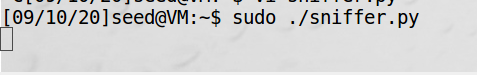
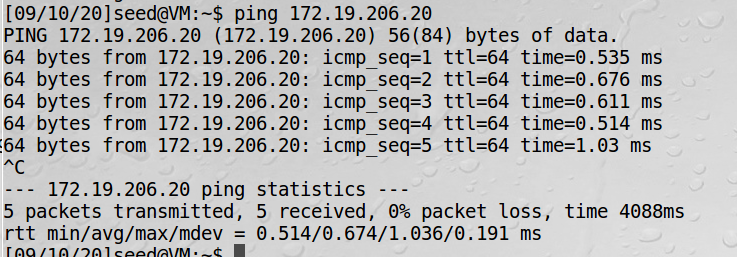


2、监听来自指定IP地址172.19.206.20、目的端口为23的TCP连接：

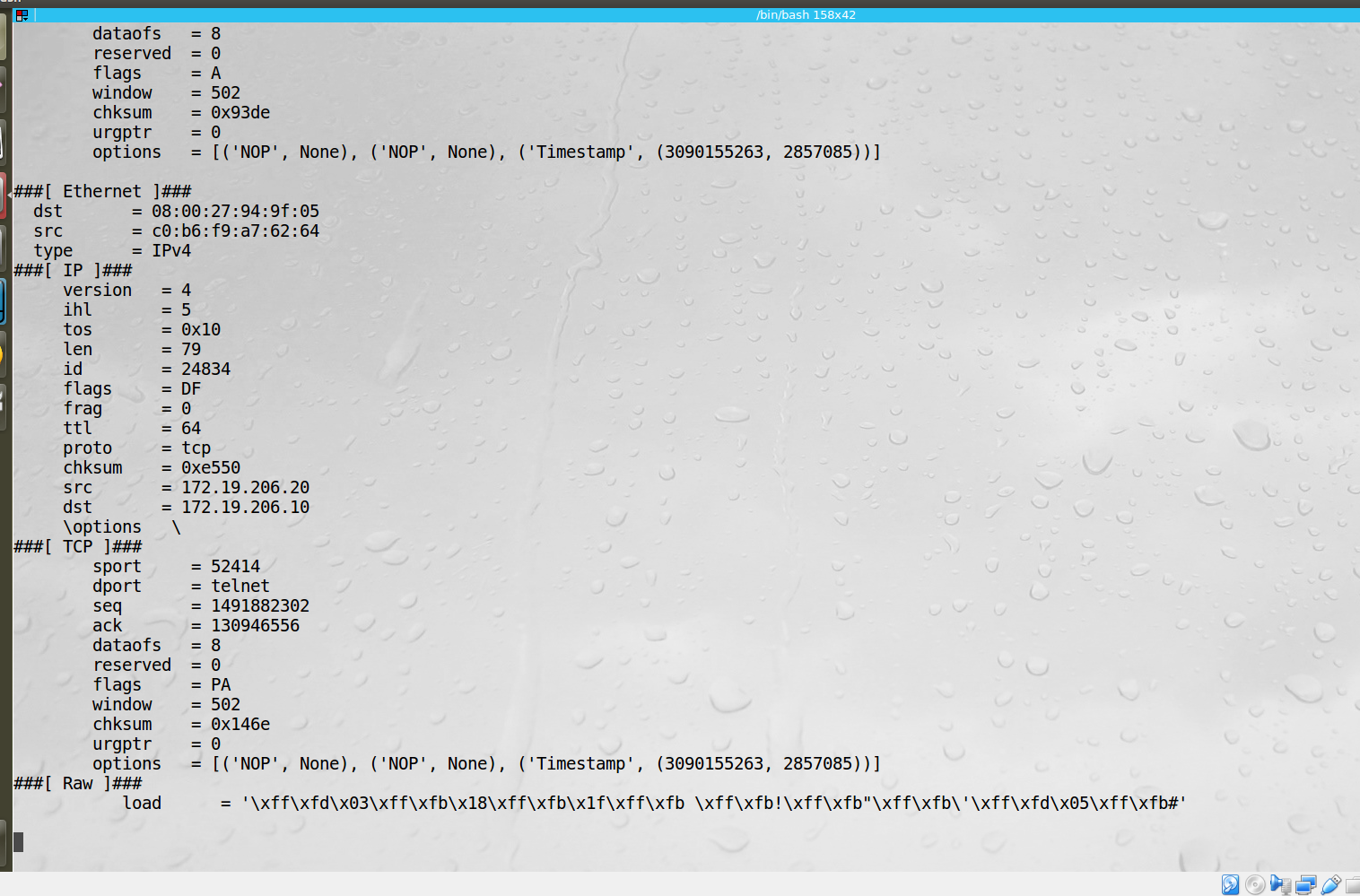
先更改sniffer.py代码：



执行sniffer.py，另一个终端中ping，发现未能监听到数据包：

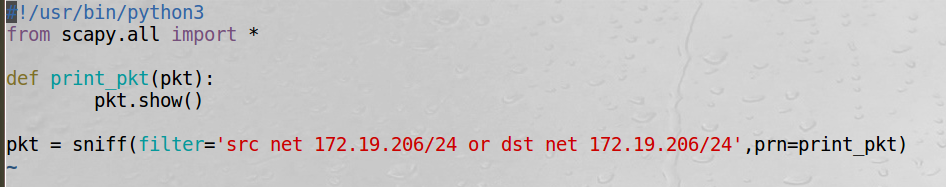


在172.19.206.20上打开telnet，请求连接seed虚拟机，sniffer.py监听到多个数据包：



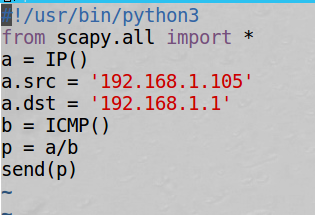
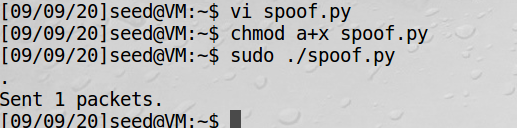
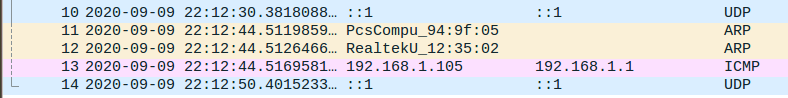
3、监听源IP或目的IP为特定子网的数据包：

更改sniffer.py程序：



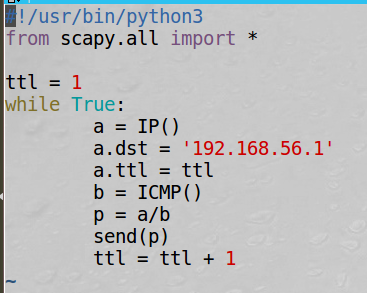
ping任意IP，sniffer.py没有监听到数据包，ping 172.19.206.30，成功监听到数据包。

**Task 1.2: Spoofifing ICMP Packets**

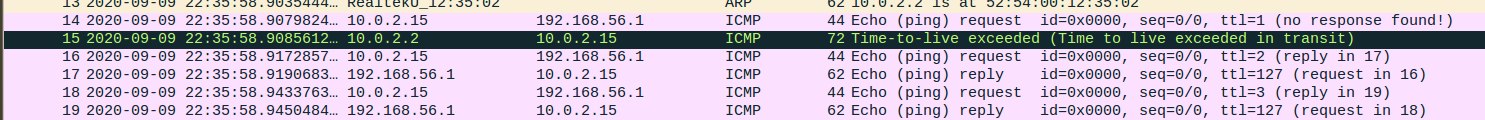
1. 编写spoof.py并执行：
2. 用wireshark查看：

**Task 1.3: Traceroute**

1. 编写traceroute.py并执行：



2、用wireshark查看，第一个reply在ttl=2时出现，经过了两个路由器：



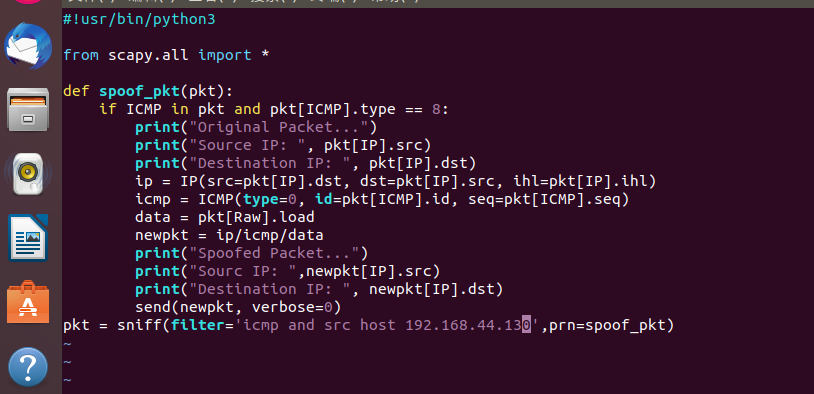
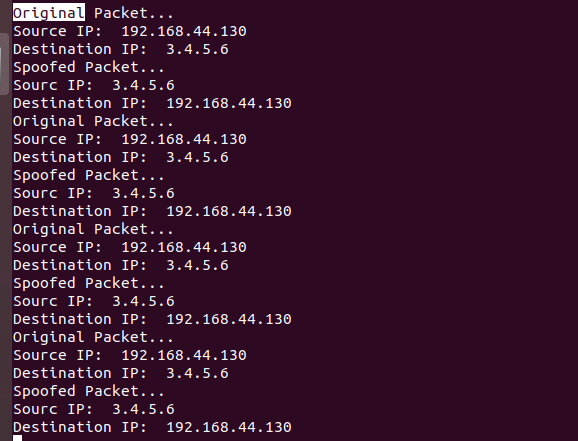
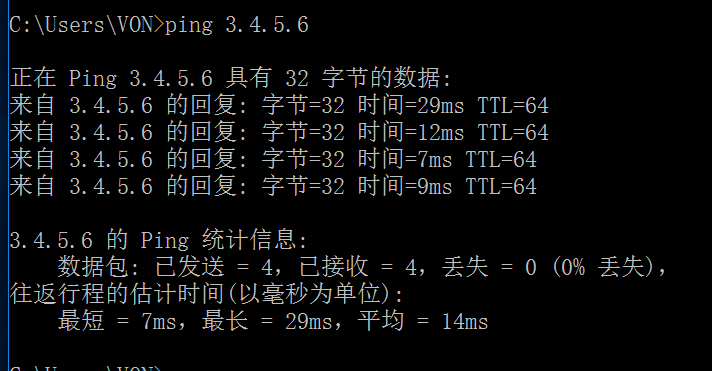
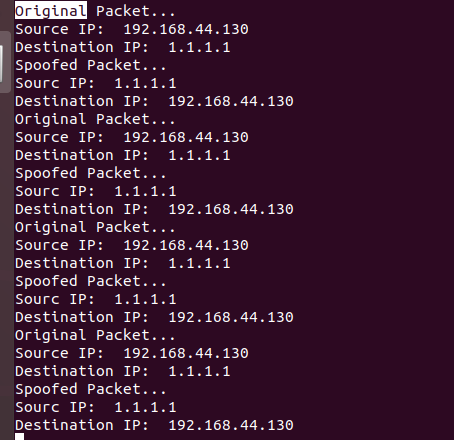
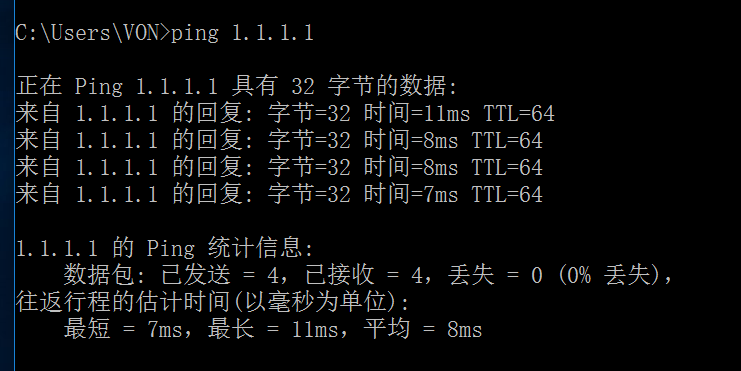
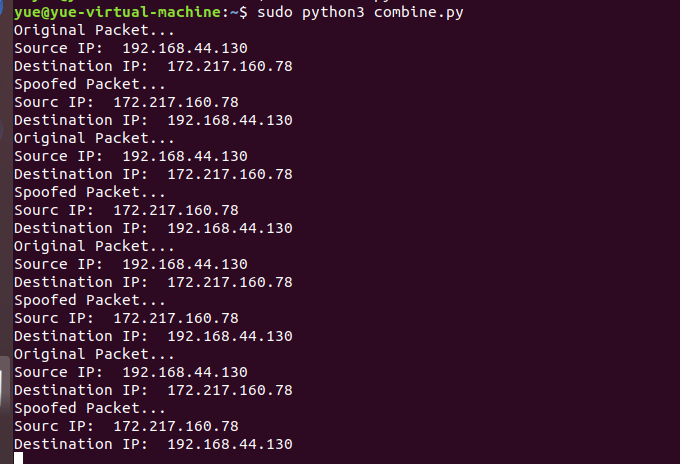
**Task 1.4: Sniffifing and-then Spoofifing**

1、指定发起攻击的主机，其IP为192.168.44.129：



1. 指定被攻击的主机，其IP为192.168.44.130，发起攻击的主机和被攻击主机位于同一局域网中：



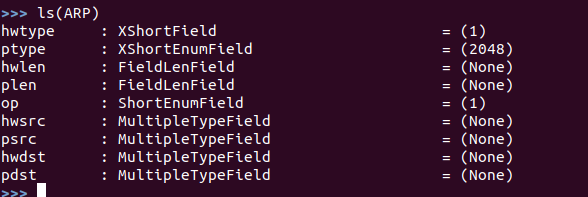
1. 编写combine.py并执行，进行监听：
2. 在被攻击主机中ping任意IP，都能ping通，且都在发起攻击主机中监听到请求，说明攻击成功：

**ARP Cache Poisoning Attack Lab实验报告**

**57117207 高晓悦**

**Task 1: ARP Cache Poisoning**

先查看scapy的ARP对象方法：

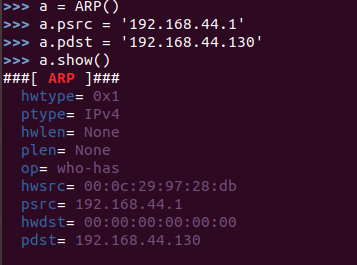


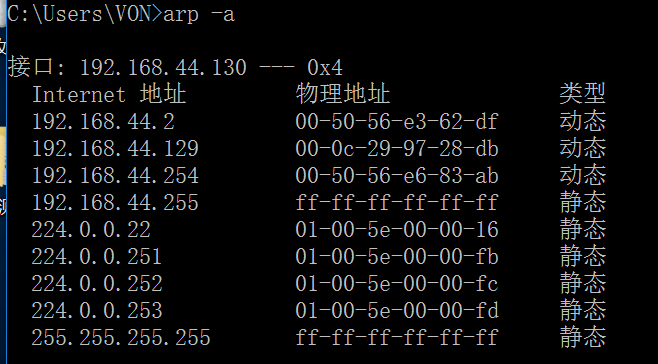
**Task 1A (using ARP request)：**

1、需要准备三台主机——A（192.168.44.130）、B（192.168.44.1）、M（192.168.44.129）：

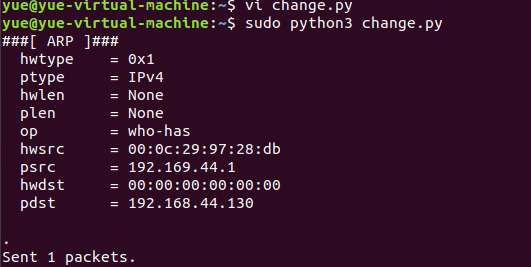
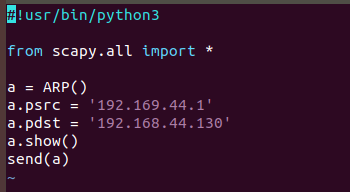


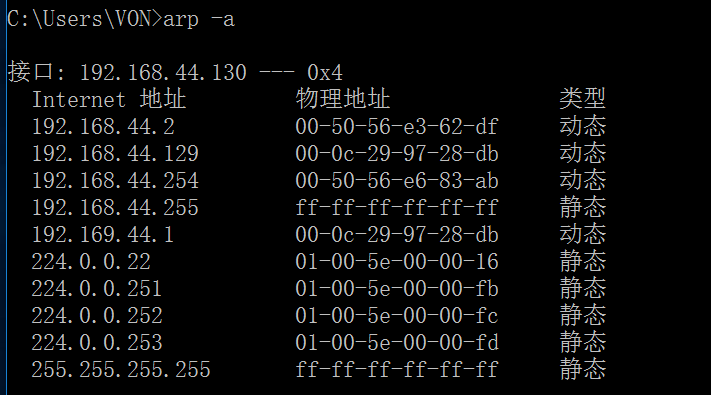


2、创建ARP对象a，a的psrc设为主机B的IP，pdst设为主机A的IP，让主机A认为请求来自主机B，继而将主机B的MAC地址变为主机M的MAC地址：

3、查看主机A的ARP表：

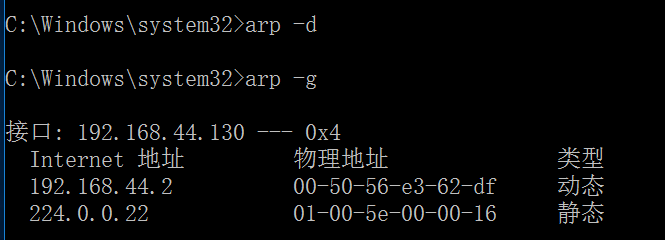
4、执行脚本，发送数据包：

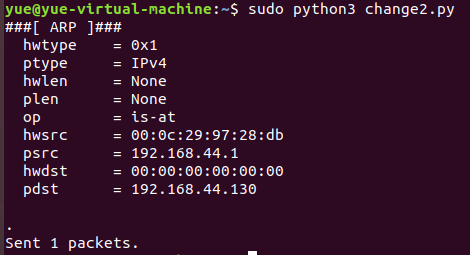
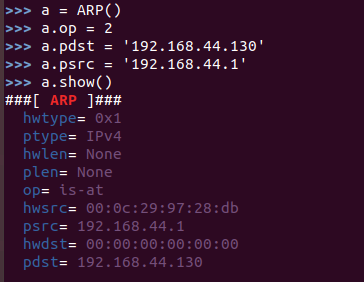


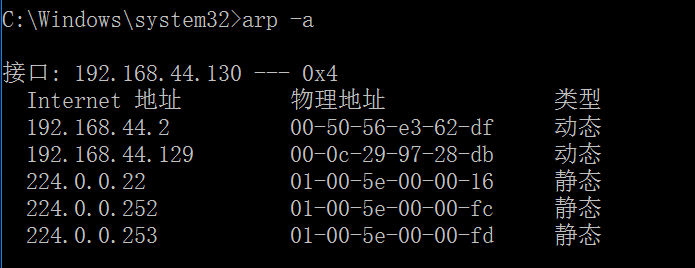
5、再次查看主机A的ARP表，发现主机B（192.168.44.1）的MAC地址变为主机M的MAC地址：

**Task 1B (using ARP reply)：**

1、清除主机A的ARP表：

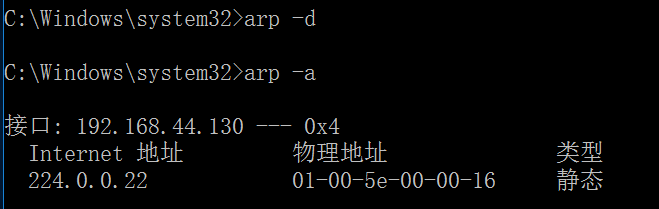


2、构建ARP包并发送：

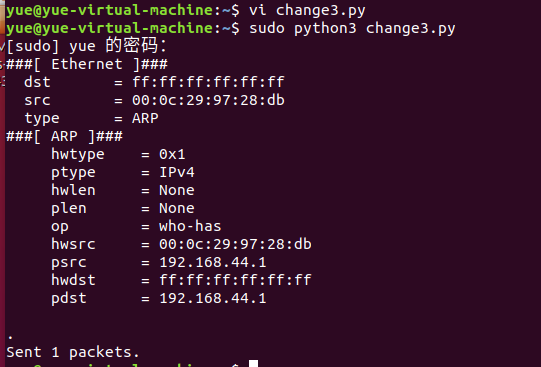
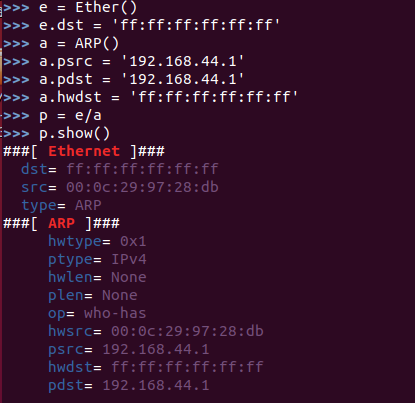
3、发送后再次查看主机A的ARP表，发生改变：

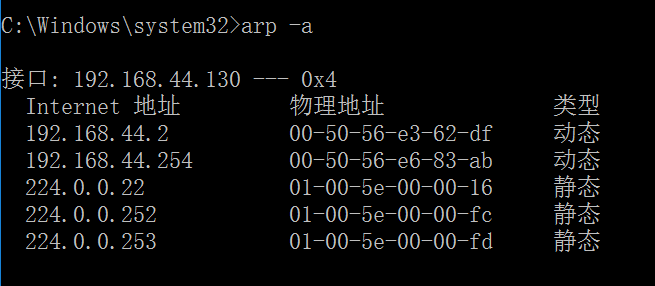
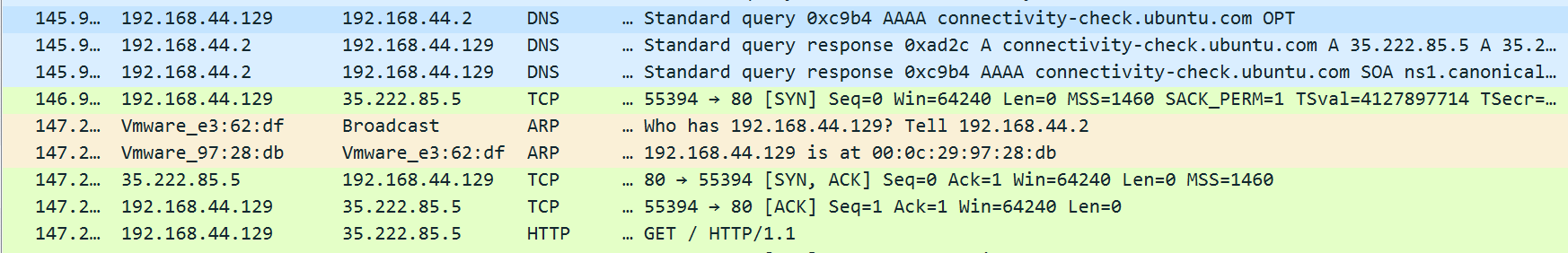
**Task 1C (using ARP gratuitous message)：**

1、清除主机A的ARP表：



2、构造ARP包并发送：



1. 发送后再次查看主机A的ARP表，发现没有改变：
2. 查看wireshark抓包情况，发现原因在于主机广播了自己的MAC地址：

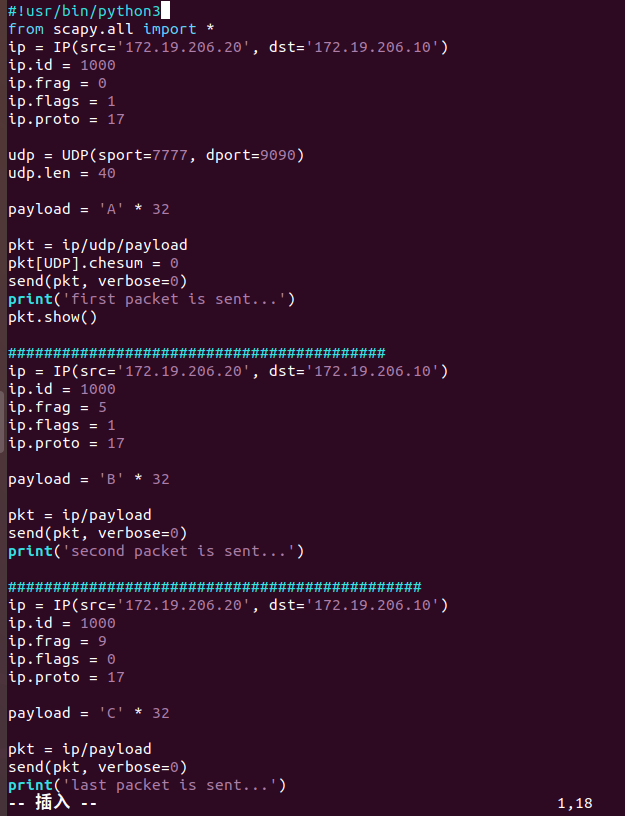
**IP/ICMP Attacks Lab实验报告**

**57117207 高晓悦**

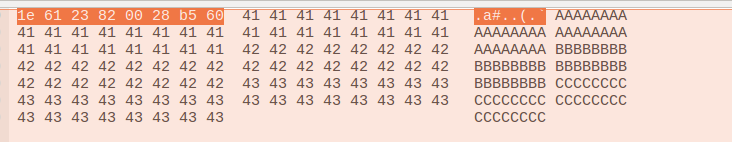
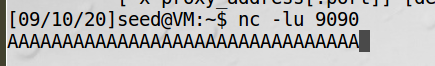
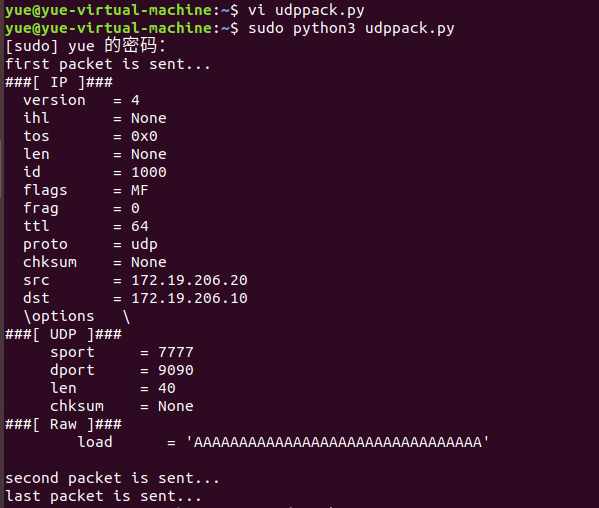
**Tasks 1: IP Fragmentation**

**Task 1.a: Conducting IP Fragmentation**

1、编写udppack.py，构造三个数据包：

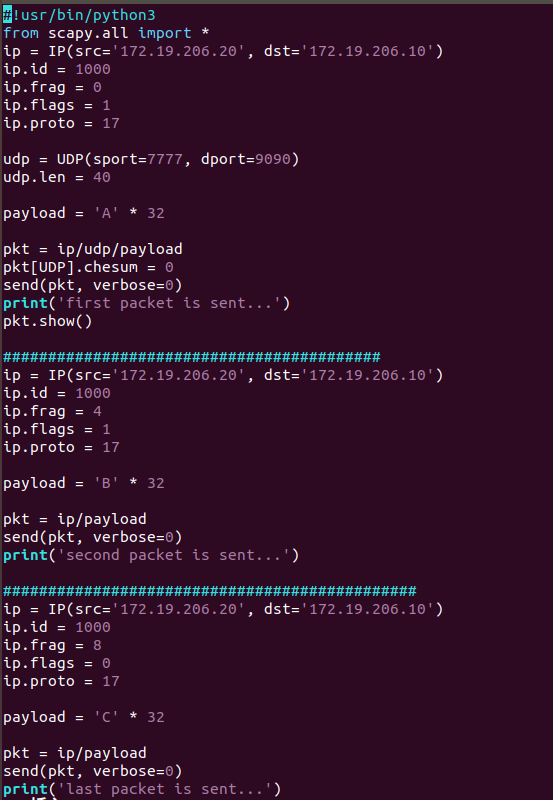


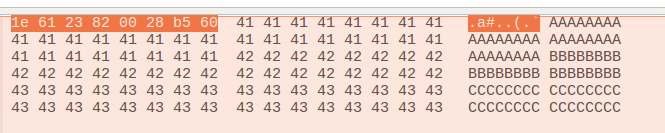
1. 开启udp监听和wireshark抓包：

3、执行脚本发送包，172.19.206.10收到172.19.206.20发过去的数据，wireshark中也捕获到了响应数据包，三个数据包拼接在一起：

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

1、编写udppack2.py，把第二个数据包的片偏移量frag设为4，第三个数据包的frag设为8：



2、观察wireshark抓包结果，第二个数据包的前8个字节与第一个数据包的后8个字节重合：

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

1. 编写udppack3.py，不停发送长为210的分片：

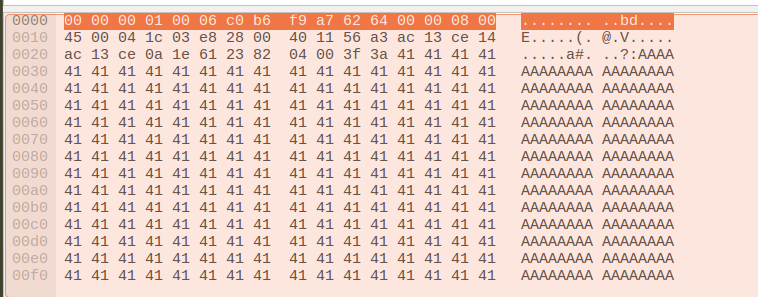
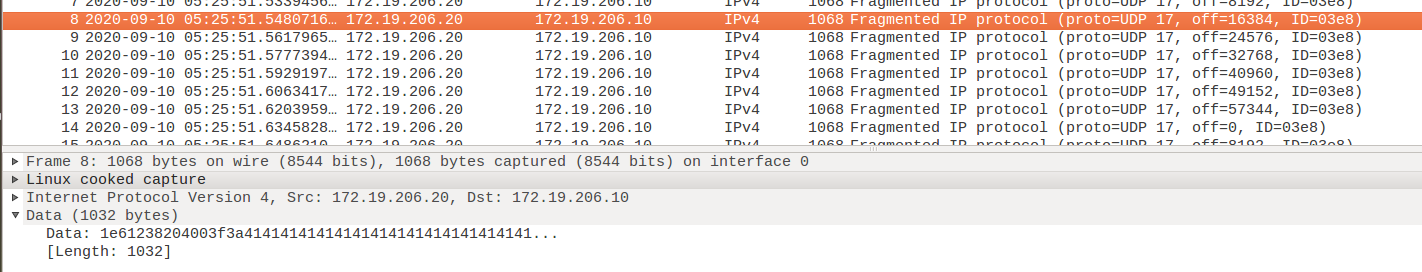


1. 执行脚本，发送数据包，服务端没有反应：



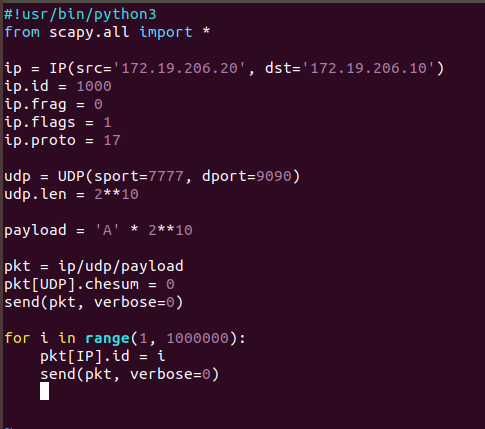


3、wireshark抓包结果为：



**Task 1.d: Sending Incomplete IP Packet**

1、编写udppack4.py，发送大量id不同的片段消耗资源：



2、观察服务端与wireshark抓包结果，数据包中出现大量id不同的片段：