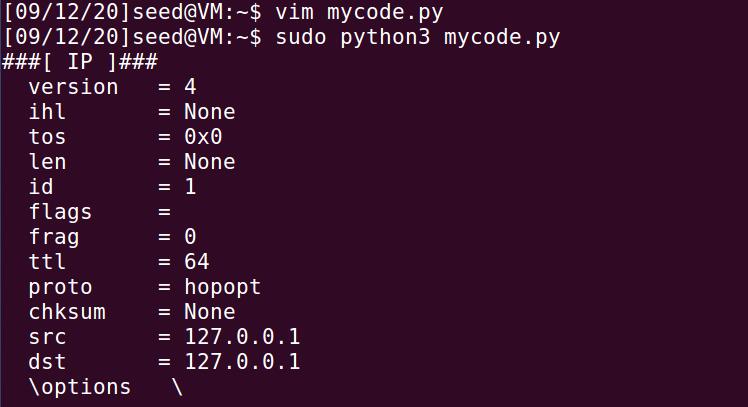
Lab3实验报告

57117124 陈浩

1. Packet Sniffing and Spoofing Lab

用root权限运行程序，捕获到一个ICMP报文，并打印出了它的相关信息：

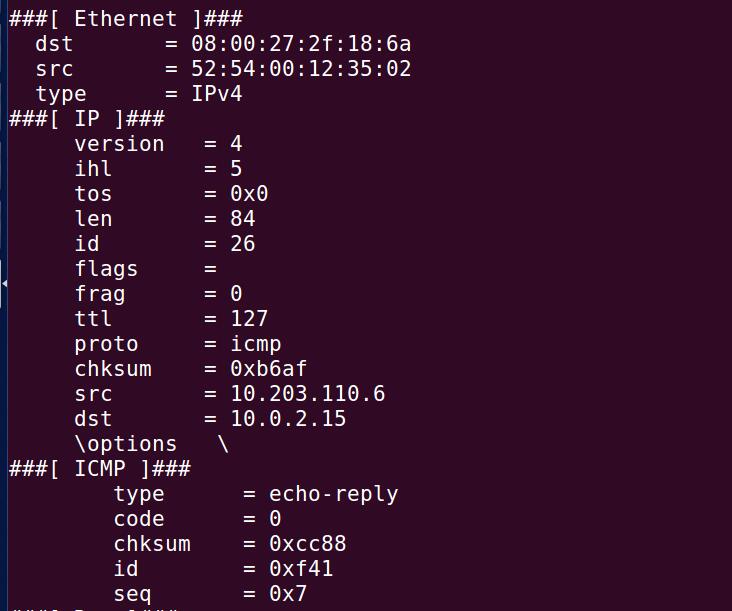


Task1.1a Sniffing Packets

Physical machine: 10.203.110.6

Virtual machine(Ubuntu): 10.0.2.15

虚拟机ping物理机：



Task1.1b

1.只抓ICMP(见上)

filter=’icmp’

2. 捕获来自特定IP且目标端口号为23的任何TCP数据包:

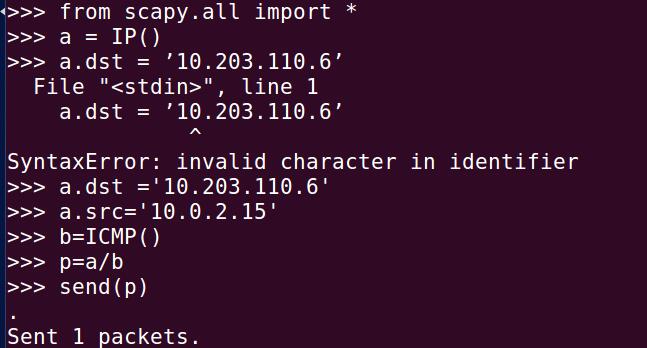


3. 捕获来自或前往特定子网的数据包。

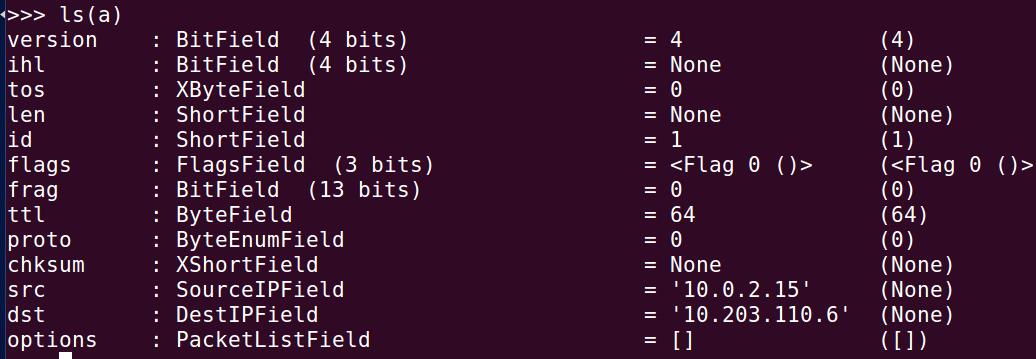
filter = ‘net 128.230.0.0/16’

Task1.2 Spoofing ICMP Packets

修改a.src，修改后的程序如下:

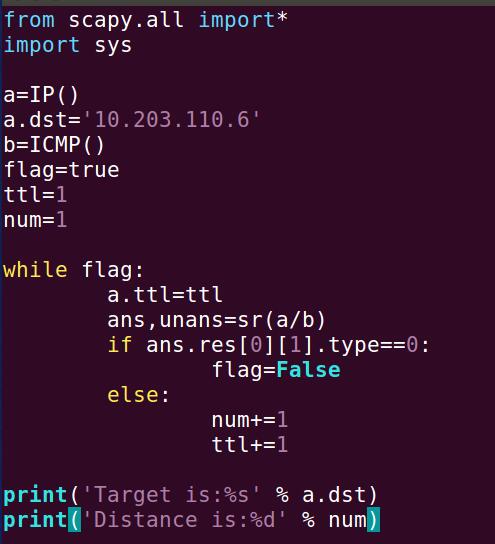


查看属性：

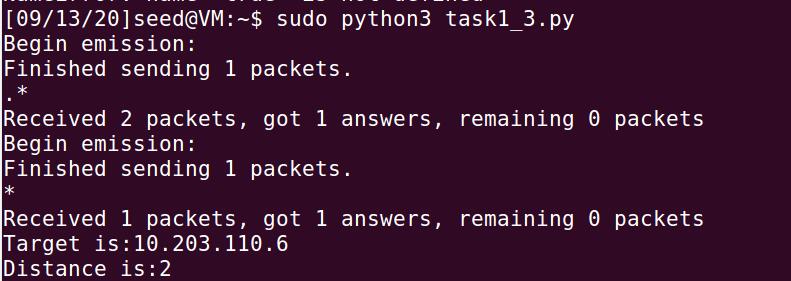


Task 1.3 Traceroute

根据所给代码提示，写Python代码以计算虚拟机到宿主机的距离：

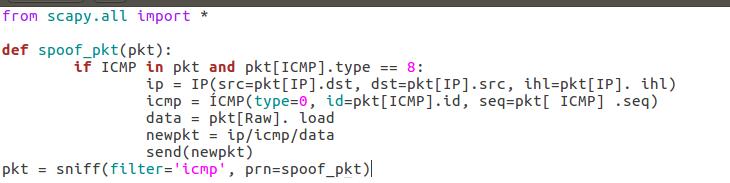


运行结果：

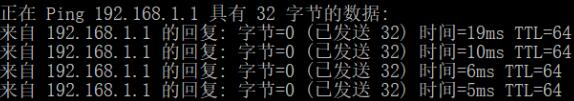


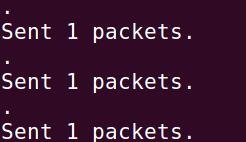
Task1.4: Snifﬁngand-thenSpooﬁng

任务的目的是将ICMP报文的源地址和宿地址互换，然后发送形成伪造攻击。代码如下：



运行此代码，然后在物理机的命令行内ping一个伪造的地址如192.168.1.1，本应无法ping通，但此时却ping通：

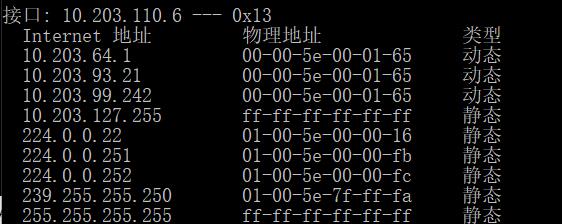




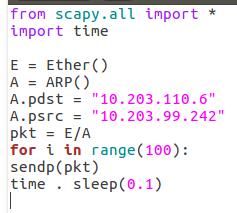
2. ARP Cache Poisoning Attack Lab

Task1 ARP Cache Poisoning

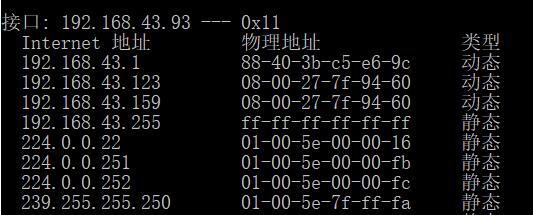
使用ARP请求污染。首先在宿主机（下面称为A）中使用arp -a命令查看ARP缓存表：



这里我们攻击的目标IP地址是10.203.99.242（下面称为B），其MAC地址为，写发送ARP请求报文的代码，pdst和psrc的地址分别为A和B的IP地址，然后执行，就能让A不断向B发送ARP请求报文，这样就能将其污染。



运行程序，发送了大量包后，用arp -a命令查看：



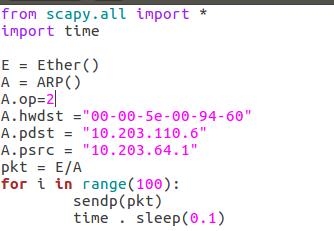
此时发现主机B的MAC地址变成了00-00-5e-00-94-60，污染成功

此时再让B向A发起几次ping通信，可以把ARP缓存表刷新。

Task1B

使用ARP应答污染。

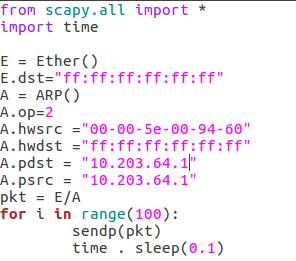
代码根据Task 1A的代码修改，如下：



执行程序后同样会看到B的MAC地址被修改，表明污染成功

Task1C

通过不断地广播ARP报文达到污染的效果。代码在原有基础上进行修改:



将源、宿IP地址均设置为污染目标IP地址，宿MAC地址为全f。同样执行上述程序后，用arp -a命令查看，发现污染成功。

3. IP/ICMP Attacks Lab

Task1: IP Fragment

1.a参照手册，将代码补全，保存为frag.py，然后打开Wireshark进行监测：



在攻击端运行frag.py，通过Wireshark监测发包结果：



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

修改frag.py文件，将第二片报文的前8个字节与第一片报文的后8个字节重合，然后将第二片报文的载荷中的A全部改为B：



Wireshark结果：



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

将IP头中的字段设置为，然后不断发送为1的报文，也就是一直继续分片。当分片总长超过后，设置其为0。此时UDP服务器崩溃了

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

依旧改写frag.py，改成只发送部分分片，并且不断改变id的程序：



结果导致作为UDP服务器的虚拟机（ip为192.168.43.159）内存占用不断升高