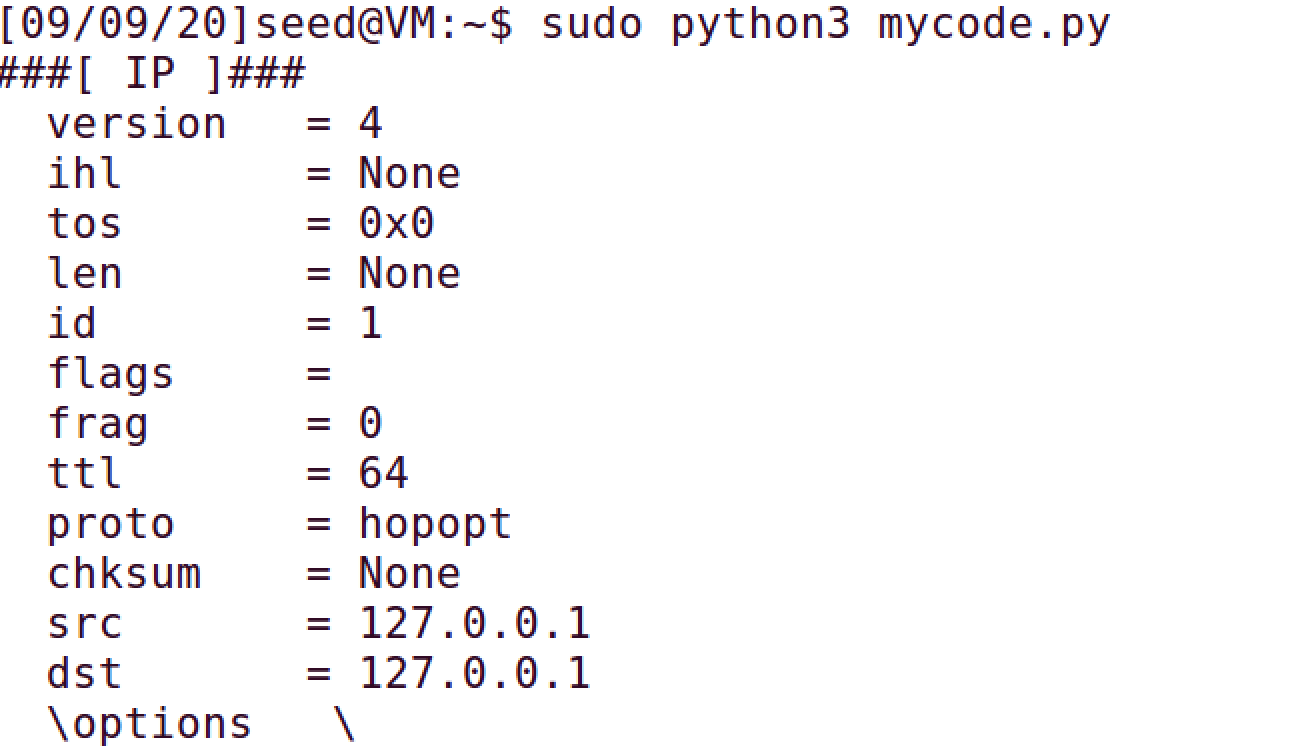
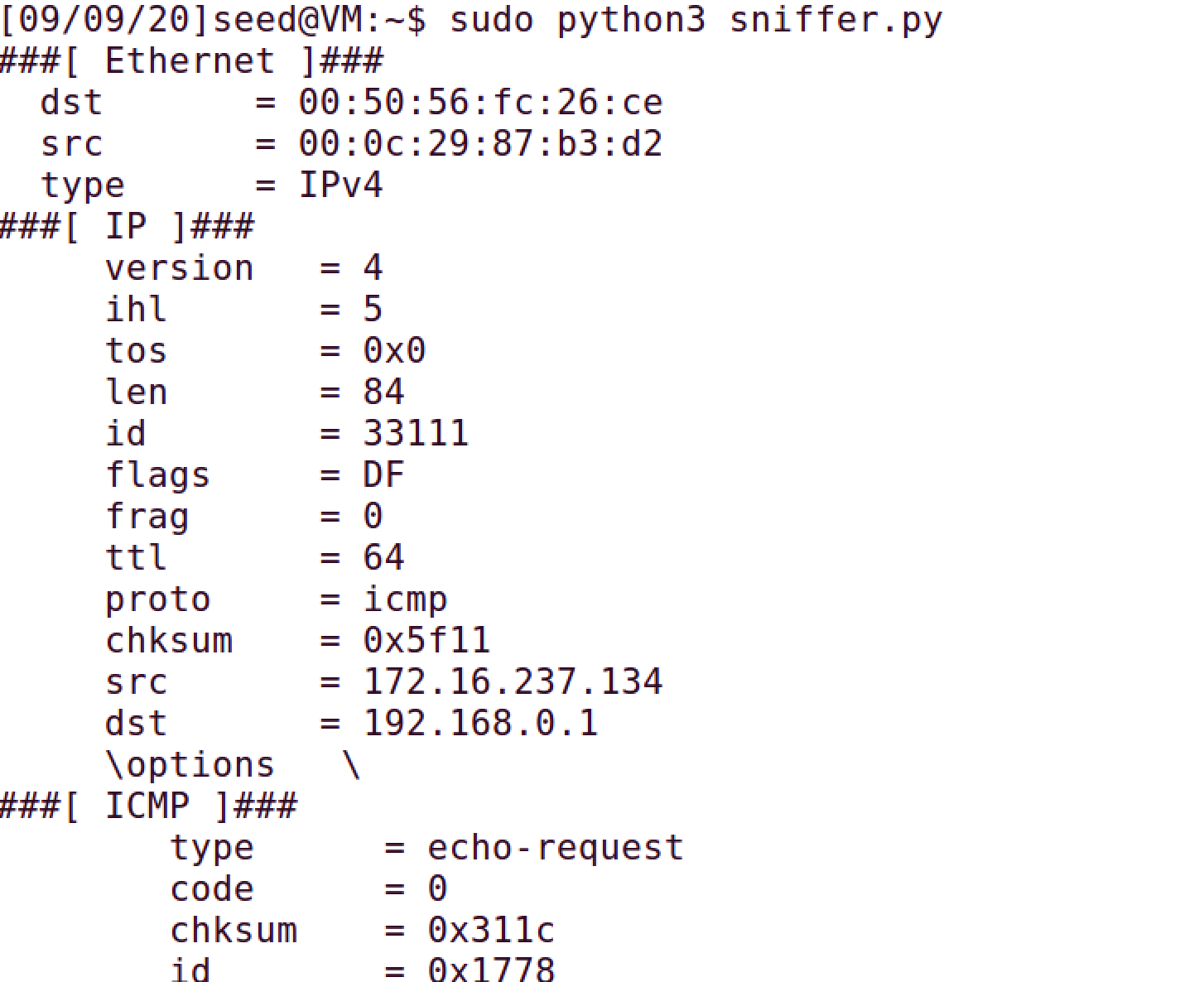
Lab Task Set 1: Using Tools to Sniff and Spoof Packets

57117132费越

用管理员权限运行测试代码：



Task 1.1: Sniffing Packets



在使用root权限时可以成功执行

使用普通用户权限运行程序报错：



Task 1.1B

仅捕获ICMP报文：

from scapy.all import\*

def print\_pkt(pkt):

pkt.show()

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

捕获特定IP发出的，目的端口为23的TCP包：

from scapy.all import\*

def print\_pkt(pkt):

pkt.show()

pkt = sniff(filter='src host 192.168.1.1 and tcp dst port 23',prn=print\_pkt)

捕获从特定子网中发起或前往特定子网的报文：

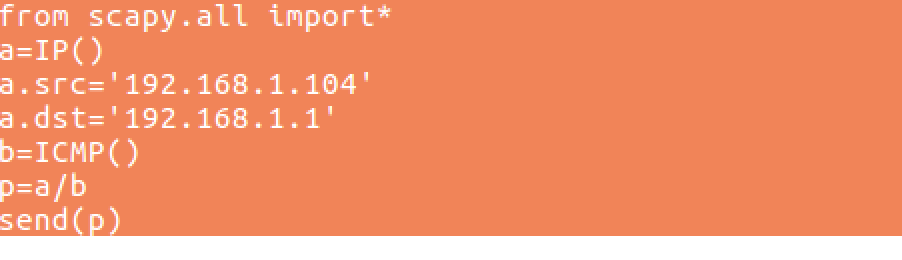
from scapy.all import\*

def print\_pkt(pkt):

pkt.show()

pkt = sniff(filter='net 128.230.0.0/16',prn=print\_pkt)

Task 1.2: Spoofing ICMP Packets



将a的src设置为伪装的源地址，dst设置为目标IP后，使用Wireshark查看：



成功伪装

**Task1.3 Traceroute**

编辑代码

from scapy.all import\*

ttl=1

while True:

a=IP()

a.dst='180.101.49.12'

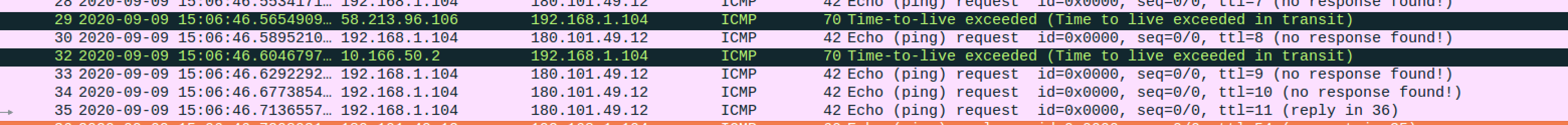
a.ttl=ttl

b=ICMP()

p=a/b

send(p)

ttl=ttl+1



第一个回应在ttl=11时出现，所以经过11个路由器

Task 1.4: Sniffing and-then Spoofing

编写代码：

from scapy.all import\*

def spoof\_pkt(pkt):

if ICMP in pkt and pkt[ICMP].type ==8:

ip=IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)

icmp=ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)

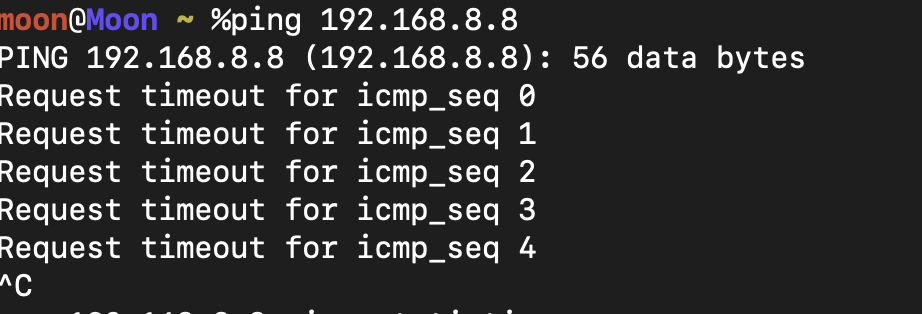
data=pkt[Raw].load

newpkt=ip/icmp/data

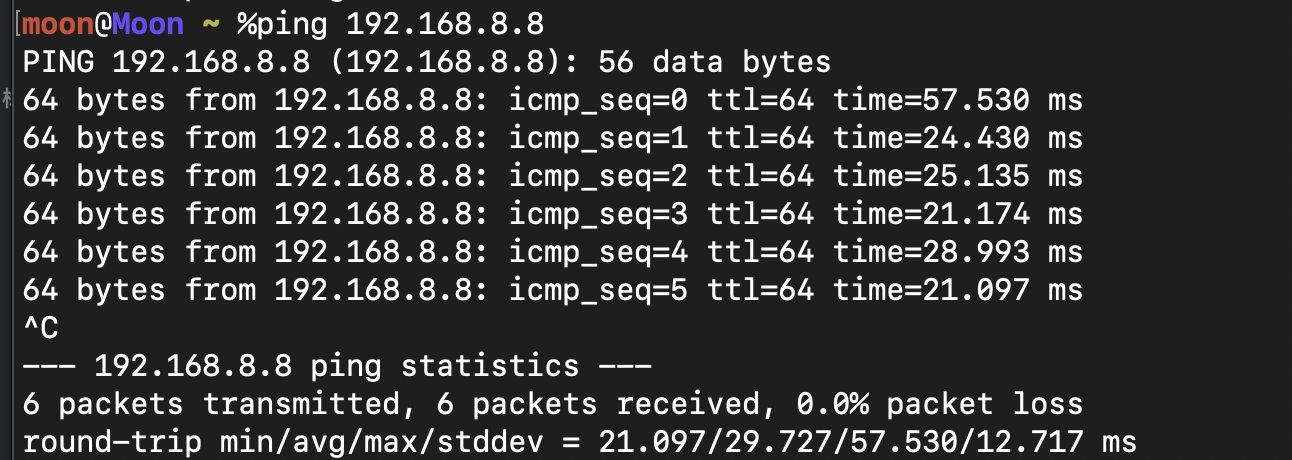
send(newpkt)

pkt=sniff(filter=’icmp’,prn=spoof\_pkt)

在宿主机上ping一个不存在的地址

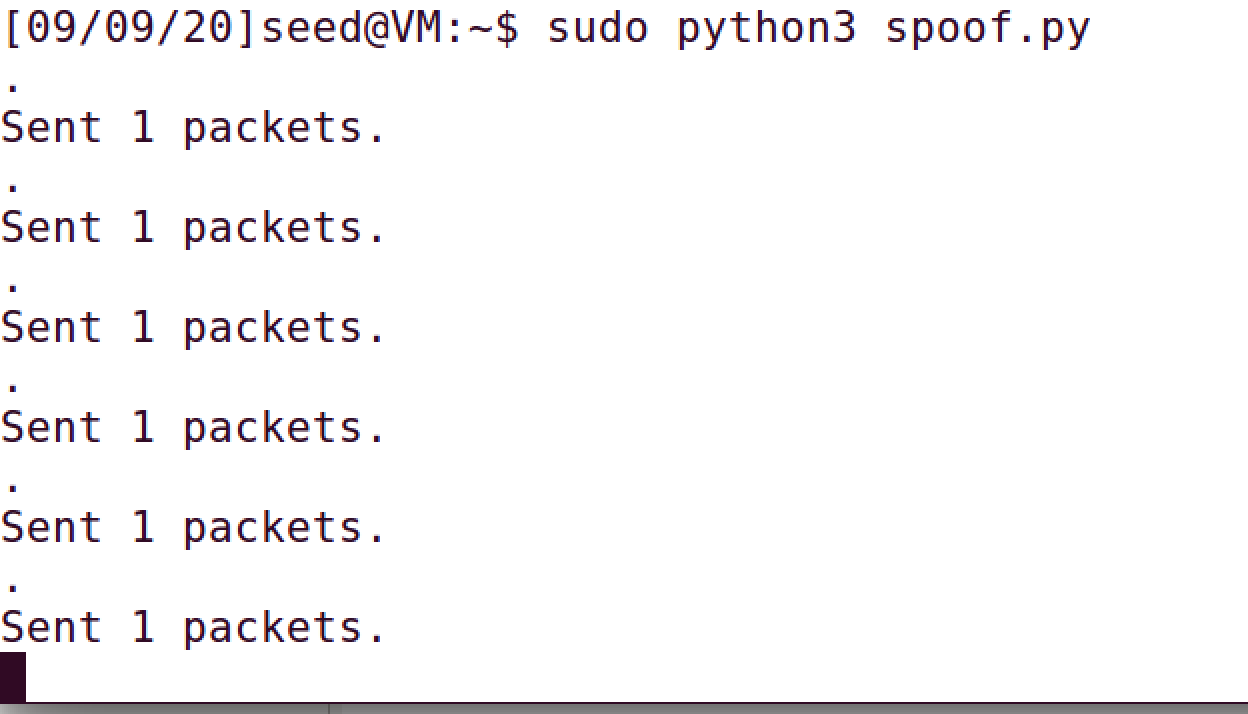


运行脚本，再次尝试



成功ping通

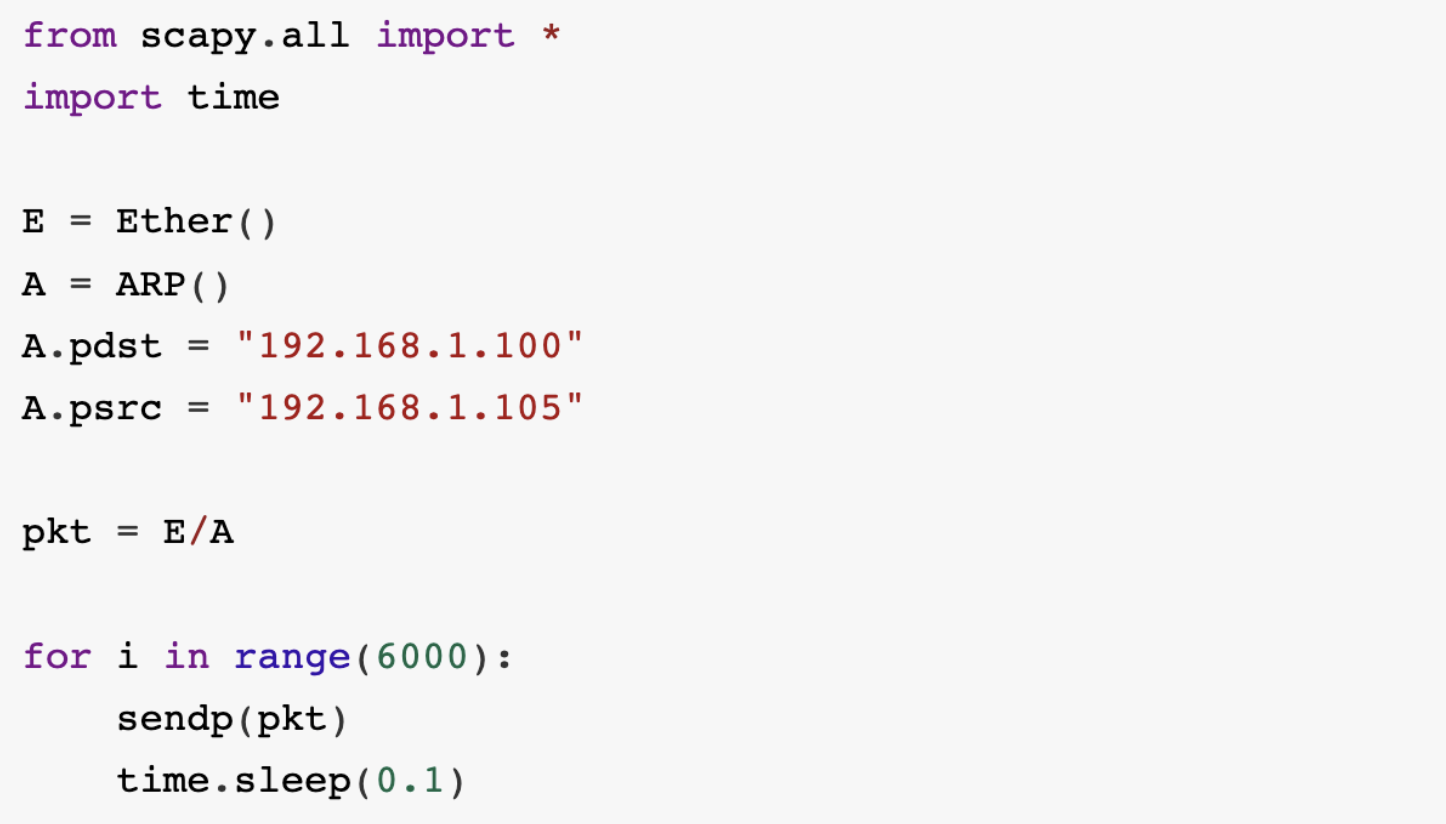
此时脚本显示：



报文伪造成功。

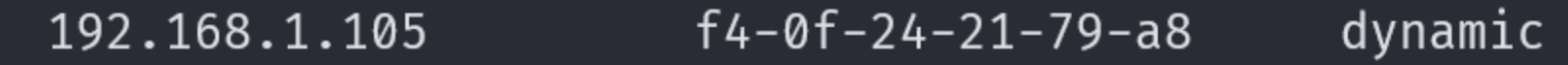
ARP Cache Poisoning Attack Lab

Task 1A (using ARP request).

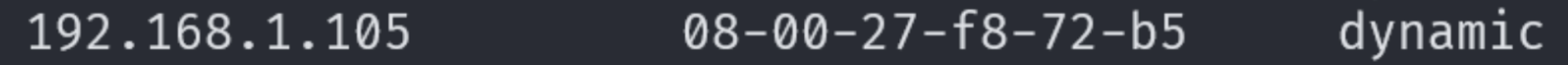


不停地发送ARP请求报文

初始的ARP表

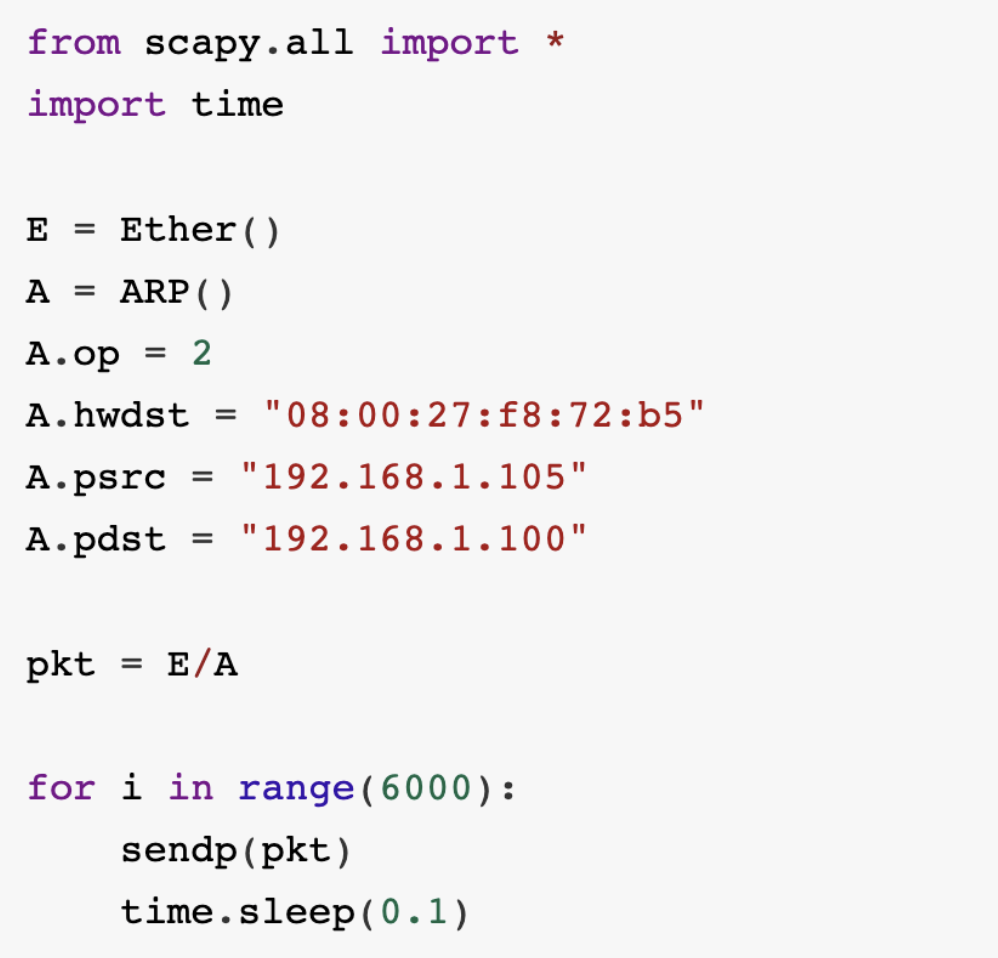


运行脚本后查看



已经成功污染ARP缓存表

Task 1B (using ARP reply)

  
不停地发送ARP响应报文

运行结果与task1A相同

Task 1C (using ARP gratuitous message)



不停地广播ARP gratuitous报文，将源宿地址都设为要污染的IP地址，宿MAC地址设置为ff-ff-ff-ff-ff-ff，源MAC地址设为攻击者的MAC地址。

运行结果与task1A相同

Tasks 1: IP Fragmentation

Task 1.a: Conducting IP Fragmentation



将UDP报文分片：

总长度：头部8字节+载荷96字节，共104字节；

第一片报文片偏移为0，flags为1，表示接下来还有分片；

第一片IP报文包含UDP首部和前32字节载荷；

第二片IP报文片偏移量为32/8+1=5，不再包含UDP首部；

第三片IP报文片偏移量为5+32/8=9，flags设为0，表明后面没有分片

在另一个虚拟机中使用命令sudo nc -lu 9090

在虚拟机1中运行脚本，在虚拟机2中可以准确接收到96个A。

Task 1.b: IP Fragments with Overlapping Contents

将第二片报文的片偏移量frag设置为4，第三片设置为8，UDP报文长度设置为96，

使得第二片报文的前8字节与第一片报文的后8字节重合。

把第二片报文的载荷中A全改为B



运行脚本，虚拟机2中收到的前24字符是A，中间有32个B，然后有32个A，报文发生重叠时，后面的内容会覆盖住前面的内容。

交换第二片IP报文与第一片IP报文发出顺序，结果相同，因为IP报文的重组是在获得IP报文的所有分片之后才开始进行的。

Task 1.c: Sending a Super-Large Packet

将IP头中len字段设为2的16次方，然后不停的发送分片，使得分片长度超过2的16次方，最后设置flags=0结束报文，最终会使得接收方虚拟机的接收程序无法正常工作。

Task 1.d: Sending Incomplete IP Packet

修改发送程序，使发送程序用不同的ID只发送第1和第3片报文，即接收方永远缺少中间的第二片报文，所以接收方缓存会把接收到的报文一直存储，知道内存被完全占满。形成拒绝服务攻击（DOS）。