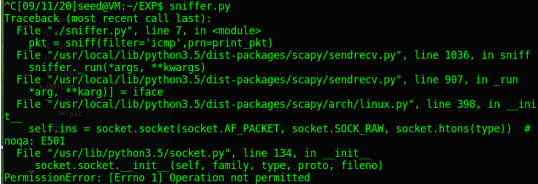
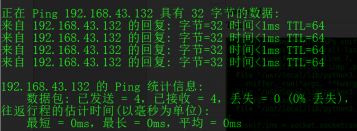
LAB1

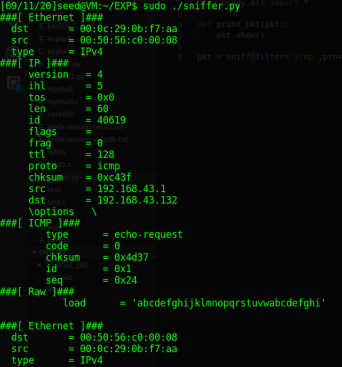
TASK1.1a



用 sudo 运行后，在主机上 ping 虚拟机：

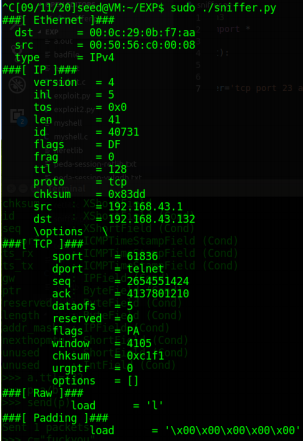


虚拟机收到如下四个报文



TASK1.1b

只抓 tcp，且源地址为 192.168.43.132，目的端口 23(telnet 端口)：



TASK1.2

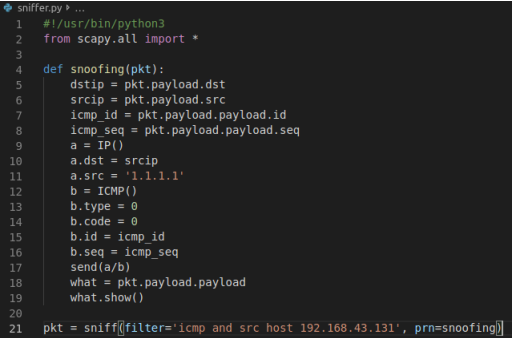
在虚拟机上监听到了这个 dst 为 10.0.2.3 的报文。



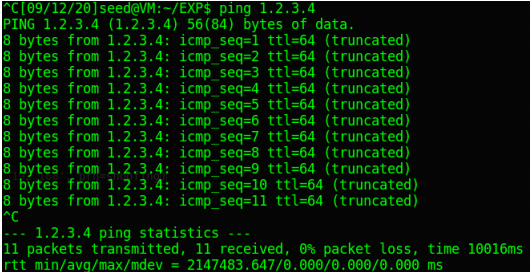
TASK1.3



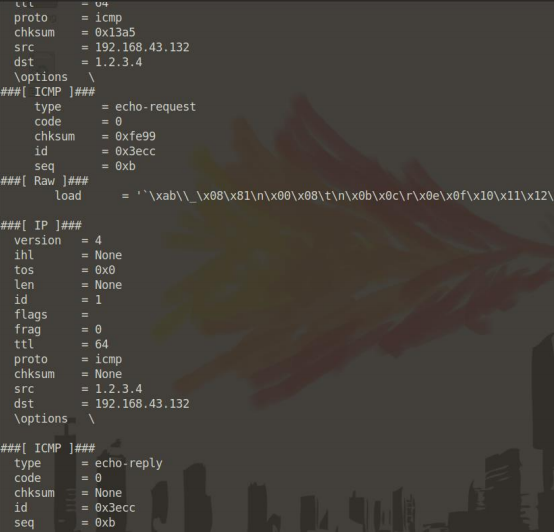
TASK1.4



运行。。。监听源地址为 192.168.43.132 的另一台虚拟机 2； 虚拟机 2 向一个 ip 发送 icmp 报文



Sniffer 进行拦截并回复：



如图分别是 request 报文的一部分和伪造的 reply 的报文，可以看到对应的 icmp 报文中 id 和 seq 是对应的，而 ip 源宿地址是相反的。 对一个无效 ip 发起 ping，结果收到 reply 说明伪造成功。

LAB2

TASK1A



初始的ARP表

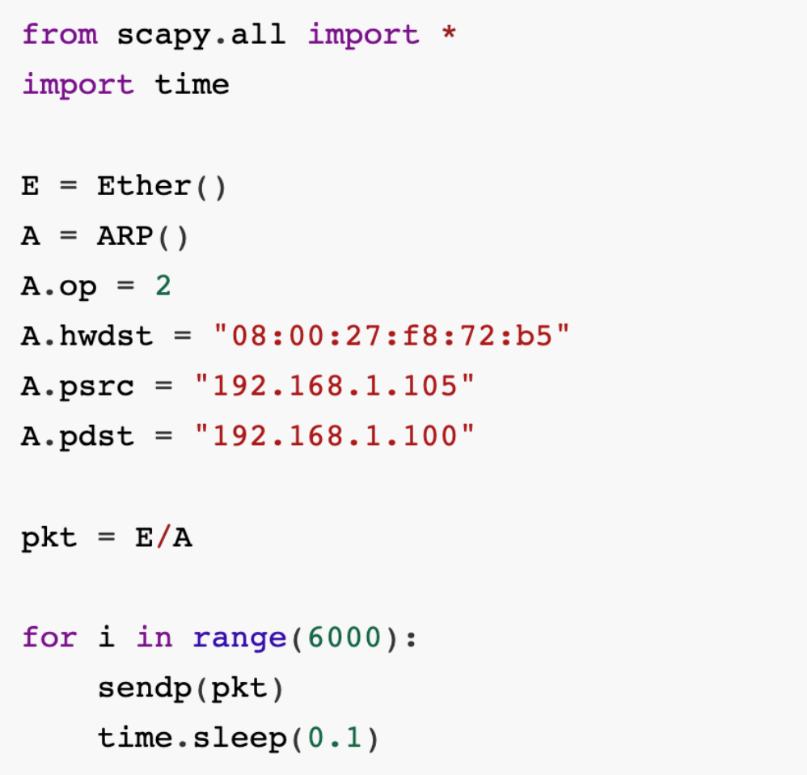


运行脚本后查看



已经成功污染ARP缓存表

Task 1B

  
不停地发送ARP响应报文

运行结果与task1A相同

TASK1C

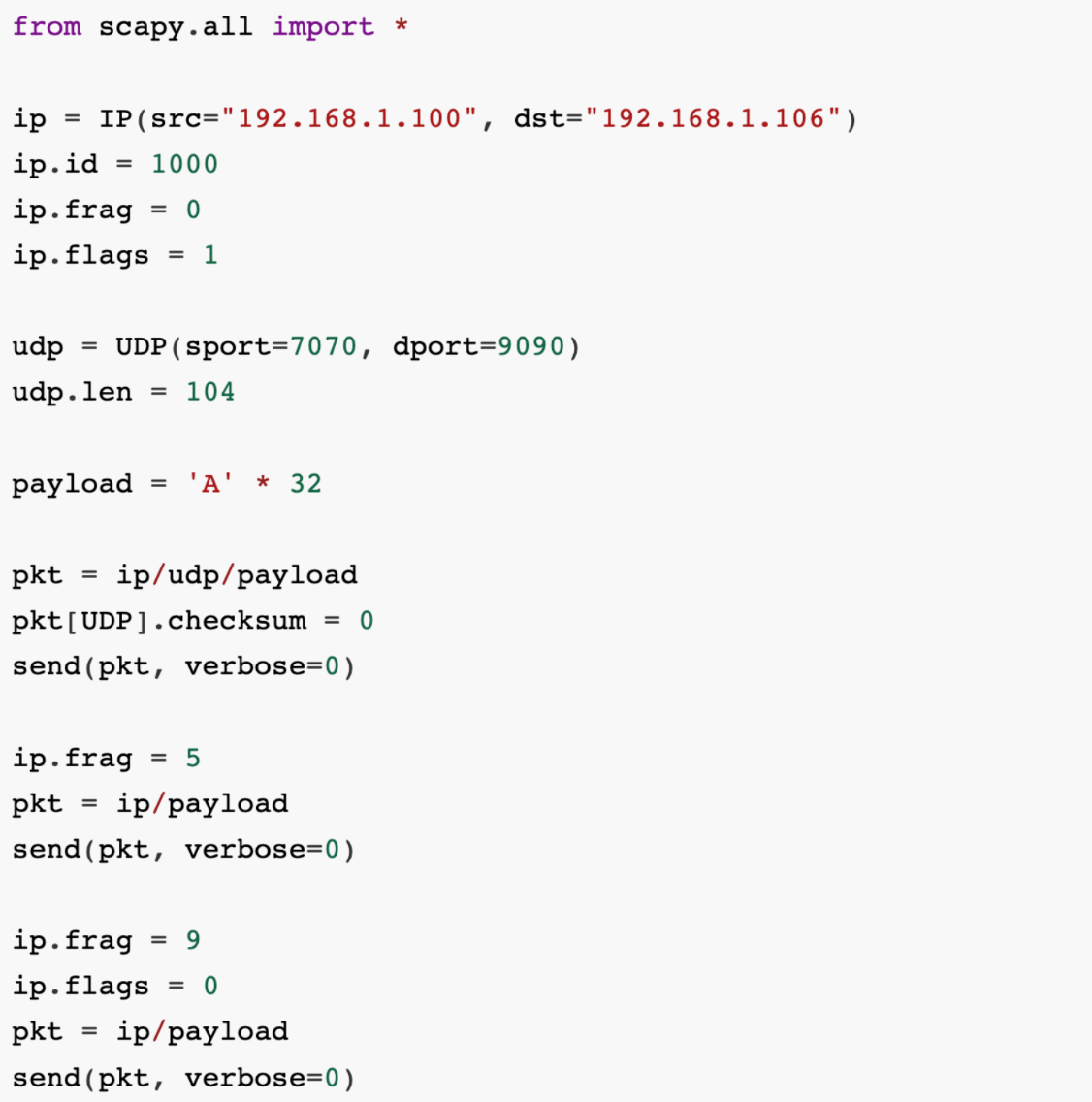


不停地广播ARP gratuitous报文，将源宿地址都设为要污染的IP地址，宿MAC地址设置为ff-ff-ff-ff-ff-ff，源MAC地址设为攻击者的MAC地址。

运行结果与task1A相同

LAB3

Task 1.a:



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

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

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

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）。