Lab 4 57118226舒钰淇

Task 1: ARP Cache Poisoning

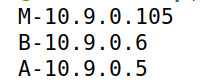


图1.1 docker主机名与IP信息

Task 1.A (using ARP request)

编写如下程序，在恶意主机M上伪造发送一个ARP请求报文，将源地址改为主机B的地址，由于主机A的ARP缓存表中没有B的信息，因此会将报文中的源mac地址当做主机B的MAC地址写入缓存表。

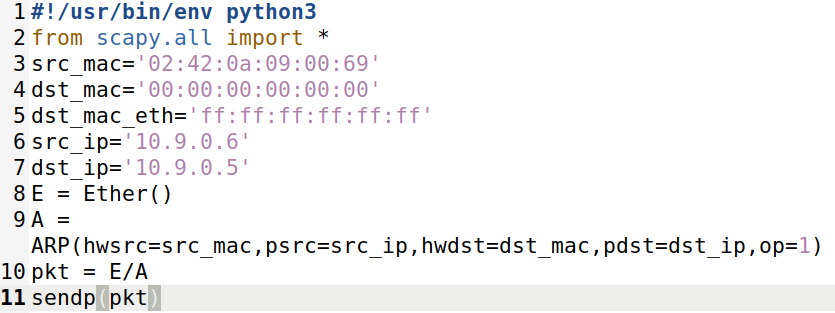


图1.2 task1A.py

如图所示，攻击成功

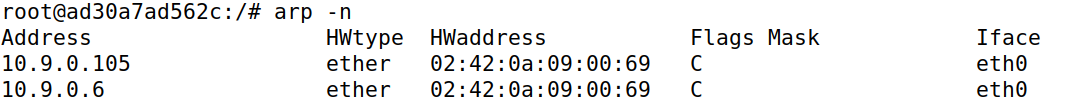


图1.3 主机A上的ARP缓存表

Task 1.B (using ARP reply)

编写如下所示的ARP rely伪造报文发送程序

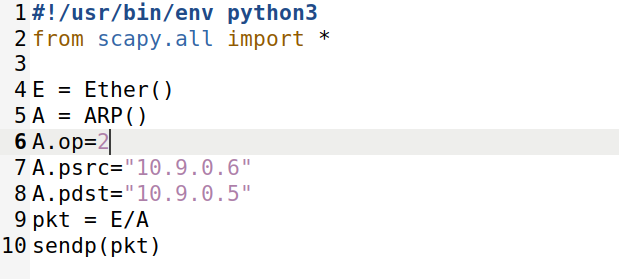


图1.4 task1b.py

先在主机A中无ARP缓存信息时运行程序，可以看到缓存表不受影响。



图1.5 主机A的ARP缓存为空

先用主机B ping 主机A,写入MAC地址信息后再运行攻击程序，可以看到攻击成功。

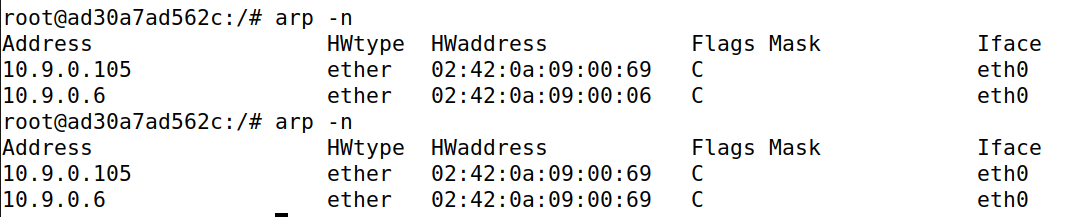


图1.6 主机A的ARP缓存被修改

Task 1C (using ARP gratuitous message)

编写如下程序，发送ARP gratuitous报文

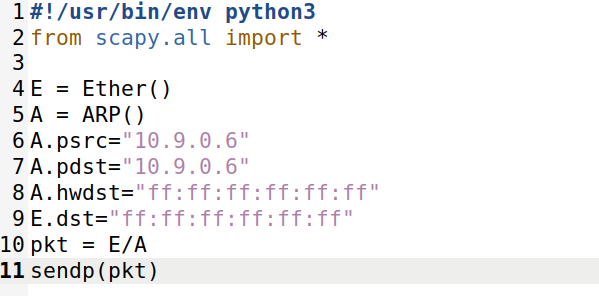


图1.7 task1c.py

在不同条件下进行攻击，可以看到与task1b相同，只有受害主机已经存在相关信息时才能对ARP缓存进行污染



图1.8 主机A的ARP缓存为空

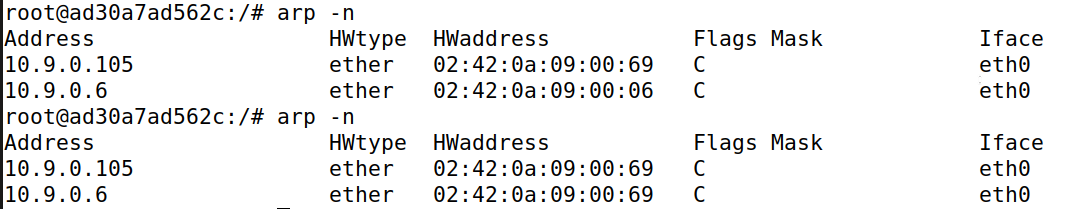


图1.9 主机A的ARP缓存被修改

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

分别编写两个程序，对主机A和B同时进行攻击，为了确保ARP缓存保持在所需状态（使M为中间人），需要M持续发送攻击报文。

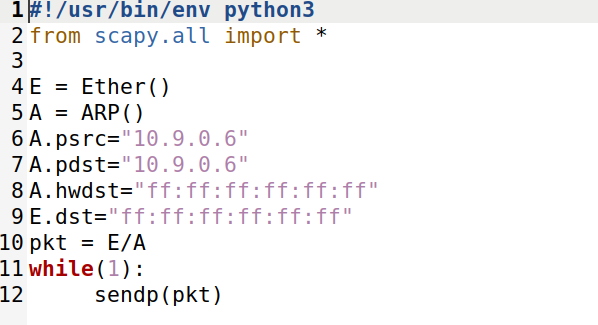


图2.1 对A进行攻击的程序

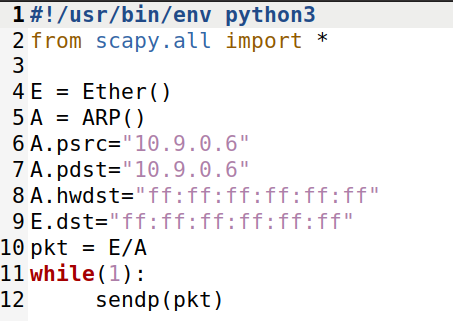


图2.2 对B攻击的程序

关闭M的转发功能后，在A和B上互相ping，都无响应，wireshark上也显示ICMP请求报文报文没有收到response

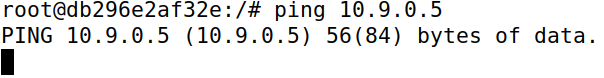
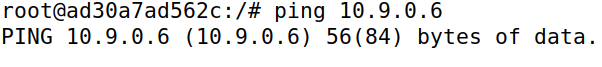


图2.3 B ping A

图2.4 A ping B

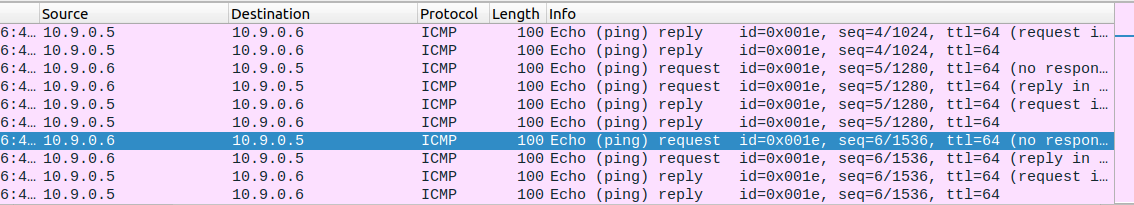
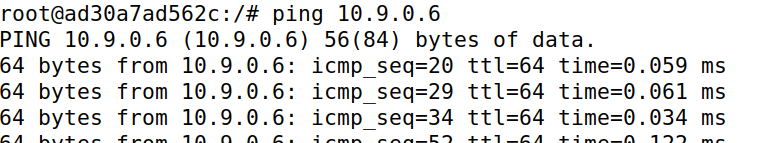


图2.5 wireshark抓到的ICMP报文

打开M上的转发功能，两台主机可以ping通



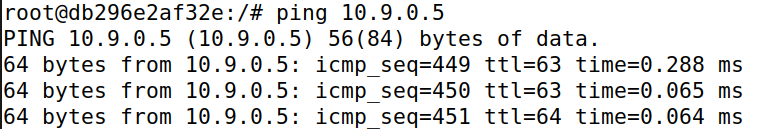


图2.6 两台主机互相ping通

编写如下代码，当源地址为A，宿地址为B时，将其中的数据全部改为字符Z

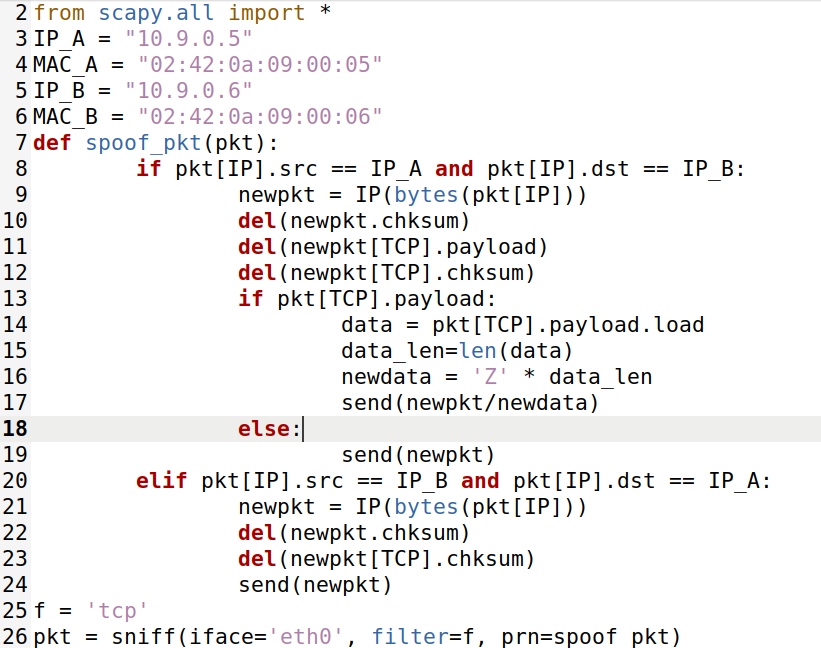


图2.7 task2c.py

先开启M的转发功能，在主机A上telnet主机B后，将转发功能关闭并运行上述代码，在主机A的telnet界面输入的字符都变为Z。



图2.8 主机A的telnet输入都变为Z

使用wireshark抓包，可以看到由A发向M的数据部分为a，而M发向B的数据部分变为Z。

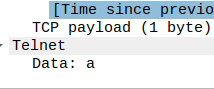


图2.9 A发向M的数据部分



图2.10 M发向B的数据部分

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

重复上一个task的步骤，A与B之间用netcat进行连接，在A上输入SYQ和回车键后，B中显示的内容为ZZZ

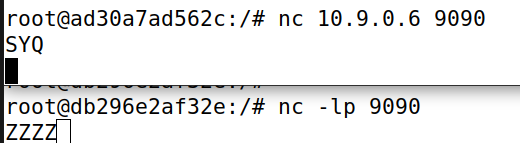


图3.1 netcat连接被中间人攻击