**实验三 Packet Sniffing and Spoofing Lab**

**ARP Cache Poisoning Attack Lab**

**IP/ICMP Attacks Lab**

**实验报告**

57117202 朱惟璇

**一、Packet Sniffing and Spoofing Lab**

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

可以使用许多工具进行Sniff和Spoof，但是大多数工具仅提供固定功能。Scapy有所不同：它不仅可以用作工具，还可以用作构建其他监听和欺骗工具的构件，即可以将Scapy功能集成到自己的程序中。在这组任务中将为每个任务使用Scapy。当前版本的SEED VM可能没有为Python3安装Scapy。使用以下命令为Pyhon3安装Scapy。

$ sudo pip3 install scapy

要使用Scapy，可以编写一个Python程序，然后使用Python执行该程序。请参见以下示例。使用root特权运行Python，因为该特权是欺骗数据包所必需的。在程序开始（第➀行）时，导入Scapy的所有模块。

可以进入Python的交互模式，然后在python提示符下一行一行地运行程序。如果需要在实验中频繁更改代码，这将更加方便。

$ sudo python3

>>> from scapy.all import \*

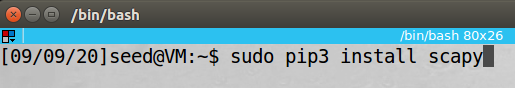
>>> a = IP()

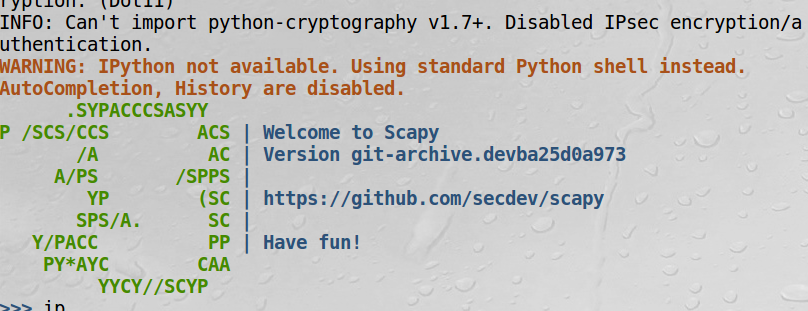
>>> a.show()

###[ IP ]###

version = 4

ihl = None





![C:\Users\zhuweixuan\AppData\Roaming\Tencent\Users\467718102\QQ\WinTemp\RichOle\R}GQR7](RIAYH(5W@VLTH8N.png](data:image/png;base64,)

（做这一步的时候虚拟机总是报错，花了很多时间差点心态崩了，幸好最后还是安装出来了。）

**1.1 Task 1.1: Sniffing Packets**

Wireshark是最受欢迎的嗅探工具，并且易于使用，将在整个实验中使用它。但是，很难将Wireshark用作构建其他工具的基础。为此目的使用Scapy。该任务的目的是学习如何使用Scapy在Python程序中进行数据包嗅探，以下提供了示例代码。

**Task 1.1A.**上面的程序嗅探数据包。对于每个捕获的数据包，将调用回调函数print\_pkt（）；此功能将打印出有关数据包的一些信息。以root特权运行该程序，并演示确实可以捕获数据包。之后，再次运行该程序，但不使用root特权；描述和解释观察结果。

// Make the program executable

$ chmod a+x sniffer.py

// Run the program with the root privilege

$ sudo ./sniffer.py

// Run the program without the root privilege

$ sniffer.py

**Task 1.1B.**通常，当嗅探数据包时，只对某些类型的数据包感兴趣。可以通过在嗅探中设置过滤器来做到。Scapy的过滤器使用BPF（伯克利包过滤器）语法；可以从互联网上找到BPF手册。请设置以下过滤器，然后再次演示嗅探器程序（每个过滤器应单独设置）：

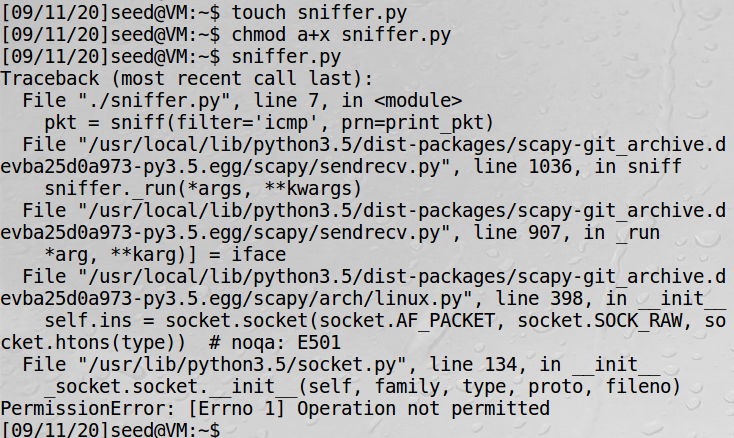
仅捕获ICMP数据包。

捕获来自特定IP且目标端口号为23的任何TCP数据包。

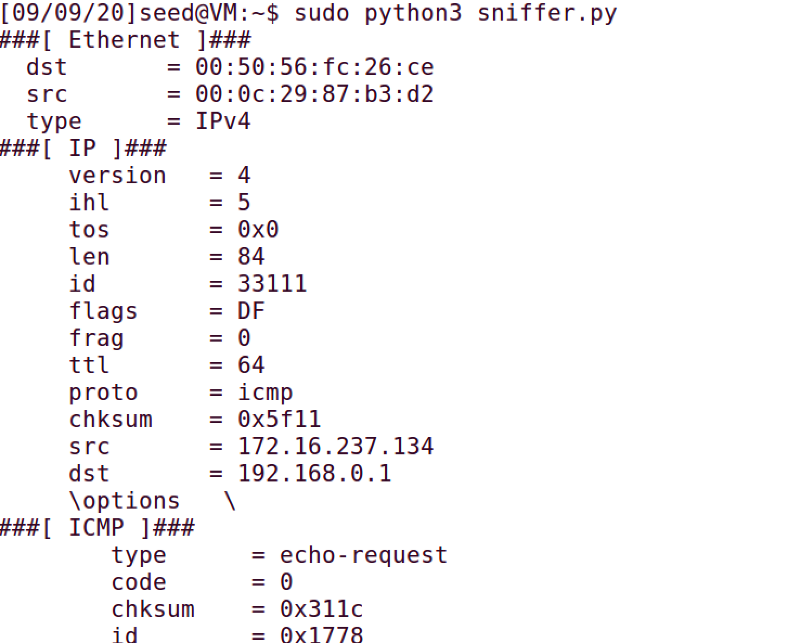
捕获数据包来自或进入特定子网。可以选择任何子网，例如128.230.0.0/16；不应选择VM所连接的子网。

Task 1.1A.

如果不使用root权限会被阻拦：



使用root权限可以看到如下的结果：



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)

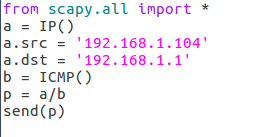
**1.2 Task 1.2: Spoofing ICMP Packets**

作为数据包欺骗工具，Scapy允许将IP数据包的字段设置为任意值。该任务的目的是欺骗具有任意源IP地址的IP数据包。将欺骗ICMP回显请求数据包，并将其发送到同一网络上的另一个VM。将使用Wireshark来观察接收者是否接受请求。如果被接受，则回显应答包将被发送到欺骗的IP地址。以下代码显示了如何欺骗ICMP数据包的示例。

在上面的代码中，第①行从IP类创建一个IP对象。为每个IP标头字段定义一个类属性。可以使用ls（a）或ls（IP）查看所有属性名称/值。还可以使用a.show（）和IP.show（）进行相同的操作。➁行显示了如何设置目标IP地址字段。如果未设置字段，则将使用默认值。

第➂行创建一个ICMP对象，默认类型为echo request。在➃行中，将a和b堆叠在一起以形成一个新对象。/运算符被IP类重载，因此不再表示除法；相反，这意味着将b添加为a的有效负载字段并相应地修改a的字段。结果得到一个代表ICMP数据包的新对象。现在可以在第➄行中使用send（）发送此数据包。请对示例代码进行任何必要的更改，然后证明可以欺骗具有任意源IP地址的ICMP回显请求数据包。

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：



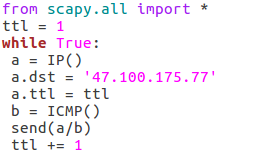


可以看到伪装成功。

**1.3 Task 1.3: Traceroute**

该任务的目的是使用Scapy来估计VM与选定目标之间的距离（以路由器数量为单位）。这基本上是traceroute工具实现的。在此任务中将编写自己的工具。这个想法非常简单：只需将数据包（任何类型）发送到目的地，其生存时间（TTL）字段首先设置为1。该数据包将被第一个路由器丢弃，第一个路由器将向我们发送ICMP错误消息，告知已经超过了生存时间。这就是获取第一个路由器的IP地址的方式。然后将TTL字段增加到2，发送另一个数据包，并获得第二个路由器的IP地址。重复此过程，直到数据包最终到达目的地。应当注意的是，该实验仅获得了估计的结果，因为从理论上讲，并非所有这些数据包都采用相同的路由（但实际上，它们可能会在很短的时间内）。以下代码显示了该过程的一轮操作。如果是经验丰富的Python程序员，则可以编写工具来自动执行整个过程。如果不熟悉Python编程，则可以通过在每个回合中手动更改TTL字段来实现，并根据Wireshark的观察记录IP地址。只要得到结果，任何一种方法都是可以接受的。

新建traceroute.py，代码如图：



结果如图，第一个回应在ttl=14时出现，所以经过14个路由器。

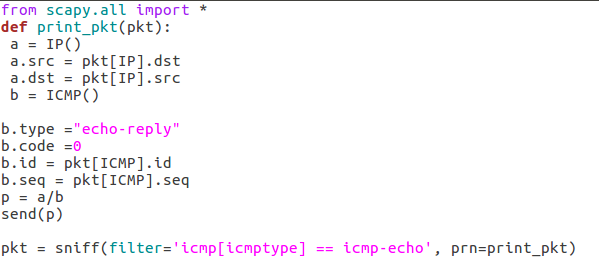




**1.4 Task 1.4: Sniffing and-then Spoofing**

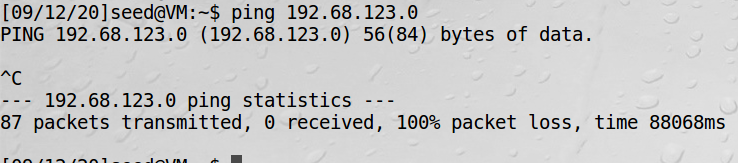
在此任务中将结合嗅探和欺骗技术来实现以下sniff-and-then-spoof程序。这需要在同一LAN上有两个VM。从VM A ping IPX，这将生成ICMP回显请求数据包。如果X仍然存在，则ping程序将收到回显答复，并打印出响应。sniff-and-then-spoof程序在VM B上运行，VM B通过数据包嗅探监视LAN。每当它看到ICMP回显请求时，无论目标IP地址是什么，程序都应立即使用数据包欺骗技术发出回显应答。因此，无论机器X是否处于活动状态，ping程序都将始终收到答复，指示X处于活动状态。需要使用Scapy来完成此任务。在报告中需要提供证据来证明技术有效。

编写的代码如下：

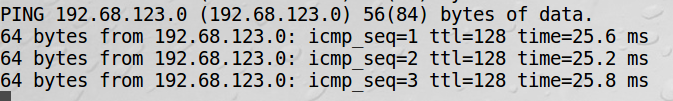


在VM 1上运行以上程序，并且在另一个VM 2上运行ping 192.68.123.0,可以得到以下打印结果。与正常ping 的结果相比较可以发现VMB 上会得到两次结果返回，一次会被截断（truncated）,第二次（DUP!）表明一个ICMP请求包得到了重复的回复。

一开始ping的结果：



正常ping的结果：



**二、ARP Cache Poisoning Attack Lab**

**1 Task 1: ARP Cache Poisoning**

此任务的目的是使用数据包欺骗对目标发起ARP缓存中毒攻击，以便当两个受害机器A和B尝试相互通信时，攻击者将截获它们的数据包，攻击者可以进行更改攻击，然后就可以成为介于A和B之间的中间人。这称为中间人（MITM）攻击。在本实验中使用ARP cahce中毒来进行MITM攻击。

以下代码框架显示了如何使用Scapy构造ARP数据包。

上面的程序构造并发送一个ARP数据包。请设置必要的属性名称/值以定义自己的ARP数据包。可以使用ls（ARP）来查看ARP类的属性名称。如果未设置字段，那么将使用默认值（请参见输出的第三列）。

在此任务中，我们有三个VM，A，B和M。我们要攻击A的ARP缓存，以便在A的ARP缓存中获得以下结果。

B’s IP address --> M’s MAC address

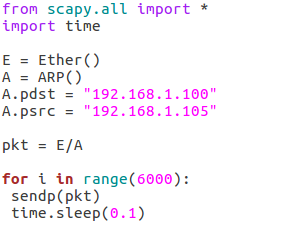
进行ARP缓存中毒攻击的方法有很多。需要尝试以下三种方法，并报告每种方法是否有效。

* **Task 1A（使用ARP请求）。**在主机M上，构造ARP请求数据包并发送到主机A。检查M的MAC地址是否已映射到A的ARP缓存中B的IP地址。
* **Task 1B（使用ARP答复）。**在主机M上，构造一个ARP应答数据包，然后发送到主机A。检查M的MAC地址是否已映射到A的ARP缓存中B的IP地址。
* **Task 1C（使用ARP免费消息）。**在主机M上，构造ARP免费数据包。ARP免费数据包是一种特殊的ARP请求数据包。当主机需要更新其他所有计算机的ARP缓存上的过时信息时，可以使用该功能。免费ARP报文具有以下特点：
* 源IP地址和目标IP地址相同，它们是发出免费ARP的主机的IP地址。
* ARP标头和以太网标头中的目标MAC地址都是广播MAC地址（ff：ff：ff：ff：ff：ff：ff）。
* 预计没有答复。

首先查看arp缓存表：可以得到虚拟机MAC地址为08-00-27-f8-72-b5，想要进行污染的地址为192.168.1.105。

**Task 1A：**

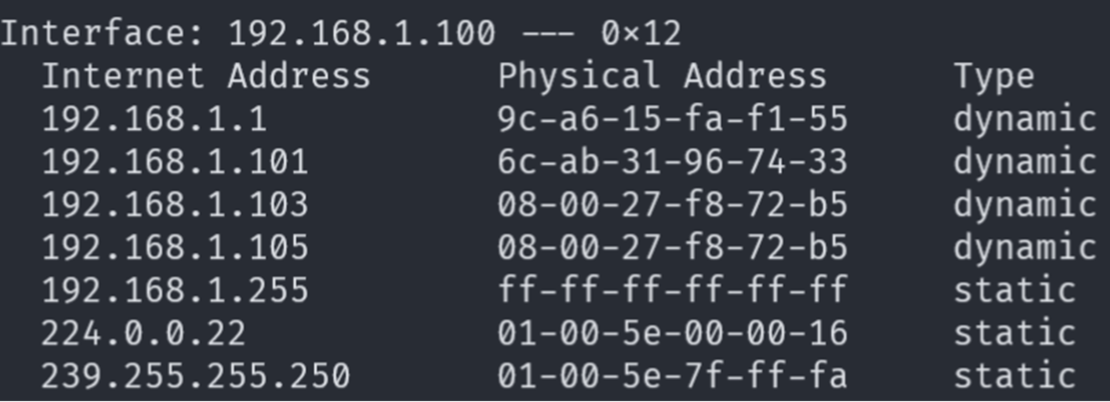
编写task1a.py程序如下：



初始ARP表如下：

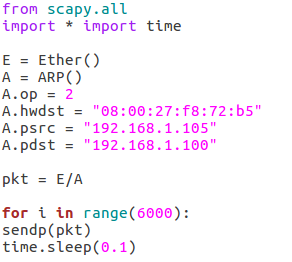


运行task1a.py后如下，对应的192168.1.105地址被污染：



**Task 1B：**

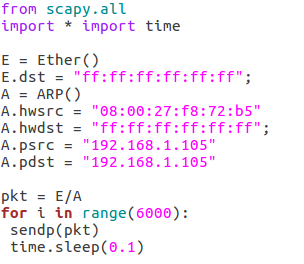
新建task1b.py，编写程序：



向目标主机发送ARP响应，被污染的ip对应的MAC地址为虚拟机的MAC地址，得到了与task1A一样的结果。

**Task 1C：**

和前面一样，新建task1c.py，编写程序：



不停地广播ARP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。

运行结果与前两个实验相同。

**三、IP/ICMP Attacks Lab**

**1 Tasks 1: IP Fragmentation**此任务需要两个VM。它们应连接到同一网络，以便它们可以相互通信。

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

在此任务中，需要构造UDP数据包并将其发送到UDP服务器。可以使用“nc -lu 9090”启动UDP服务器。无需构建单个IP数据包，而是需要将数据包分成3个片段，每个片段包含32个字节的数据（第一个片段包含8个字节的UDP标头和32个字节的数据）。如果一切正确，服务器将总共显示96字节的数据。以下是用于构建第一个片段的示例代码。

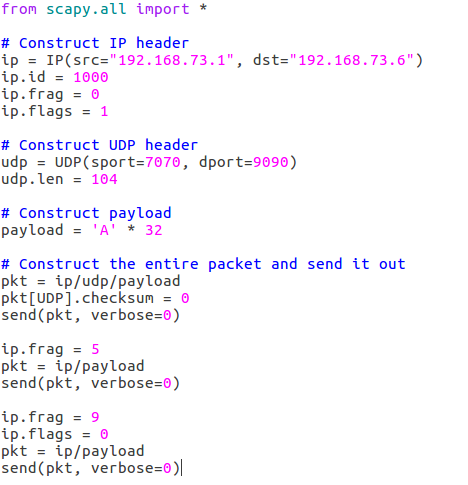
请注意，UDP校验和字段需要正确设置。如果不设置此字段，Scapy将计算校验和，但是此校验和将仅基于第一个片段中的数据，这是不正确的。如果将校验和字段设置为零，Scapy将不理会它。此外，如果接收方在校验和字段中看到零，则接收方将不会验证UDP校验和，因为在UDP中，校验和验证是可选的。

如果使用Wireshark观察流量，则还应注意，默认情况下，Wireshark会在最后一个片段数据包中重组片段，并将其显示为完整的IP / UDP数据包。要更改该行为，应该在Wireshark首选项中禁用IP片段重组。单击以下菜单序列：编辑→首选项；单击“协议”下拉菜单，找到并单击“IPv4”。取消选中“重组分段的IPv4数据报”选项。

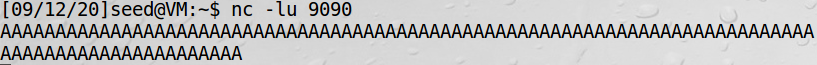
首先将UDP报文分片：

* UDP的总长度：头部8字节+载荷96字节，共104字节；
* 第一片IP报文片偏移frag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，其余不变，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。

编写程序如下：



在虚拟机1中运行脚本，在虚拟机2中使用命令sudo nc -lu 9090，可以接收到96个A。



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

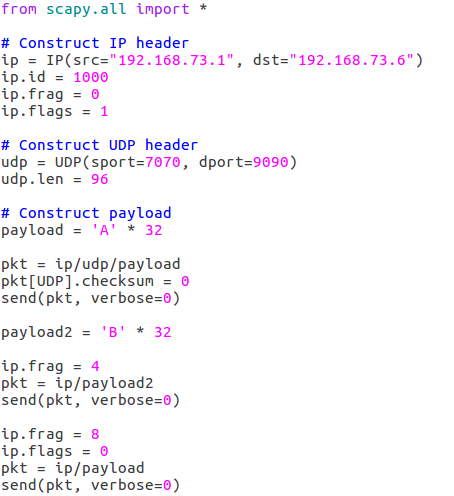
与任务1.a相似，需要构造3个片段以将数据发送到UDP服务器。每个片段的大小取决于自己。此任务的目的是创建重叠的片段。特别是前两个片段应重叠。请使用实验来说明发生重叠时会发生什么。请分别尝试以下重叠方案：

* 第一个片段的末尾和第二个片段的开始应有K字节重叠，即，第一个片段中的最后K个字节的数据应与第二个片段中的前K个字节的数据具有相同的偏移量。K的值由学生决定（K应该大于零且小于两个片段的大小）。在报告中，应指出他们的K值是多少。
* 第二个片段完全包含在第一个片段中。第二个片段的大小必须小于第一个片段（它们不能相等）。

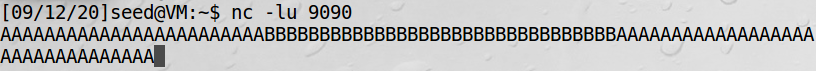
请尝试两个不同的命令：（1）首先发送第一个片段，（2）首先发送第二个片段。请报告结果是否相同。

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。接着把第二片报文的载荷中A全改为B。

修改后的程序如下：



运行程序，虚拟机2中收到的前24字符是A，中间有32个B，然后又有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。

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

众所周知，IP数据包的最大大小为216个八位位组，因为IP报头中的长度字段只有16位。但是，使用IP分段可以创建超出此限制的IP数据包。请构造一个这样的数据包，将其发送到UDP服务器，并查看服务器如何响应这种情况。请报告观察。

将IP头中len字段设为2的16次方，即0xFFFF，然后不断发送flags为1的报⽂，一直继续分片，使得分片长度超过2的16次方，最后设置flags=0结束报文。

理论上最终会使得接收方虚拟机的接收程序无法正常工作。nc命令产生的UDP服务器没有反应，可能是检查udp.len不正确未组装。

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

在此任务中，将使用机器A对机器B发起拒绝服务攻击。在攻击中，机器A向B发送许多不完整的IP数据包，即这些数据包由IP片段组成，但是一些碎片丢失了。所有这些不完整的IP数据包将保留在内核中，直到超时。这在潜在中可能导致内核提交大量内核内存。过去，这导致了对服务器的拒绝服务攻击。请尝试这种攻击并描述观察。

修改程序，不再发送第⼆⽚分⽚，只发送第⼀⽚、第三⽚分⽚，并不断改变id。



接收方缓存会把接收到的报文一直存储，直到内存被完全占满。形成拒绝服务攻击（DOS）。（但实际上实验的时候好像并没有明显的内存上涨？）

**四、总结**

这三章主要进行的是Sniff and Spoof、攻击APR和UDP服务器，难度都相当的大，甚至在一开始安装scapy的时候就出现了很多莫名其妙的错误。在实现攻击的过程中被卡了很多次，IP值经常不对又不知道为什么出了错。有时候还经常粗心大意把命令和文件名输入错误。深刻感受到自己对网络IP这块的了解和掌握还有很大的缺漏，甚至计算机网络的很多知识都已经忘记了，还需要继续努力学习。希望下一次的实验有进步。