

网络程序设计课程大作业

学 院  **计算机学院**

专 业  **信息安全**

学生姓名  **刘家瑞**

学 号  **2015301500261**

年 级  **2015级4班**

**Linux系统下编程实现仿NetCat网络工具箱（含TCP代理，网络流量嗅探，UDP抓包等功能）**

**实验目的：深入理解Linux操作系统下的Socket编程，客户端，服务器端编程框架以及原始套接字，IP层，ICMP的解析。**

**实验原理：**利用Python套接字编程，自己编写客户端服务器端的框架，并使用原始套接字来解析底层的数据包从而实现一个自己编写的简易netcat以及UDP抓包网络工具。

**实验环境：**Kali Linux ，Python 3 ，PyCharm ，VMware虚拟机

**简易网络工具包实现的功能：**

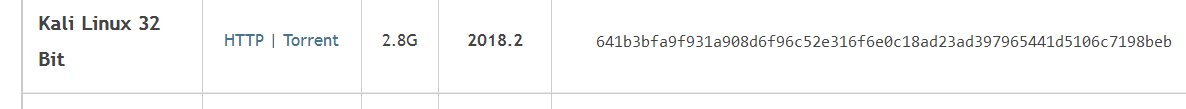
1. 命令行的帮助功能；
2. 实现命令行操作以及命令行解析的功能；
3. 对指定的端口进行监听；
4. 从标准输入发送指定的命令行命令，在服务器端执行后返回执行结果到到标准输出；
5. 从标准输入上传文件到服务器端；
6. 实现TCP代理，嗅探本地的流量

**实验步骤：**

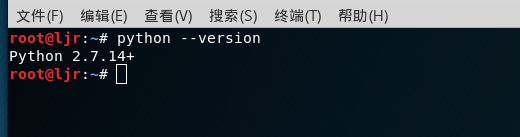
**仿netcat小工具myNetcat.py:**

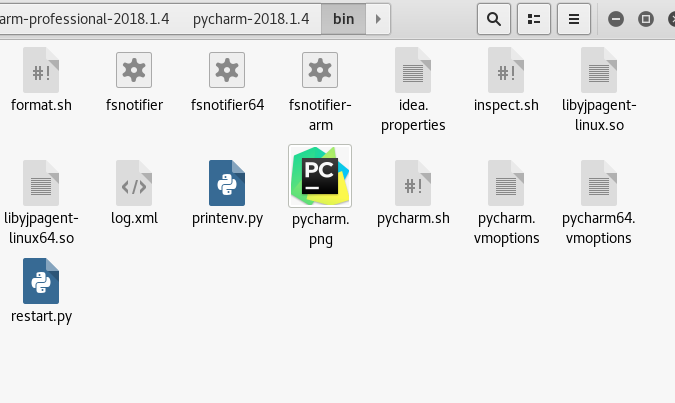
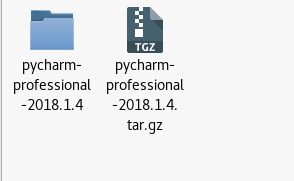
1. **安装编程环境：本次编程任务我们将才用Linux下的Python编程来实现，首先在自己的Windows 8操作系统上安装VMware12 虚拟机，安装步骤省略。**

**接着我们登陆Kali Linux官方网站下载32位最新版本的Kali Linux。**

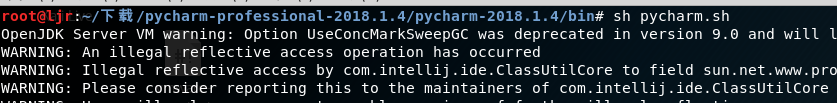


1. **从VMware12安装Kali Linux，系统的默认用户名为root。首先我们需要检查系统上是否安装了正确的Python版本。在命令行下输入**

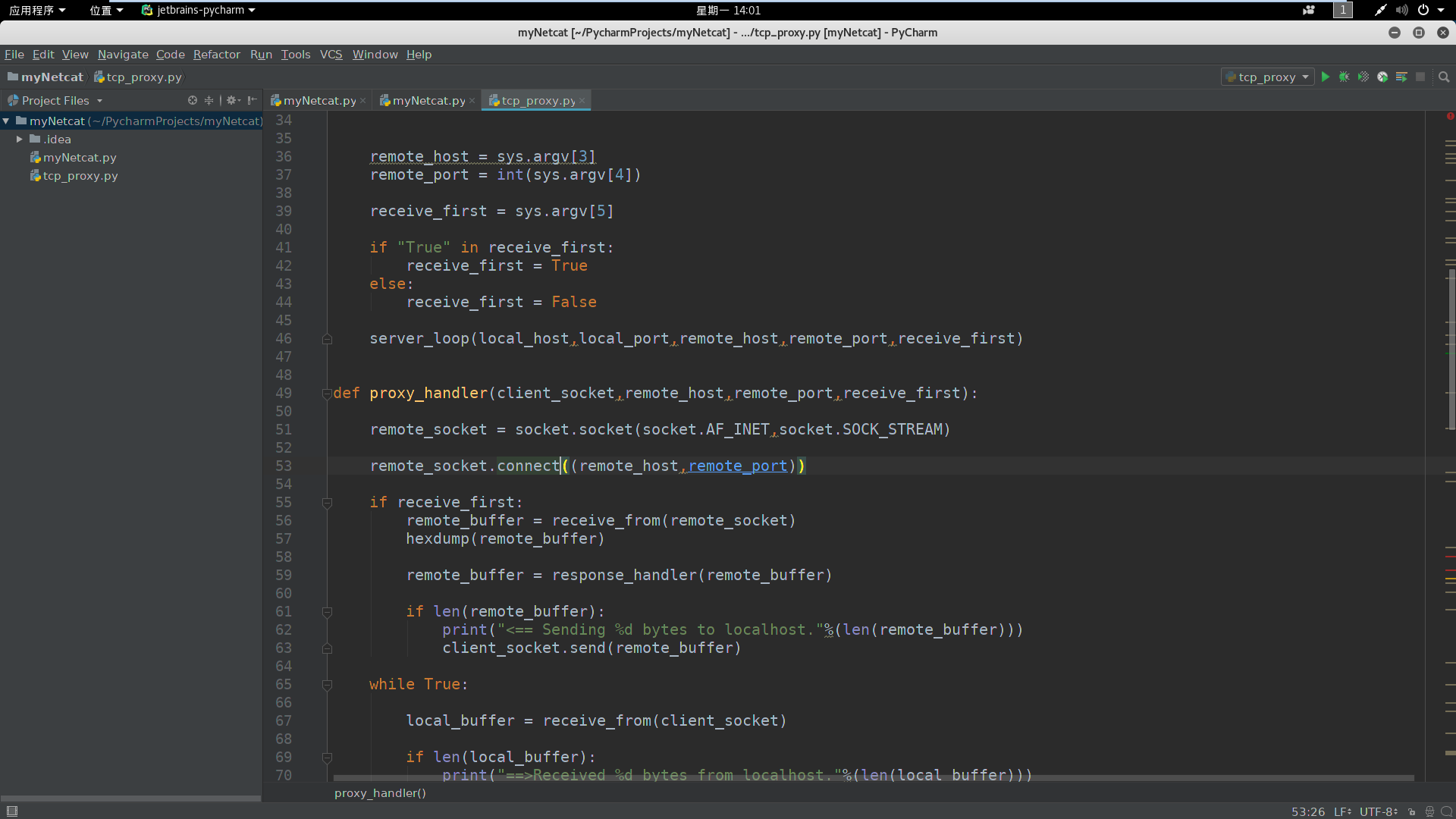


**可以看到我们已经安装了正确的python版本。接下来我们需要安装一个集成的python开发环境，我们将选择界面较为友好的Pycharm作为我们的编程IDE。我们需要从pycharm的官方网站上下载Linux版本的Pycharm，下载下来之后是一个压缩包，通过解压后我们进入/bin目录下**

**在此处运行命令行终端，并在终端中输入sh pycharm.sh**



**之后我们就可以成功的运行Pycharm了**



1. **接下来我们开始编写我们的网络工具包：在python网络编程中和C语言一样我们使用的最为核心的编程模块就是我们的socket模块。这个模块展示了快速创建TCP和UDP服务器及客户端、使用原始套接字等所必须的代码。使用的方法如下：**

**建立一个TCP客户端：client=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)**

**两个参数分别为协议族类型和套接字类型。**

**紧接着我们需要使用connect函数连接到客户端，参数中需要指定目的IP和目的端口号。最后我们可以使用send()来发送数据，使用recv()来接收数据，与C语言不同的是Python的socket并没有write() read() writev() readv()等函数，而是一律使用send()和recv()来从套接字中读取数据。**

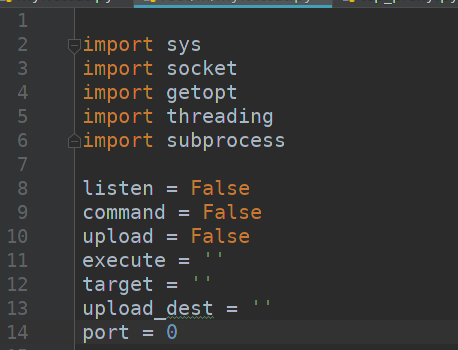
**建立一个TCP服务器端：**

**首先还是一样建立一个流式套接字，接着使用bind函数指定要侦听的端口和地址，接着使用listen()来设置侦听队列的长度，最后使用accept函数来接受连接，函数将返回一个新的用于连接的套接字描述符。**

**建立连接之后，我们将使用threading.Tread()函数来建立一个新的线程来处理到来的连接请求。**

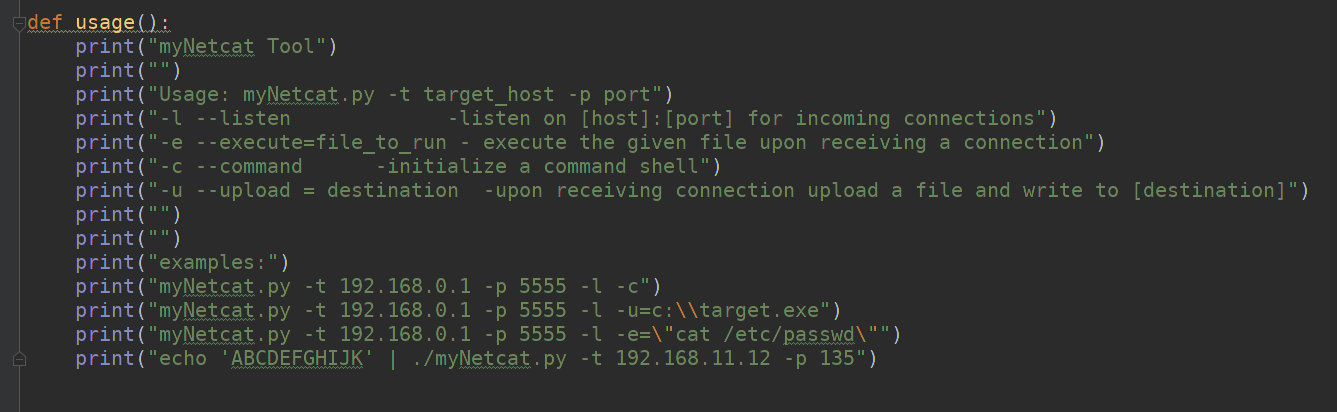
1. **了解了基本的客户端和服务器端的编程框架之后，我们开始真正的着手于我们的网络工具包。netcat被誉为网络界的“瑞士军刀”，它广泛应用于网络攻击，是黑客的常用工具，所以聪明的管理员都会将它从系统中移除。不止在一个场合，我们进入的服务器端没有安装netcat却安装了python。在这种情况下，需要创建一个简单的客户端和服务器用来传递想使用的文件，或者创建一个监听端让自己拥有控制命令行的操作权限。如果你是通过Web应用漏洞进入服务器的，那么在后台调用Python创建备用的控制通道就显得尤为适用，这样就不需要首先在目标机器上安装木马或后门了。**
2. **现在来开始编写代码：**

**首先导入一些必要的库文件和定义一些全局变量：**



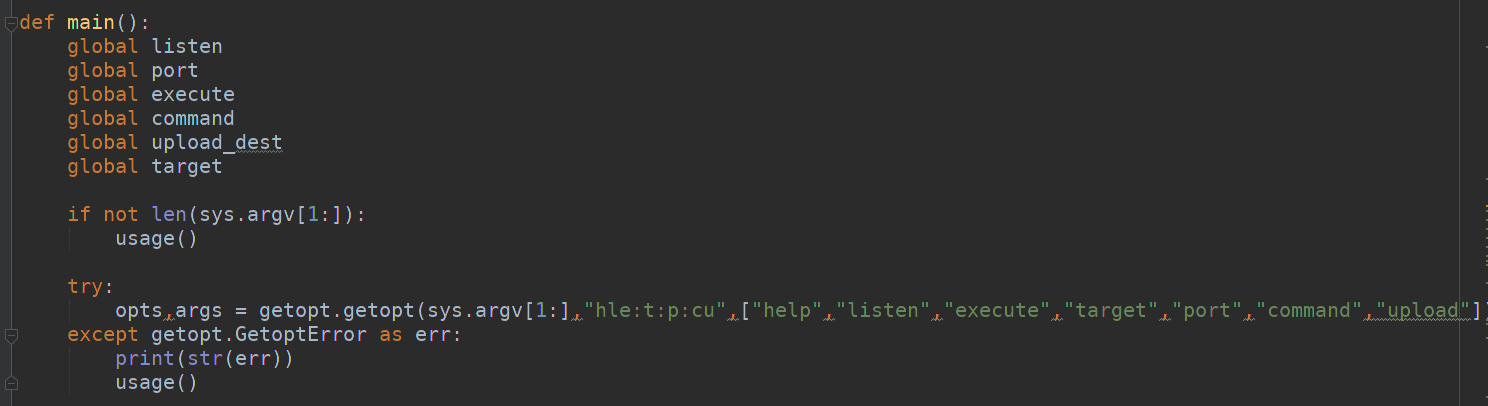
**listen表示是否开启侦听选项，comman表示是否开起一个命令行shell，upload表示是否上传文件。execute表示需要执行的命令“”。target和port表示目的IP地址和端口号。upload\_dest表示上传的目的地址路径。**

1. **接下来我们来编写使用指南函数：**



**在这个函数中我们主要实现的功能是，当用户在命令行shell中输入-h指令时，我们需要在命令行中打印出相应的帮助指南的相关内容。从代码中我们可以看到，命令行中各个指令所代表的含义。-t后接目的ip地址，-p后接目的端口号，输入过后将连接到指定的目的地。-l表示开启侦听模式，-e后接需要执行的指令并执行该指令。-c表示开启命令行的shell。-u后接上传到目的地址路径。**

1. **接下来我们来编写main（）函数并编写程序运行的逻辑框架：**



**首先我们先在main()函数中申明之前所定义的全局变量。接着我们检查函数的输入参数，如果没有输入参数的话，那么我们将调用usage()函数打印出帮助信息。接着使用try-except异常处理结构，使用getopt模块来处理命令行参数，下面来介绍一下getopt这个python下用来处理命令行参数的模块:首先我们大家都知道在C语言中main函数的基本格式是main(int argc,char \*\*argv),argc表示输入参数的数量，而argv表示一个指向参数的指针。在python**

**中也是一样，当我们在python命令行中输入如下的命令时**

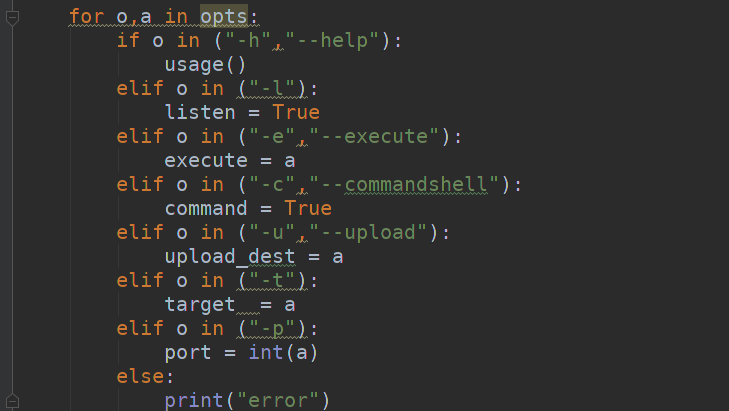
**python a.py 1 2 3**

**那么我们的argv[0 ]就是我们的程序名a.py ,sys.argv[1]为1，sys.argv[2]就是2，sys.argv[3]就是3.所以在程序中我们使用sys.argv[1:]来过滤掉第一项：函数getopt(args, shortopts, longopts = []) 参数args一般是sys.argv[1:],shortopts表示短格式 (-) ，longopts 表示长格式(--)。如这个例子：options,args = getopt.getopt(sys.argv[1:],"hp:i:",["help","ip=","port="])**

**“hp:i:”短格式-h 后面没有冒号：表示后面不带参数，p：和 i：后面有冒号表示后面需要参数。["help","ip=","port="]长格式 --- help后面没有等号=，表示后面不带参数，其他三个有=，表示后面需要参数。返回值 options 是个包含元祖的列表，每个元祖是分析出来的格式信息，比如 [('-i','127.0.0.1'),('-p','80')] ;args 是个列表，包含那些没有‘-’或‘--’的参数，比如：['55','66']。**

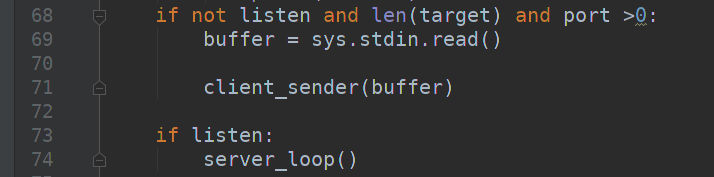
**所以我们可以知道在本程序中opts保存的是命令行参数的列表，而args保存的是一个不带命令只有参数的列表。最后在except中填写异常处理的相关函数。**

1. **接下来我们使用循环结构来一次解析我们之前使用getopt所读取的命令行信息。**

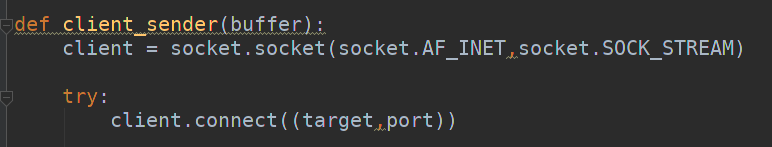


**a表示的是args列表中不带“-”或 “--”命令的参数信息，如目的IP:target和端口号port还有目的上传路径地址upload\_dest。**

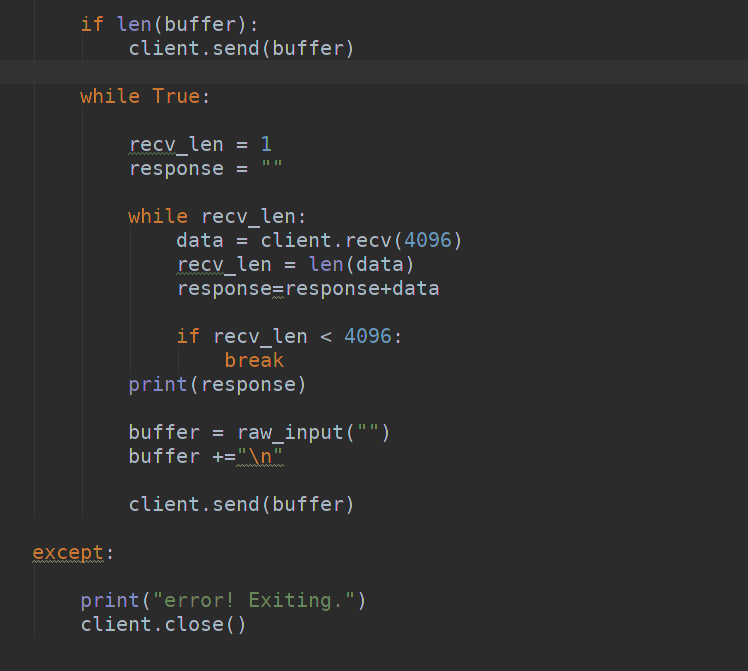
1. **接下来我们需要判断我们是否是作为服务器端进行监听。如果不是监听即listen=False且我们有相应的目的IP和目的端口号，这就意味着我们是作为客户端进行发送数据的，因此我们可以从标准输入中读取数据到缓冲区buffer中，最后调用client\_sender()函数将数据发送出去。而另一种情况是，如果我们是进行监听的服务器端，那么我们就要调用server\_loop()创建服务器端并进行循环侦听。**



1. **接下来我们来实现client\_sender(buffer)函数，首先我们这个函数的功能是作为客户端向服务器端发送数据，所以我们一开始先使用socket函数进行客户端的创建工作：**



**连接之后我们开始对缓冲区的数据进行判断，如果我们的缓冲区中有数据的话，那么我们使用send()函数将数据发送到目的端口。接下来我们轮询的等待数据的回传，我们使用recv()函数来接收来自服务器端的响应信息，并把它保存在data中，当接收的数据长的的值小于4096的时候，我们就停止接收跳出轮询接收，并开始等待更多的输入，接收到输入后再使用send()函数将数据发送出去。**



1. **接下来我们需要进行服务器段的创建，我们还是使用socket函数创建一个服务器端的套接字描述符并使用bind()函数绑定到指定的地址和端口，接着使用threading.Thread()函数创建一个线程来接受新到来的客户端连接。**



**这里说明一下threading.Thread()函数，target后的参数表示线程所挂载的处理函数，我们这里挂载的是client\_handler这个函数，args后接需要传入处理函数中的参数。**

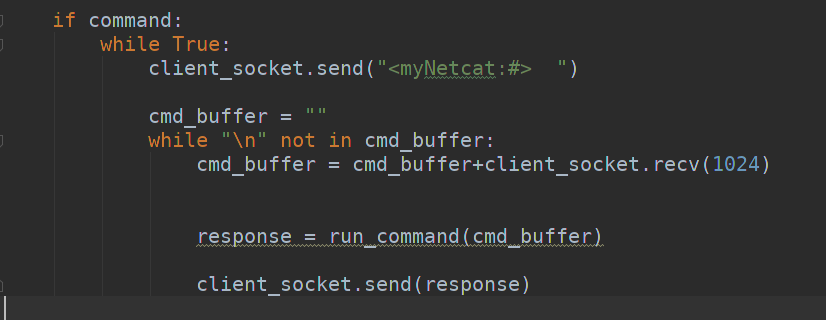
1. **接下来我们来编写client\_handler客户端连接处理函数：**

**首先我们检测我们的命令行中是否要求我们上传文件：，读取所有的字符并写下目标，利用while循环持续的读取数据直到没有符合的数据，接着我们使用异常处理结构try，except把我们接收的这些数据写入我们制定的文件中。**



**创建一个文件描述符fd，将数据写入指定的文件中。**

**接下来我们检查我们是否需要启动命令行shell，如果需要的话我们弹出一个窗口<myNetcat:#>以表示目前已经进入了命令行shell，接下来我们轮询的接收数据直到发现换行符表示命令结束，并将所有的命令都保存在cmd\_buffer中，之后调用run\_comman()函数执行输入的命令，最后使用send()将相应数据返回给客户端。**



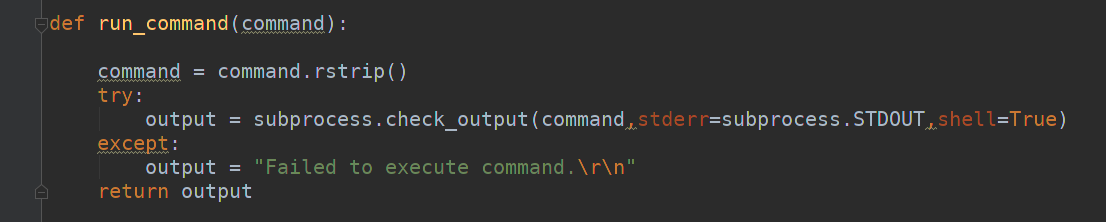
**我们还需要判断是否有-e选项，也就是是否需要我们执行一个命令：**



**这里直接调用run\_command()进行命令的执行。**

1. **最后我们来编写执行命令的函数run\_command，首先使用rstrip()方法去除命令的空格，接着使用subprocess库运行命令并将输出返回。**

**subprocess库提供了强大的进程创建接口，它可以提供给你多种于客户端程序交互的方法，在这里，我们运行了用户输入的命令，在目标的操作系统中运行，然后通过套接字连接将命令结果返回到我们控制的客户端。异常处理代码用来处理一般的错误并将错误信息回传，告诉你命令执行失败。**



1. **最后调用main()函数执行。**

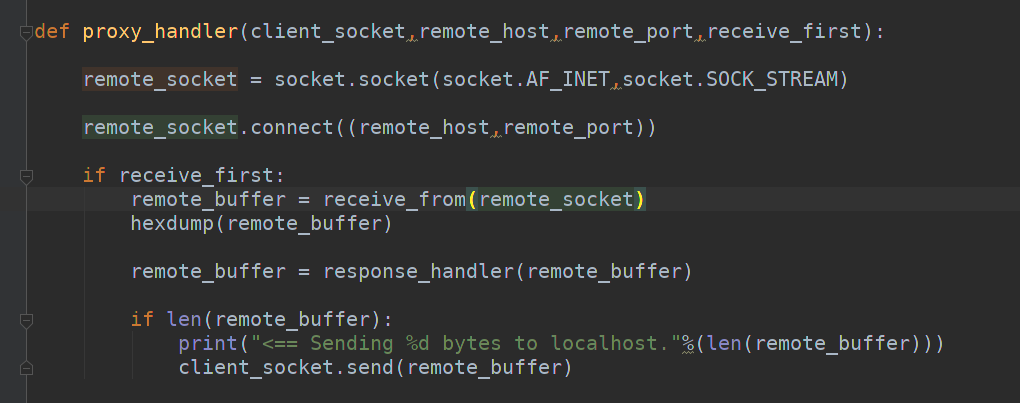


**TCP代理tcp\_proxy:**

1. **在端口侦听，上传文件，执行命令的基础上，我们的工具箱还很需要一个TCP代理，因为这个工具在网络攻击和网络安全领域中应用非常的广泛，它不仅可以将流量从一个主机转发给另一个主机，而且可以评估基于网络的软件，在企业级环境下进行渗透测试时你会使用它来嗅探本地的流量。还可以修改发送到应用的数据包。**
2. **接下来我们来编写代码：首先我们导入一些必要的库文件，接着编写server\_loop函数。像之前一样，我们需要使用socket创建一个服务器端的套接字并进行端口和地址的绑定，最后设置侦听队列并使用threading.Thread()函数创建一个线程，只不过在这里我们需要挂接的函数是用来处理与远程主机通信的函数，而传入的参数将会是与本地主机相连的套接字描述符，远程主机地址remote\_host，端口号remote\_port以及是否首先从远程主机接收数据这一个bool型变量receive\_first.**

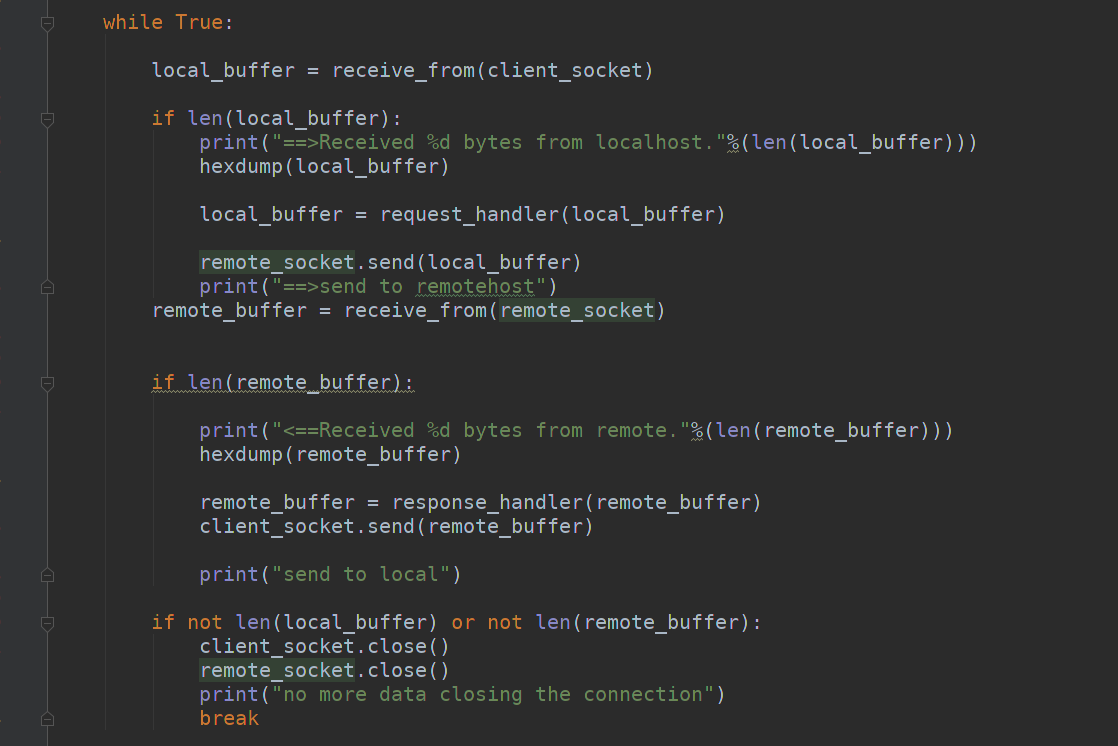


1. **接下来我们来编写挂载的远程主机处理函数proxy\_handler()，在这里我们首先使用socket创建一个与远程主机连接的套接字remote\_socket,接着我们将这个套接字连接到远程主机上。如果我们一定要首先从远程主机接收数据的话，那么我们就设置一个缓冲区remote\_buffer来接收从远程主机传回的信息，接下来使用一个十六进制导出函数hexdump()(这个函数我是从开源代码中获得的，并不是自己写的)。接着把缓冲区的数据发送给我们的响应处理函数，if len(remote\_buffer)如果我们有数据要传给本地客户端的话，那么久使用send()函数发送给本地客户端。**

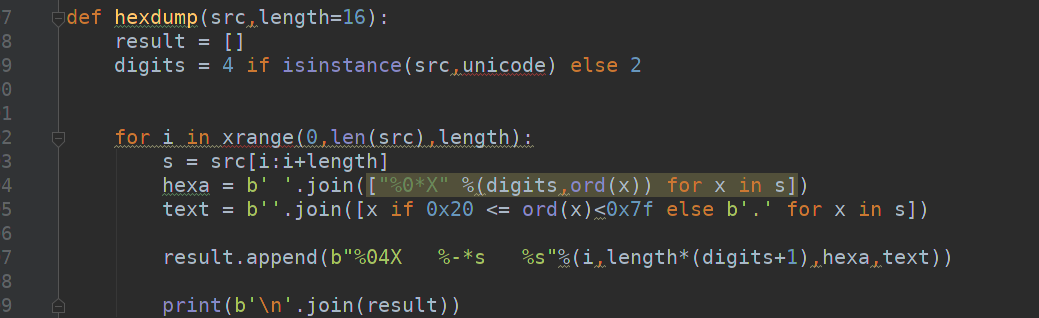


**下一步，我们从本地循环读取数据并把数据放到local\_buffer缓冲区中，紧着使用远程连接套接字remote\_socket将缓冲区中的数据发送给远程客户端。**

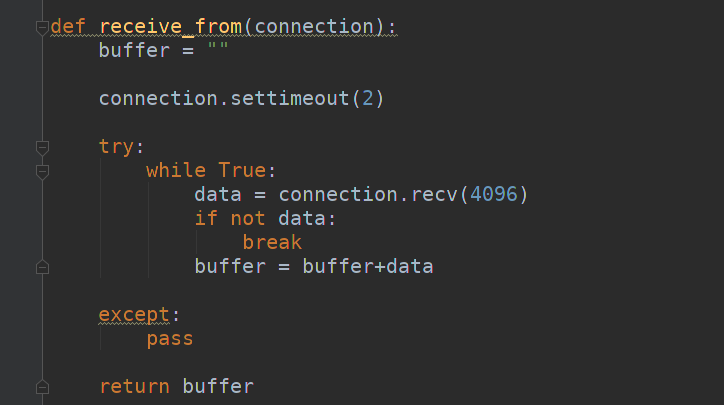
**接下来我们设置remote\_buffer用来监听并保存远程客户端的相应数据，最后将响应数据发送给本地的socket，client\_socket。如果两边都没有数据的话就关闭连接。**



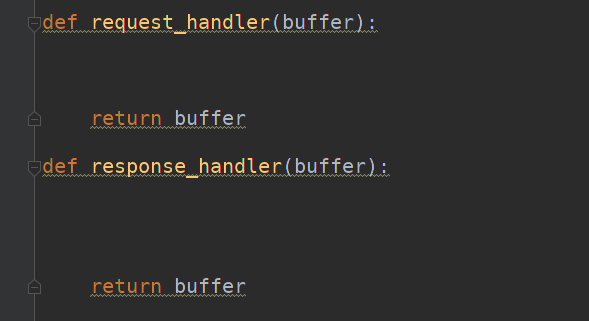
1. **接下来我们编写十六进制导出函数hexdump()，该函数借鉴了网上的开源代码:**



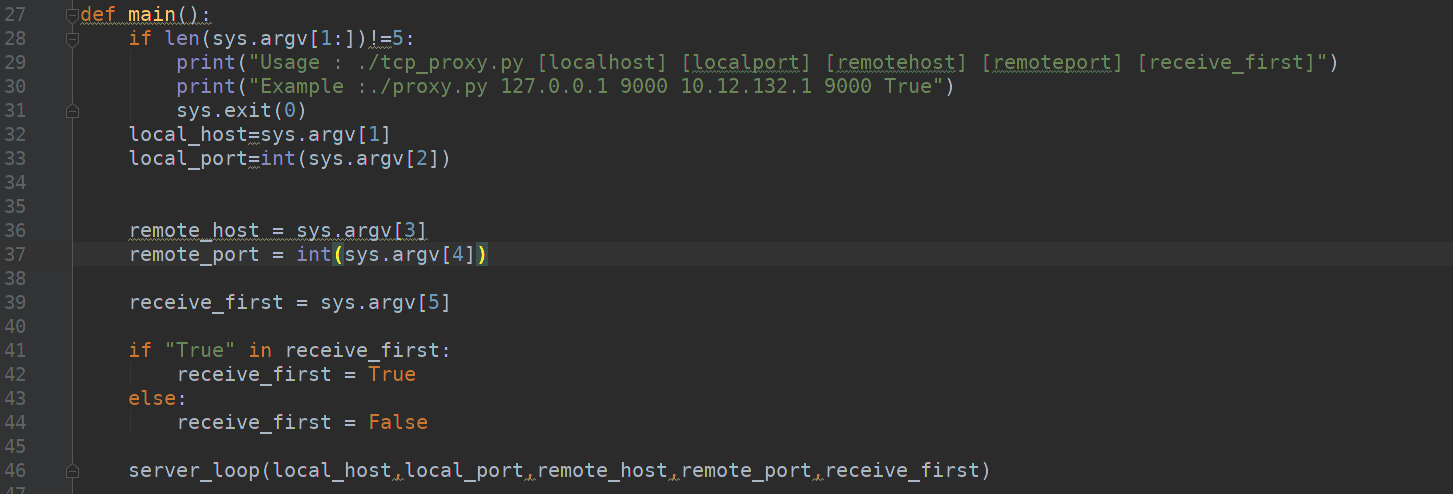
1. **我们还特意封装了一个接收函数，在其中调用了recv()函数并设置了两秒的超时，不过这取决于特定的情况，如果对于代理流量到其他国家或者网速慢的网络显得过于苛刻：**



1. **最后分别设置请求处理函数和相应处理函数，用来对目标是远程主机的请求和响应进行修改。在这个函数中我们可以修改数据包的内容，进行模糊测试任务，检测认证问题等这里由于测试环境的限制，我们没有填写具体的操作。**



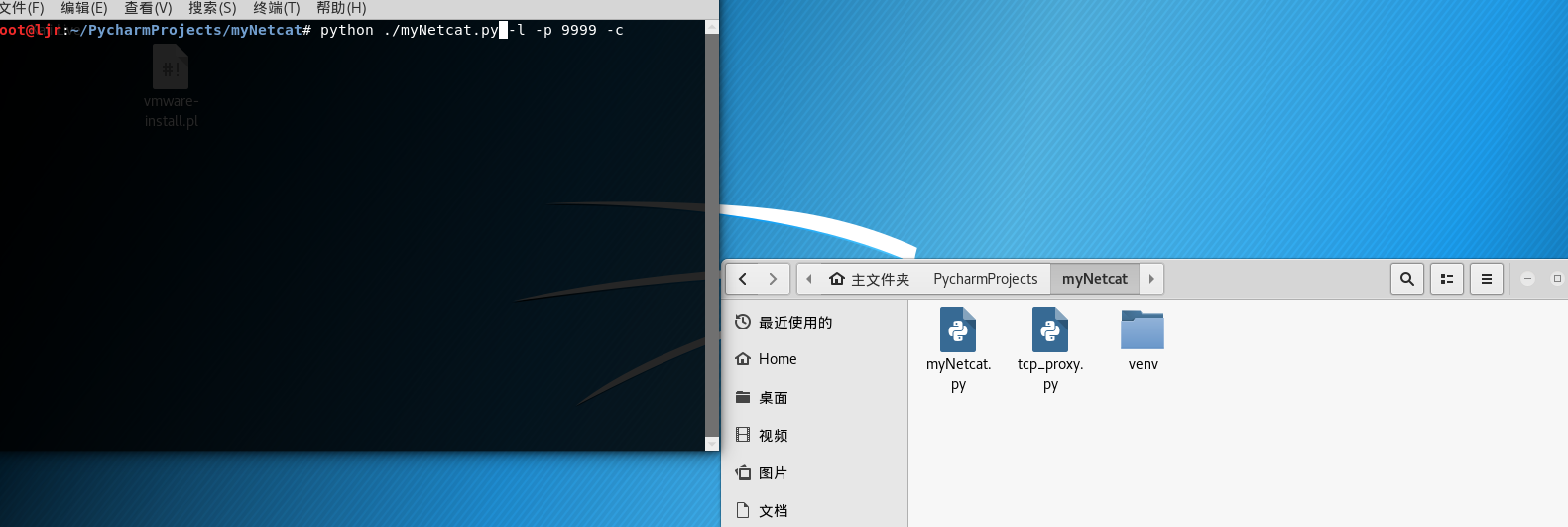
1. **最后编写main()函数，首先我们读入命令行参数进行解析，然后运行服务端的循环以监听连接请求。当一个新的请求到达时，我们将它交给proxy\_handler函数处理，此函数接收每一个比特的数据，然后发送到目标远程主机。**



**实验运行测试结果：**

**myNetcat运行测试：**

**首先进入程序文件目录下，在当前目录下开启终端，之后在当前目录下打开终端，输入如下的指令：python ./myNetcat –l –p 9999 –c建立一个侦听服务器等待连接。**

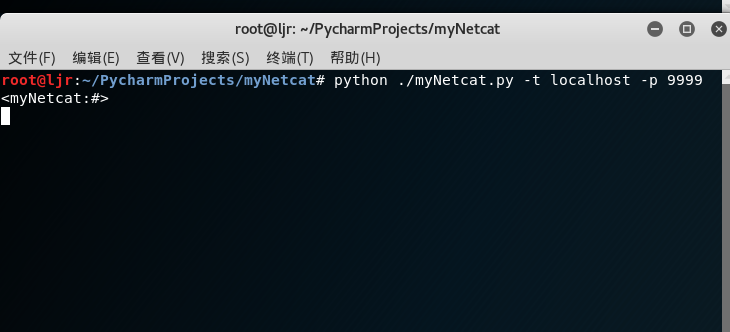


如下图表示启动了侦听：

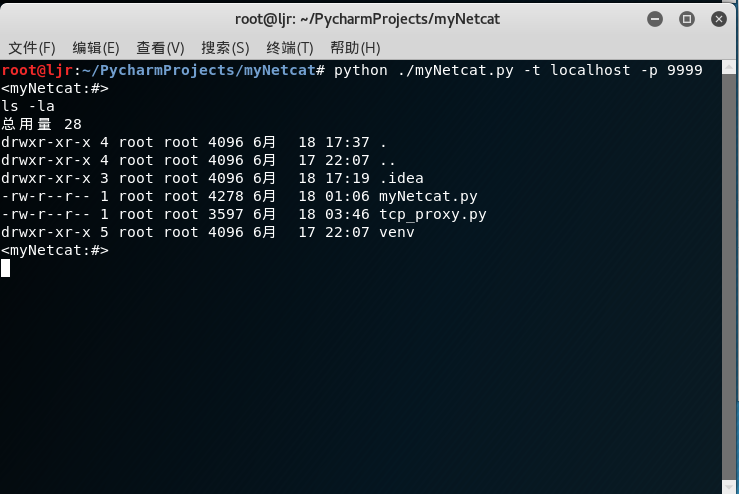


**现在我们启动另外一个终端，以客户端模式运行脚本。我们的脚本读取的是标准输入直到接收到EOF标志，按Ctrl+D组合键可以发送EOF命令。**

**我们打开另外一个终端输入客户端连接的命令python ./myNetcat.py –t localhost –p 9999连接之后进入命令行shell：**

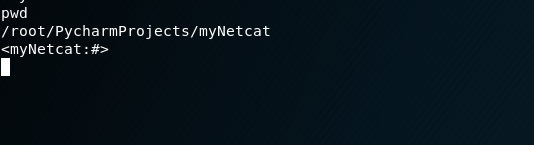


**如上图可以看到我们已经进入了服务器端的命令行shell下，在这运行命令行代码会在服务器端执行并返回结果到客户端。下面我们输入ls –la命令查看目录下的文件信息。**



**可以看到我们已经成功的在客户端运行了命令行代码并返回到了客户端。**

**接着我们再输入pwd命令进行测试：**



**结果都正确。**

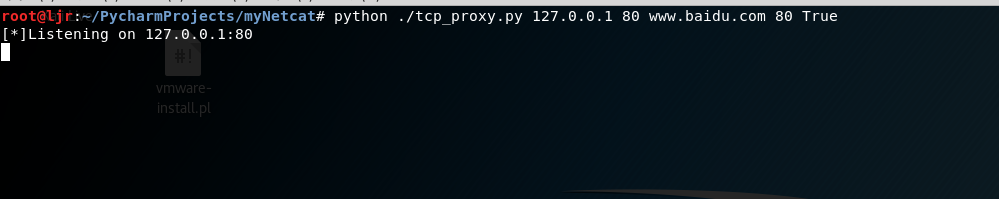
**我们还可以直接利用客户端发送http请求:**



**tcp\_proxy运行测试：**

**在命令行下的运行方法如下:python ./文件名.py [localhost] [localport] [remotehost] [remoteport] [receive\_first]**

**在这里我们采用的测试方法是先运行如下的命令：python ./tcp\_proxy 127.0.0.1 80 www.baidu.com 80 True把我们的服务器本地侦听端口设置为80 地址为127.0.0.1而远程的目的地址为DNS地址**[**www.baidu.com**](http://www.baidu.com)**，端口号为80。之后设置浏览器的代理为本地服务器的监听端口号，这也就意味着，浏览器作为了一个客户端连接到了本地的服务器所侦听的端口，之后浏览www.baidu.com发送一个HTTP的GET请求，这也就相当于浏览器作为一个客户端，将GET请求发送给本地的80端口，又我们的程序监听的是本地的80端口，就会将GET请求发送给远程主机（即我们开启程序时设置的www.baidu.com的80端口），最后再将www.baidu.com返回的数据给回程序，再给回浏览器。**





**设置完成后，我们打开浏览器浏览**[**www.baidu.com**](http://www.baidu.com)

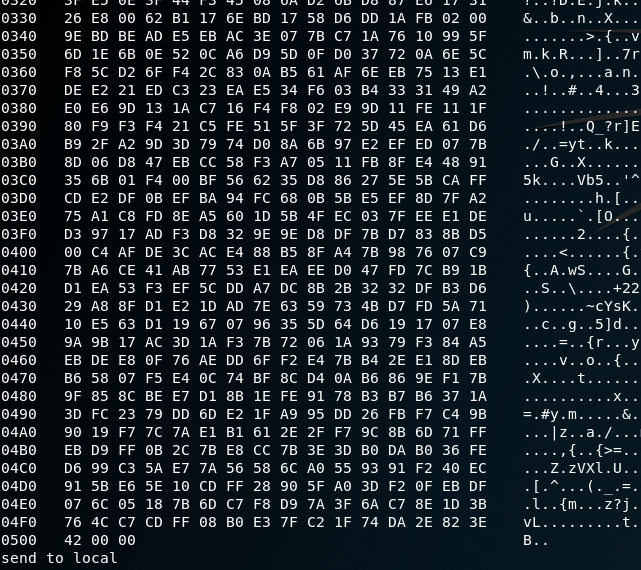


**这时可以看到在命令行中**



**出现了从本地发出的HTTP请求的数据，发送到远程的目的地址和端口。**





**接下来收到了从远程客户端发来的响应数据。**

**运行测试成功！**

**实验总结：**

**本次课程大作业中，我主要实现了一个黑客的常用工具——netcat，以及实现了一个简单的TCP代理工具，实验的环境是Kali Linux。选题的主要原因是由于我们本属于信息安全专业，所以编程的实验项目也应尽量贴近于信息安全，网络安全的相关内容，我们知道TCP代理以及netcat都是网络安全领域的常用工具，原本我们都只是去使用编写好了的网络工具，但是光是使用别人写好的工具对自己能力的提高并没有帮助。这学期学习了Linux下的网络编程，了解了基于C语言的套接字编程。但是考虑到要实现上述的两个工具，使用C语言在编程方面会较为复杂，代码量会比较大，所以我决定采用python来编写。其实除了编码上的简略，整体上的网络编程框架是没有什么区别的，都是采用套接字实现客户端与服务器端的编程。事实上，很多的黑客网络攻击工具都是采用python来编写的。在试验中，我遇到了一些对于python下命令行解析的问题，最后通过搜索相关资料，找到了getopt()以及subprocess两个函数的相关用法，从而解决了这个问题。其次，由于实验中我是采用虚拟机进行的，所以程序测试环境较为有限，只是测试完成了其中的一部分功能，更多的功能需要在特定的网络环境下进行。**