链滴

Python 制作 Netcat（2）

作者：tionch

原文链接：https://ld246.com/article/1540047416080

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

知道了UDP和TCP通信之后，我们现在正式的来设计一下Python版本的NetCat。

首先指定编码以及引入必要的模块,全局变量。

```
#!/usr/bin/python
#-*- coding:utf8 -*-

import sys
import socket
import getopt
import threading
import subprocess

listen              = False
command             = False
upload              = False
execute             = ""
target              = ""
upload_destination  = ""
port                = 0
```

  由于博主很少用Python，所以当时看书的时候没看懂getopt和subprocess这两个模块，相信很多Python初学者应该也对这两个模块比较的陌生。

getopt

大部分学习Python语言的人应该都学过C语言，其中C语言中主函数的写法是这样的

```
#include <stdio.h>

int main(int arg,char *argv[])
{
    printf("%d\n",arg);
    for(int i=0;i<arg;i++)
    {
        printf("%s\n",argv[i]);
    }
    return 0;
}
```

博主使用的是arch linux，使用gcc编译后输入

```
gcc a.c -o a
./a 1 2 3
```

得到的输出为

```
4
./a
1
2
3
```

  所以可以发现，第一个变量arg就是数组argv的长度，而argv数组中argv[0]

你运行时输入的那个东西（我也不知道怎么解释，你可以理解为文件名）,其余的几个就是你输入的参。

而在Python中也是差不多的意思。

getopt模块中有两个函数，

```
getopt.getopt
getopt.gnu_getopt
```

属性：

```
getopt.error
getopt.GetoptError
```

这两个属性主要是用来抛出错误信息的，非常友好不是吗？

```
getopt.getopt(args, shortopts, longopts=[])
```

args指的是当前脚本接收的参数，它是一个列表，可以通过sys.argv获得

```
shortopts 是短参数      啥是短参数啊?       类似于   这样：python test.py -h # 输出帮助信息
longopts 是长参数       啥是长参数啊?       类似于   这样：python test.py -help # 输出帮助信息
```

试着运行这个Python小例子，

```
import sys
import getopt
arg = getopt.getopt(sys.argv[0:],'-h',['help'])
print(arg)
```

```
Python test.py 1 2
```

再试试吧argv中的0改成1看看。

可以发现，返回值是一个list

假设我们需要匹配例如python test.py -h一类的应该怎么办呢？

试试下面这个小例子

```
import getopt
import sys

opts,args = getopt.getopt(sys.argv[1:],'-h-f:-v',['help','filename=','version'])
for opt_name,opt_value in opts:
    if opt_name in ('-h','--help'):
        print("[*] Help info")
        exit()
    if opt_name in ('-v','--version'):
        print("[*] Version is 0.01 ")
        exit()
    if opt_name in ('-f','--filename'):
        fileName = opt_value
        print("[*] Filename is ",fileName)
        # do something
        exit()
```

试试

```
python test.py -h
python test.py -f test
```

相信你做到现在，你已经明白了如何去与命令交互了。

来详细解释一下这几行代码

首先从短参数名开始。

我定义了'-h-f:-v' 大家发现没有，在-f后面多了一个":"

这个":"代表了当前参数是有值的，是一个参数名+参数值的参数

如果我再加一个-o: 那么证明-o后面是可以接收一个值，这个值就是-o的参数值，将会保存到opts变中。

长参数名的方式和短参数差不多，唯一的区别就是长参数如果要接收值，那必须得在后面加上一个"="

subprocess模块稍微一搜，发现是一个对子进程管理的一个模块，秒懂。

与命令行的交互设计

首先设计一个函数，作为一个用户帮助函数。

```
def usage():
    print "NetCat by TionchTy"
    print
    print
    print "Usage: NetCat.py -t target_host -p port"
    print
    print "-l --listen                    - listen on [host]:[port] for incoming connections"
    print "-e --execute=file_to_run       - execute the govem file upon receiving a connection"
    print "-c --command                   - initialize a command shell"
    print "-u --upload=destination        - upon receiving connection upload a file and write to
destination]"
    print
    print
    print "Exeamples:"
    print "NetCat.py -t 192.168.0.1 -p 5555 -l -c"
    print "NetCat.py -t 192.168.0.1 -p 5555 -l -u=C:\\target.exe"
    print "NetCat.py -t 192.168.0.1 -p 5555 -l -e=\"cat /etc/passwd\""
    print "echo \"somethings\" | ./NetCat.py -t 192.168.11.12 -p 135"
```

很简单，只不过是一些输出罢了

设计一个主函数与命令行交互

```
def main():
    global listen
    global port
    global execute
    global command
    global upload_destination
    global target

    if not len(sys.argv[1:]):
```

```
        usage()

    try:
        opts,args = getopt.getopt(sys.argv[1:],"hle:t:p:cu:",["help","listen","execute","target","port
,"command","upload"])
    except getopt.GetoptError as err:
        print str(err)
        usage()

    for o,a in opts:
        if o in ("-h","--help"):
            usage()
        elif o in ("-l","--listen"):
            listen = True
        elif o in ("-e","--execute"):
            execute = a
        elif o in ("-c","--commandshell"):
            command = True
        elif o in ("-u","--upload"):
            upload_destination = a
        elif o in ("-t","--target"):
            target = a
        elif o in ("-p","--port"):
            port = int(a)
        else:
            assert False,"[!]Unhandled Option"

    if not listen and len(target) and port > 0:
        buffer = sys.stdin.read()
        client_sender(buffer)
    if listen:
        server_loop()
```

其中这里面server_loop,client_sender函数是自己写的，具体的我会在下一篇博客中详细的解释。

#Refrence

简书Python命令行:getopt模块详解