链滴

Python 制作 Netcat（3）

作者：tionch

原文链接：https://ld246.com/article/1540047738219

来源网站：链滴

数据传输其实只是一个很简单的tcp传输

1.创建数据发送函数

```
def client_sender(buffer):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        # 连接到目标主机
        client.connect((target, port))

        if len(buffer):
            client.send(buffer)

        while True:
            # 现在等待数据回传
            recv_len = 1
            response = ""
            while recv_len:
                data = client.recv(4096)
                recv_len = len(data)
                response += data
                if recv_len < 4096:
                    break
            print  response
            # 等待更多的输入
            buffer = raw_input("")
            buffer += "\n"
            # 发送出去
            client.send(buffer)
    except:
        print "[*] Exception! Exiting."
    #关闭连接
    client.close()
```

其中数据回传中，创建一个4096字节的容器，通过数据与4096的大小关系，判定数据是否传输完成。

2.创建服务器端主循环

```
def server_loop():
    global target

    # 如果没有定义目标,那我们监听所有接口
    if not len(target):
        target = "0.0.0.0"

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((target, port))

    server.listen(5)

    while True:
        client_socket, addr = server.accept()
        # 分拆一个线程处理新的客户端
```

```
        client_thread = threading.Thread(target=client_handler, args=(client_socket,))
        client_thread.start()
```

这些代码还是相对来说很简单，只是一个简单的tcp服务端的写法

3.创建运行命令的函数

```
def run_command(command):
    # 删除字符串末尾的空格
    command = command.rstrip()
    # 运行命令并将输出放回
    try:
        output = subprocess.check_output(command, stderr=subprocess.STDOUT, shell=True)
    except:
        output = "Failed to execute command.\r\n"
    # 将输出发送
    return output
```

  这里解释一下subprocess.check_output()函数，父进程等待子进程完成返
子进程向标准输出的输出结果检查退出信息，如果returncode不为0，则举出错误subprocess.Called
rocessError，该对象包含有returncode属性和output属性，output属性为标准输出的输出结果，可
try...except...来检查。

总的来说就是检查命令是否是正确或者说正确运行，如果不是，就会把错误的信息传回给你。

调用系统中shell命令,显示命令执行的结果:

```
x=subprocess.check_output(["echo", "Hello World!"],shell=True)
```

4.客户端运行

```
def client_handler(client_socket):
    global upload
    global execute
    global command

    # 检查上传文件
    if len(upload_destination):
        # 读取所有的字符并写下目标
        file_buffer = ""
        # 持续读取数据直到没有符合的数据
        while True:
            data = client_socket.recv(1024)

            if not data:
                break
            else:
                file_buffer += data

        try:
            file_descriptor = open(upload_destination, "wb")
            file_descriptor.write(file_buffer)
            file_descriptor.close()
```

```
            client_socket.send("Successfully saved file to %s\r\n" % upload_destination)
        except:
            client_socket.send("Failed to save file to %s\r\n" % upload_destination)

    # 检查命令执行
    if len(execute):
        # 运行命令
        output = run_command(execute)
        client_socket.send(output)
```