链滴

Python 制作简易 TCP 代理

&emsp;&emsp;如果需要一个命令行shell,那么我们进入另一个循环代理在实际的渗透中作不言而喻，因此时刻拥有一个代理工具总是好的。例如A,B,C三台设备，A可以访问B，B可以访问C，是A不能访问C，这个时候A就可以通过代理的方式间接的进行访问C。

这里来一个最简单的Tcp代理。代理端口是没有什么问题，代理80端口会有一些问题

1.引入相应的模块

```
import socket
import threading
import sys
```

2.创建一个流量转发的函数

```
def receive_from(connection):
    buffer = ""
    #设置一个两秒钟的延时
    connection.settimeout(2)
    try:
        while True:
            data = connection.recv(4096)
        if not data:
            break
        buffer += data
    except:
        pass
    return buffer
```

其实根据之前的文章，我们可以对这个函数进行一个改写

```
def receive_from(connection):
    buffer = ""
    connection.settimeout(2)
    try:
        data_len = 1
        while data_len:
            data = connection.recv(4096)
            data_len = len(data)
            buffer += data
            if data_len<4096:
                break
    except:
        pass
return  buffer
```

##2.数据包修改函数

```
def request_handler(buffer):
    return buffer

def response_handler(buffer):
    return buffer
```

4.代理线程

```
def proxy_handler(client_socket, remote_host, remote_port, receive_first):

    # 连接远程主机
```

```
remote_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
remote_socket.connect((remote_host, remote_port))

# 如果必要从远程主机接收数据
if receive_first:

    remote_buffer = receive_from(remote_socket)
    hexdump(remote_buffer)

    # 发送给我们的响应处理
    remote_buffer = response_handler(remote_buffer)

    # 如果我们有数据传递给本地客户端,发送它
    if len(remote_buffer):
        print "[<==] Sending %d bytes to localhost." % len(remote_buffer)
        client_socket.send(remote_buffer)

# 现在我们从本地循环读取数据,发送给远程主机和本地主机
while True:

    # 从本地读取数据
    local_buffer = receive_from(client_socket)

    if len(local_buffer):
        print  "[==>] Received %d bytes from localhost" % len(local_buffer)
        hexdump(local_buffer)

        # 发送给我们的本地请求
        local_buffer = request_handler(local_buffer)

        # 向远程主机发送数据
        remote_socket.send(local_buffer)
        print  "[==>] Sent to remote."

    # 接受响应的数据
    remote_buffer = receive_from(remote_socket)

    if len(remote_buffer):

        print "[<==] Received %d bytes from remote." % len(remote_buffer)
        hexdump(remote_buffer)

        #发送到响应处理函数
        remote_buffer = response_handler(remote_buffer)

        #将响应发送给本地socket
        client_socket.send(remote_buffer)

        print "[<==] Sent to localhost."

    #如果两边都没有数据,关闭连接
    if not len(local_buffer) or not len(remote_buffer):
        client_socket.close()
        remote_socket.close()
```

```
            print "[*] No more data. Closing connections."

            break
```

这个函数总的来说就是将本地的请求通过数据包修改函数之后发给服务器，服务器也把数据传回

```
def server_loop(local_host, local_port, remote_host, remote_port, receive_first):

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        server.bind((local_host, local_port))
    except:
        print  "[!!] Failed to listen on %s:%d" % (local_host, local_port)
        print  "[!!] Check for other listening sockets or correct permissions."
        sys.exit(0)

    print "[*] Listening on %s:%d" % (local_host, local_port)

    server.listen(5)

    while True:
        client_socket, addr = server.accept()

        #打印出本地连接信息
        print "[==>] Received incoming connectiong from %s:%d" %  (addr[0], addr[1])
        #开启一个线程与远程主机通信
        proxy_thread = threading.Thread(target=proxy_handler, args=(client_socket, remote_hos
, remote_port, receive_first))
        proxy_thread.start()
```

3.主循环，在上次已经说过

```
def main():
    # 没有华丽的命令行解析
    if len(sys.argv[1:]) != 5:
        print "Usage: ./tcpProxy.py [localhost] [localport] [remotehost] [remoteport] [receive_first
"
        print "Example: ./tcpProxy.py 127.0.0.1 9000 10.12.132.1 9000 True"
        sys.exit(0)

    # 设置本地监听参数
    local_host = sys.argv[1]
    local_port = int(sys.argv[2])

    #设置远程目标
    remote_host = sys.argv[3]
    remote_port = int(sys.argv[4])

    #告诉代理在发送给远程主机之前连接和接受数据
    receive_first = sys.argv[5]

    if "True" in receive_first:
        receive_first = True
```

```
else:
    receive_first = False

#现在设置好我们的监听socket
server_loop(local_host, local_port, remote_host, remote_port, receive_first)


if command:
    while True:
        # 跳出一个窗口
        client_socket.send("<BHP:#>")

        cmd_buffer = ""
        while "\n" not in cmd_buffer:
            cmd_buffer += client_socket.recv(1024)
        #  返回命令输出
        response = run_command(cmd_buffer)
        # 返回响应数据
        client_socket.send(response)
```