黑客派

Python 制作 Netcat（1）

作者：tionch

原文链接：https://hacpai.com/article/1540046966427

来源网站：黑客派

netcat 作为一款极为优秀的后门工具，在网络工具中有“士军刀”美誉。
 如何使用 Python 进行开发一个类似的呢？
 现在先来学习一下 TCP 和 UP 协议以及 Python 的实现

<h2 id="TCP协议-来自百度百科-">TCP 协议（来自百度百科）</h2>
<blockquote>
TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节的传输层通信协议，由 IETF 的 RFC 793 定义。在简化的计算机网络 OSI 模型中，它完成第四层传输所指定的功能，用户数据报协议（UDP）是同一层内[1] 另一个重要的传输协议。在因特网协议族（Internet protocol suite）中，TCP 层是位于 IP 层之上，应用层之下的中间层。不同主机的应用层之间常需要可靠的、像管道一样的连接，但是 IP 层不提供这样的流机制，而是提供不可靠的包交换。[1]
 应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，然后 TCP 把数据流分区成适长度的报文段（通常受该计算机连接的网络的数据链路层的最大传输单元（[1] MTU）的限制）。之后 TCP 把结果包传给 IP 层，由它来通过网络将包传送给接收端实体[1] 的 TCP 层。TCP 为了保证不发丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体已成功收到的包发回一个相应的确认（ACK）；如果发送端实体在合理的往返时延（RTT）内未收到认，那么对应的数据包就被假设为已丢失将会被进行重传。TCP 用一个校验和函数来检验数据是否有误；在发送和接收时都要计算校验和。
</blockquote>
<h2 id="UDP协议-来自百度百科-">UDP 协议（来自百度百科）</h2>
<blockquote>
UDP 是 User Datagram Protocol 的简称， 中文名是用户数据报协议，是 OSI（Open System nterconnection，开放式系统互联） 参考模型中一种无连接的传输层协议，提供面向事务的简单不靠信息传送服务，IETF RFC 768 是 UDP 的正式规范。UDP 在 IP 报文的协议号是 17。
 UDP 议全称是用户数据报协议[1] ，在网络中它与 TCP 协议一样用于处理数据包，是一种无连接的协议。在 OSI 模型中，在第四层——传输层，处于 IP 协议的上一层。UDP 有不提供数据包分组、组装和不能数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP 用支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器式的网络应用都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩经被一些类似协议所掩盖，但是即使是在今天 UDP 仍然不失为一项非常实用和可行的网络传输层协。
 与所熟知的 TCP（传输控制协议）协议一样，UDP 协议直接位于 IP（网际协议）协议的顶。根据 OSI（开放系统互连）参考模型，UDP 和 TCP 都属于传输层协议。UDP 协议的主要作用是将络数据流量压缩成数据包的形式。一个典型的数据包就是一个二进制数据的传输单位。每一个数据包前 8 个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。
</blockquote>
<h2 id="TcpServer设计">TcpServer 设计</h2>
<pre><code class="language-python highlight-chroma">#!/usr bin/python
#-*- coding:utf8 -*-

import socket
import threading

bind_ip = "0.0.0.0"

```
bind_port = 8021

#指定为ipv4的socket以及网络流传输
server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server.bind((bind_ip,bind_port))
server.listen(5)
print "[*] Listening on %s:%d" %(bind_ip,bind_port)

#客户处理线程函数
def handler_client(client_socket):
request = client_socket.recv(2048)
print "[*] Received %s" % request
client_socket.send("Hello Client")
client_socket.close()

while True:
#接受连接
client,addr = server.accept()
```

print "[*] Accepted Connecti
n from: %s :
/span> %d "
/span> % (<span cla
s="highlight-n">addr [<span class="hig
light-mi">0],
ddr [1
/span>])

#创建新线程

client_handler = <spn class="highlight-n">threading .<span
lass="highlight-n">Thread (<span class
"highlight-n">target =<span class="hig
light-n">handler_client ,<span class="hi
hlight-n">args =<span class="highlight
p">(client
))

client_handler .</spa
> start ()

</code></pre>

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>

<!-- 黑客派PC帖子内嵌-展示 -->

<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h2 id="TcpClient设计">TcpClient 设计</h2>
<pre><code class="language-python highlight-chroma">impor
 socket

target_ip = <span cl
ss="highlight-s2">"192.168.1.100"

target_port = <span
lass="highlight-mi">8021

client = <span class
"highlight-n">socket .<span class="high
ight-n">socket (<span class="highlight
n">socket .
F_INET ,
ocket .
OCK_STREAM)

client .<span
class="highlight-n">connect ((<span cla
s="highlight-n">target_ip ,<span class=

highlight-n">target_port))

client .send ("Hello Server")

response = client .recv (2048)

print response

</code></pre>

运行效果
相比较 TCP 协议，UDP 也类似的写法。
<pre><code class="language-python highlight-chroma">UDPSever设计
import socket
import threading

s = socket .socket (socket .AF_INET , socket .SOCK_DGRAM)

s .bind (('127.0.0.1' , 9999))

print ('[]Bind UDP on 9999...')

while True:

data , addr = s .recvfrom(1024)

print ('[]Received from %s:%s.' % addr)

print data

s .sendto ('Hello, %s!' % data

pan>   , addr<sp
n class="highlight-p">)

print addr

</code></pre>

UDPClient 设计
 import socket
target_ip = "127.0.0.1"
 target_port = 9999
 client = socket.socket(socket.AF_INE
,socket.SOCK_DGRAM)
 client.sendto("Hello Server",(target_ip,target_port))
 data,add
 = client.recvfrom(2048)
UDP 的使用与 TCP 类似，但是不需要建立连接。此外，服务器绑定 UDP 端口和 TCP 端口互不
突，也就是说，UDP 的 9999 端口与 TCP 的 9999 端口可以各自绑定。