

Charles安装

Charles客户端下载：

官网地址：<https://www.charlesproxy.com/download/>

选择适合自己的系统版本下载

Download Charles

The latest version of Charles is 4.6.4. [Read the Release Notes.](#)

Apple Silicon and ARM64

The latest beta release of Charles 5 supports **Apple Silicon**, Windows ARM64 and Linux ARM64. If you are using any of these platforms, please download the [latest beta version](#) instead.

v4.6.4



[Windows x86_64](#) (msi, 56.4 MB)



[macOS](#) (dmg, 54.6 MB)

Compatible with macOS 10.7 — 13.



[Linux x86_64](#) (tar.gz, 51 MB)

Or use the [APT](#) or [YUM](#) package repositories.

Mozilla Firefox add-on

Firefox no longer requires an add-on to work with Charles. Instead configure Firefox to use your system proxy settings.

下载安装完成后激活

激活网站地址：<https://www.zzzmode.com/mytools/charles/>

/

RegisterName

生成

建议安装使用最新版，官方下载地址 <https://www.charlesproxy.com/download>

描述
此工具用于计算Charles激活码，[下载代码](#)，在线运行代码：<https://play.golang.org/p/Qt2CmHbTzU>
blog介绍: <https://blog.zzzmode.com/2017/05/16/charles-4.0.2-cracked>

用法
输入RegisterName(此名称随意，用于显示 Registered to xxx)，点击生成计算出注册码，打开Charles输入注册码即可。


注意
仅供个人学习研究和交流使用，请勿用于任何商业用途。

广告
如果您有使用国外VPS的需求，不妨试试Vultr，我的推广链接 <https://www.vultr.com/?ref=6855736>

High
Performance

SSD
Storage

15
Locations

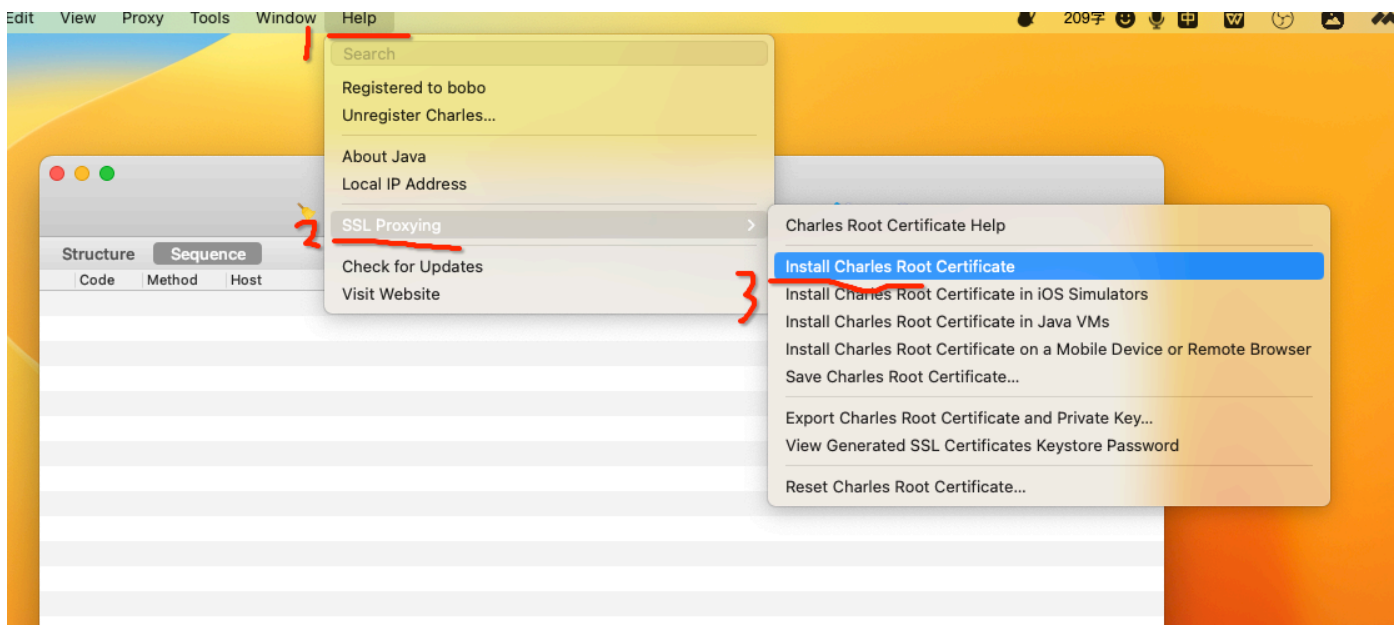
 Starting from
\$2.50/mo
\$0.004/hr

Get Started

输入任意注册
名后，点击生
成按钮，将注
册名和生成的
内容作用在
Charles的激
活环节中

打开安装好的Charles，菜单栏 Help->Register Charles 弹出注册的窗口填入Registered Name和生成的license key，点击Register。

Charles配置，打开Charles，先安装证书并且信任



证书下载好之后，找到证书文件，双击进行安装，将其安装到【受信任】目录中，出现信任选项，点击始终信任即可。

常规

详细信息

证书路径



证书信息

这个证书的目的如下:

- 所有颁发策略
- 所有应用程序策略

颁发给: Charles Proxy CA (6 五月 2019, TBN-0MNN5S)

颁发者: Charles Proxy CA (6 五月 2019, TBN-0MNN5S)

有效期从 2000/1/1 到 2048/7/2

安装证书(I)...

颁发者说明(S)

确定



欢迎使用证书导入向导

该向导可帮助你将证书、证书信任列表和证书吊销列表从磁盘复制到证书存储。

由证书颁发机构颁发的证书是对你身份的确认，它包含用来保护数据或建立安全网络连接的信息。证书存储是保存证书的系统区域。

存储位置



- ☒ 当前用户(C)
- ☐ 本地计算机(L)

单击“下一步”继续。

下一步(N)

取消



  证书导入向导

证书存储

证书存储是保存证书的系统区域。

Windows 可以自动选择证书存储，你也可以为证书指定一个位置。

☒ 根据证书类型，自动选择证书存储(U)

☐ 将所有的证书都放入下列存储(P)

证书存储:

浏览(R)...

选择证书存储



选择要使用的证书存储(C)。

- 个人
- 受信任的根证书颁发机构
- 企业信任
- 中间证书颁发机构
- 受信任的发布者
- 不信任的证书
- 第三方根证书颁发机构

☐ 显示物理存储区(S)

确定

取消

浏览(R)...

书指定一个位置。

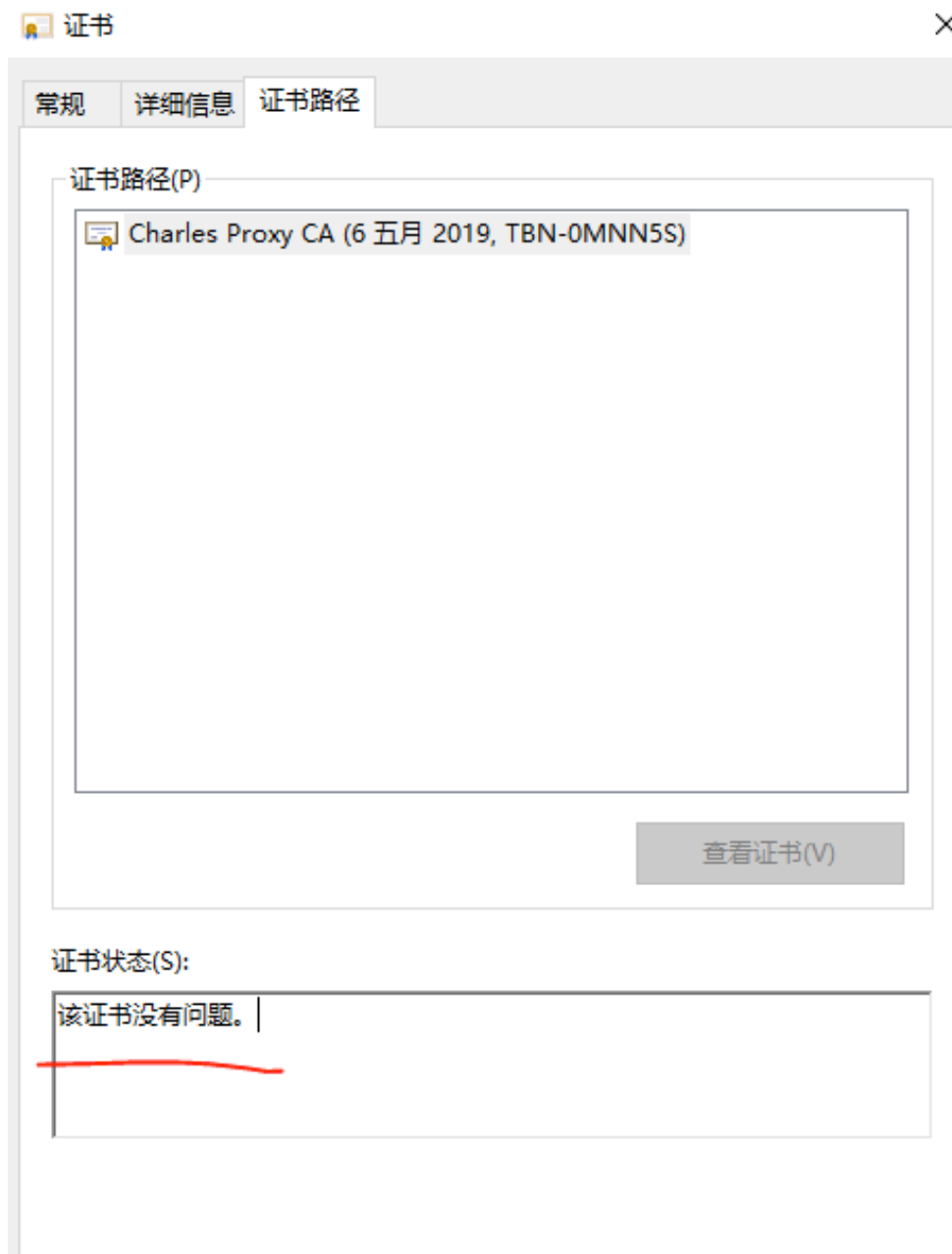
正在完成证书导入向导

单击“完成”后将导入证书。

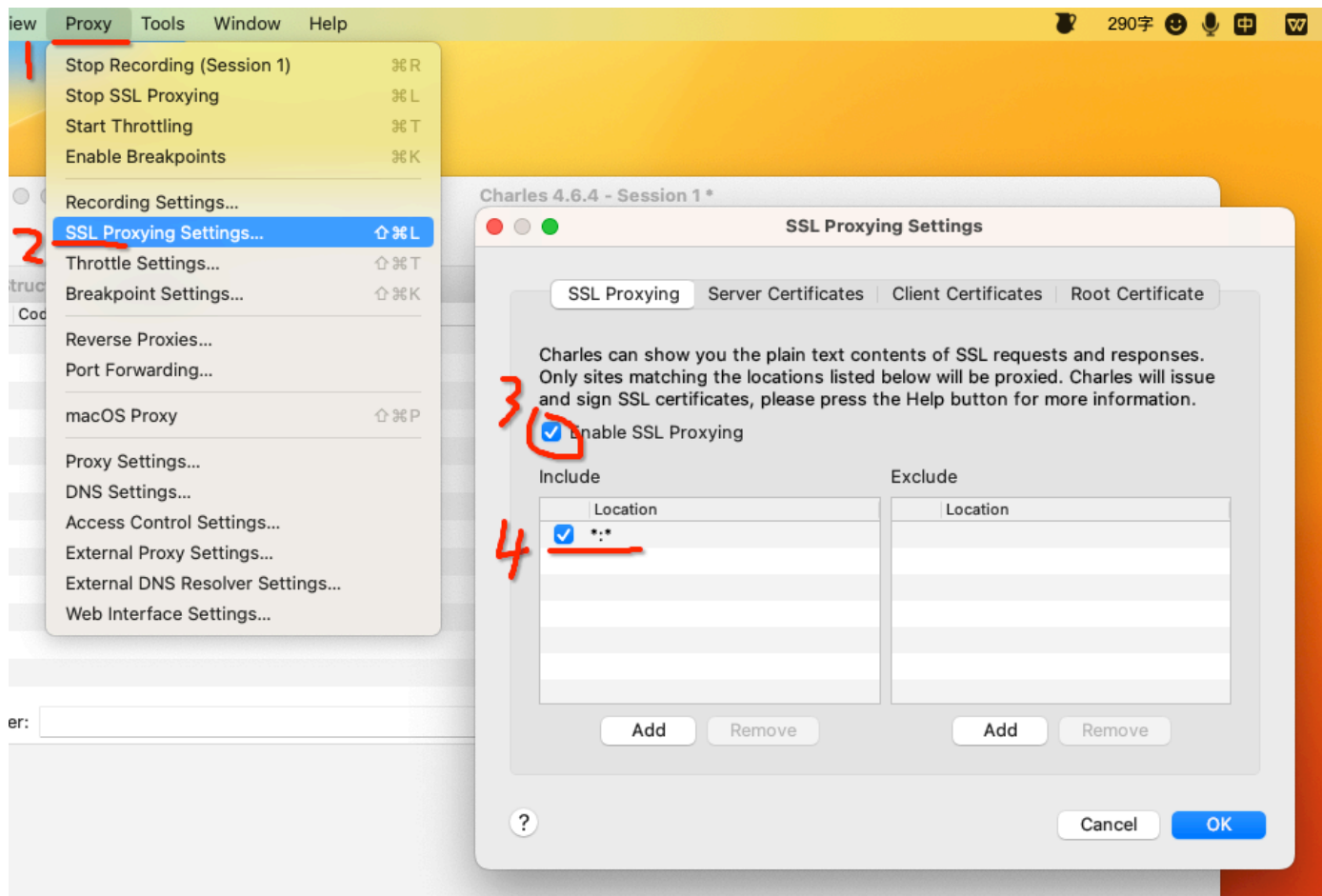
你已指定下列设置:

用户选定的证书存储	受信任的根证书颁发机构
内容	证书

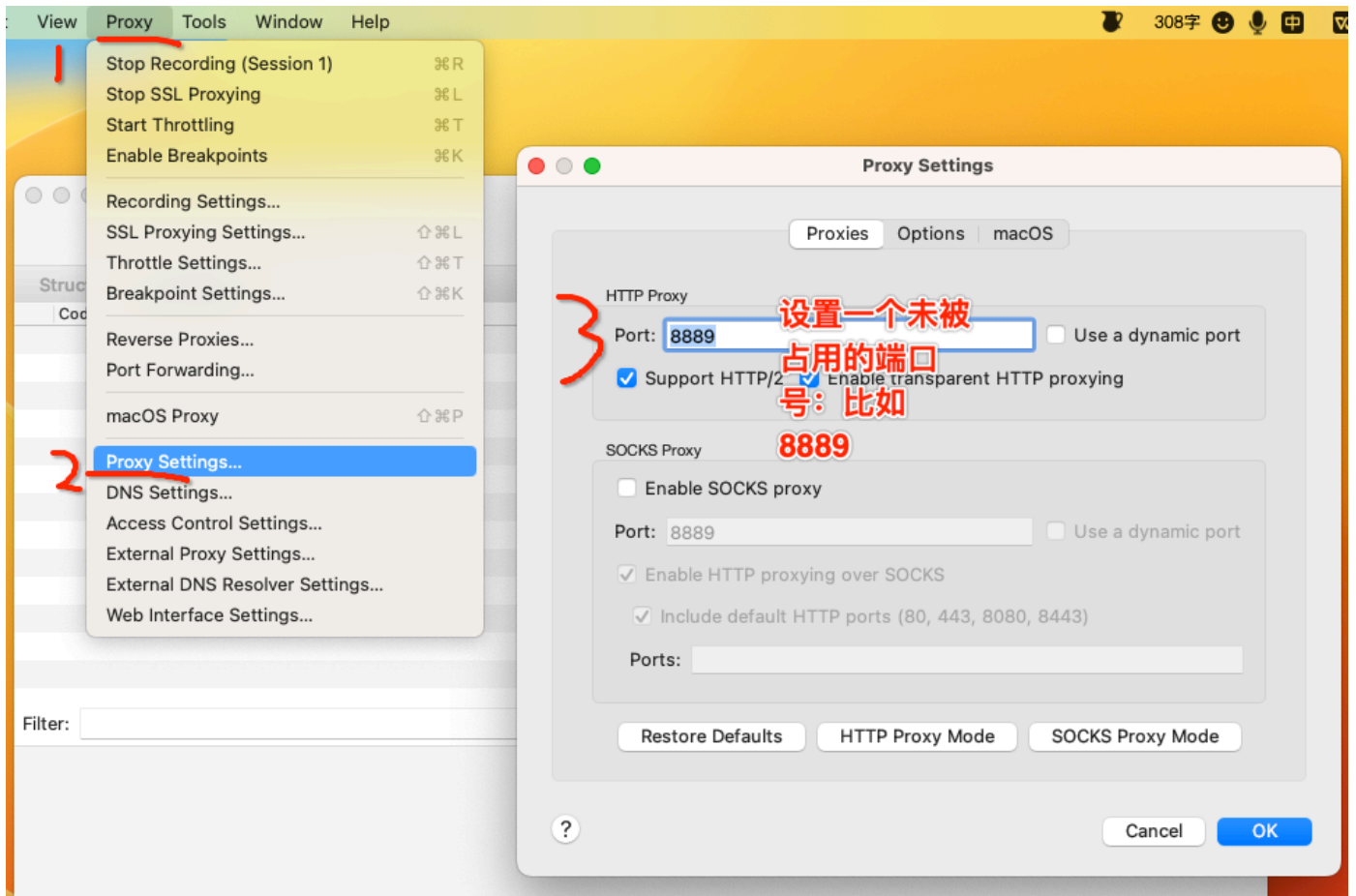
点击完成后提示导入成功。此时需要重新进入Help -> SSL Proxying -> Install Charles Root Certificate，查看证书结果，成功时如下提示：



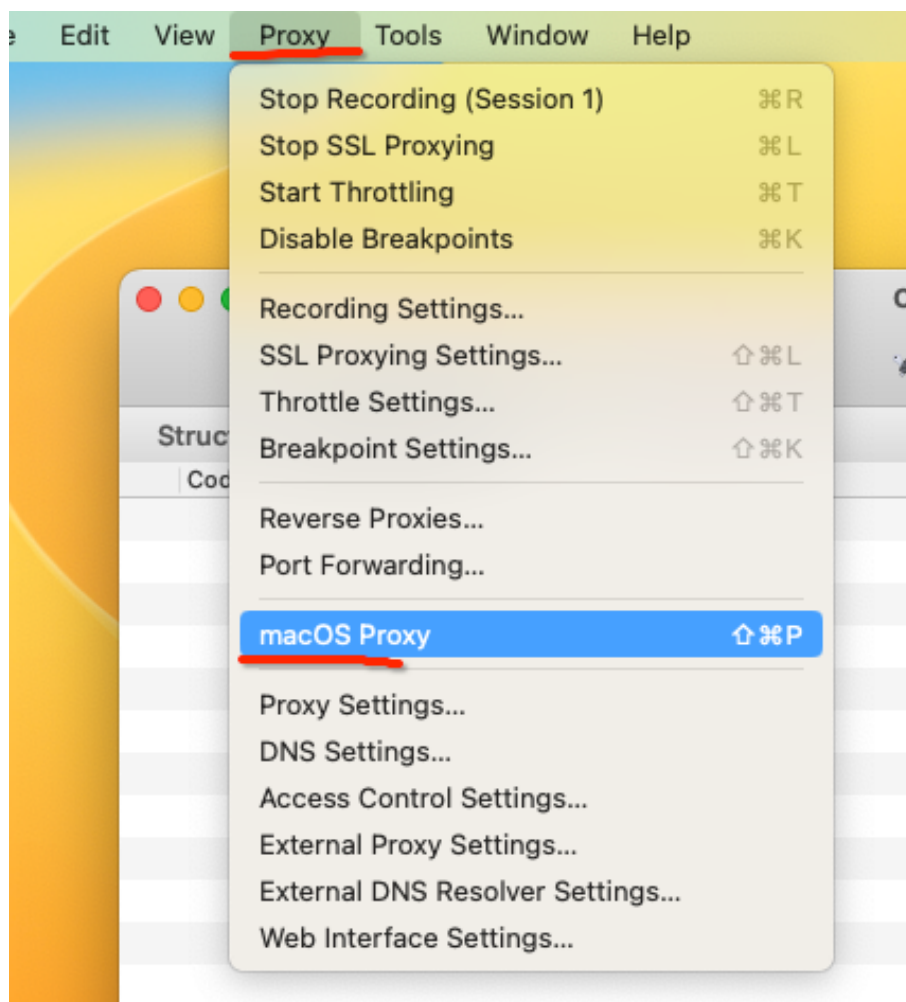
设置SSL，保证可以抓取https协议的请求：



这是端口号：

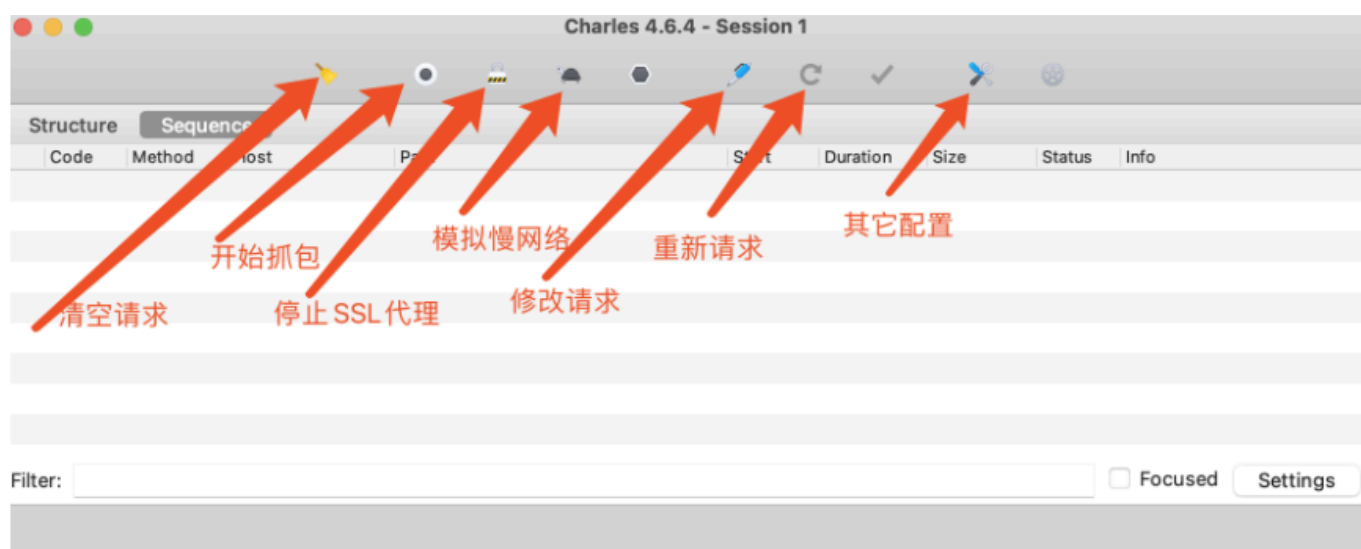


如果是mac操作系统需要如下操作，其他系统忽略该操作：



配置完成，可以使用Charles进行抓包了

工具栏功能介绍

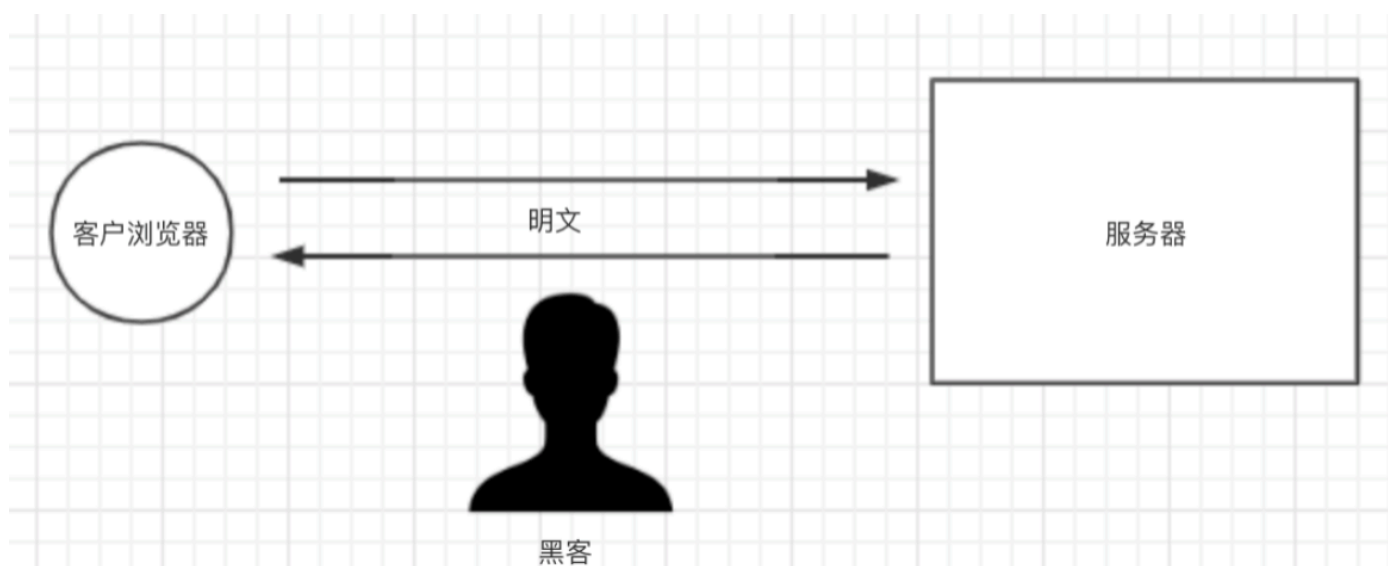


什么是证书？为何需要证书？

首先明确一点，安装证书的目的是为了是的抓包工具可以抓取https协议的请求。

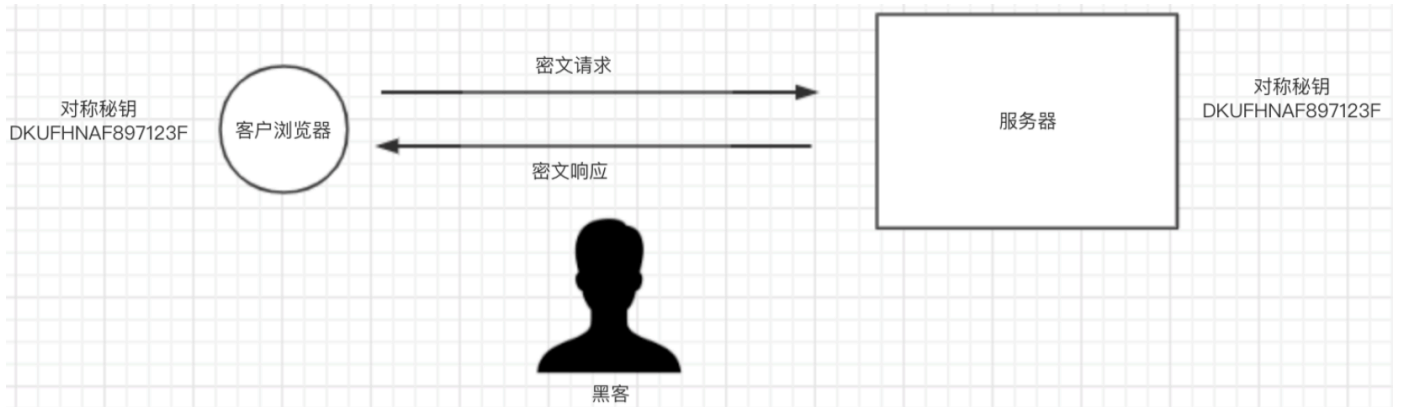
http协议是不安全的

在https诞生之前，所有网站都使用http协议，而http协议在数据传输的过程中都是明文，所以可能存在数据泄露和篡改。



使用对称密钥进行数据加密

为了防止数据泄露和篡改，我们对数据进行加密，如：生成一个对称密码【DKUFHNAF897123F】，将对称密钥分别交给浏览器和服务端，他们之间传输的数据都使用对称密钥进行加密和解密。



请求和响应流程如下：

1. 客户端使用对称密钥对请求进行加密，并发送给服务端。
2. 服务端接收到密文之后，使用对称密钥对密文进行解密，然后处理请求。最后再使用对称密钥把要返回的内容再次加密，返回给客户端。
3. 客户端接收到密文之后，使用对称密钥进行解密，并获取最终的响应内容。

如此一来，数据传输都是密文，解决了明文传输数据的问题。
但是，这么干有bug。

- 浏览器如何获取对称密钥？
- 如果每个客户端的对称密钥相同，浏览器能拿到对称密钥，那么黑客也可以拿到，所以，数据加密也就没有意义了。

非对称密钥加密

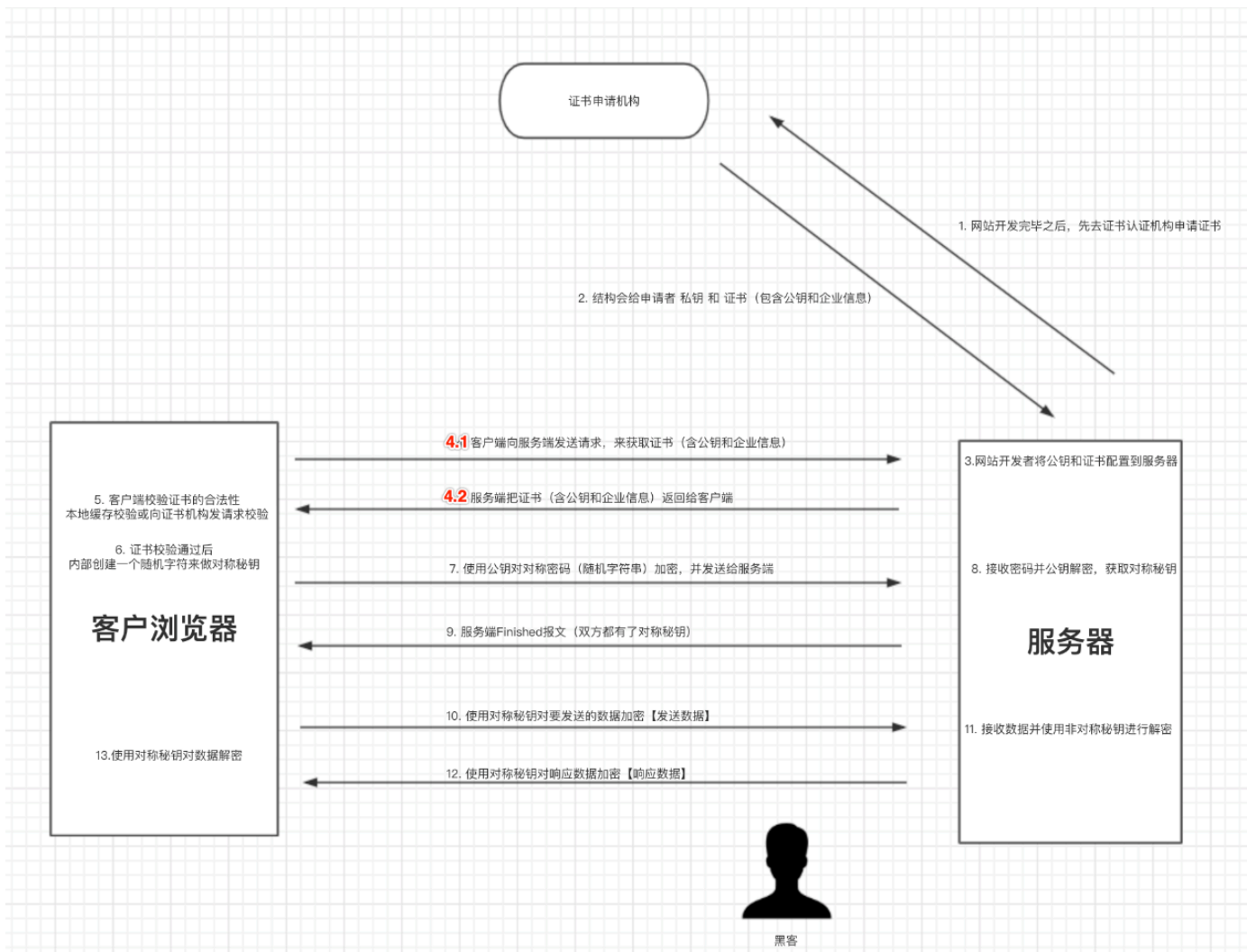
公钥私钥对儿：公钥负责加密，私钥负责解密



如此一来，解决了 动态对称密钥 和 数据加密的问题，因为每个用户的对称密钥都是随机生成且传输的过程中都使用公钥加密（公钥加密的数据只有私钥能解密），所有黑客无法截获对称密钥。而数据传输是通过对称密钥加密过的，所以黑客即使能获取数据也无法去解密看到真实的内容。看似无懈可击，**但是**，这么干还是又bug：如果黑客在上图【步骤2】劫持，黑客把自己的公钥返回给客户端，那么客户端会使用黑客的公钥来加密对称密钥，黑客在【步骤6】截获请求，使用自己的私钥获取对称密钥，后面过程全都会完蛋...

CA证书应用

使用 ca 证书可以解决黑客劫持的问题

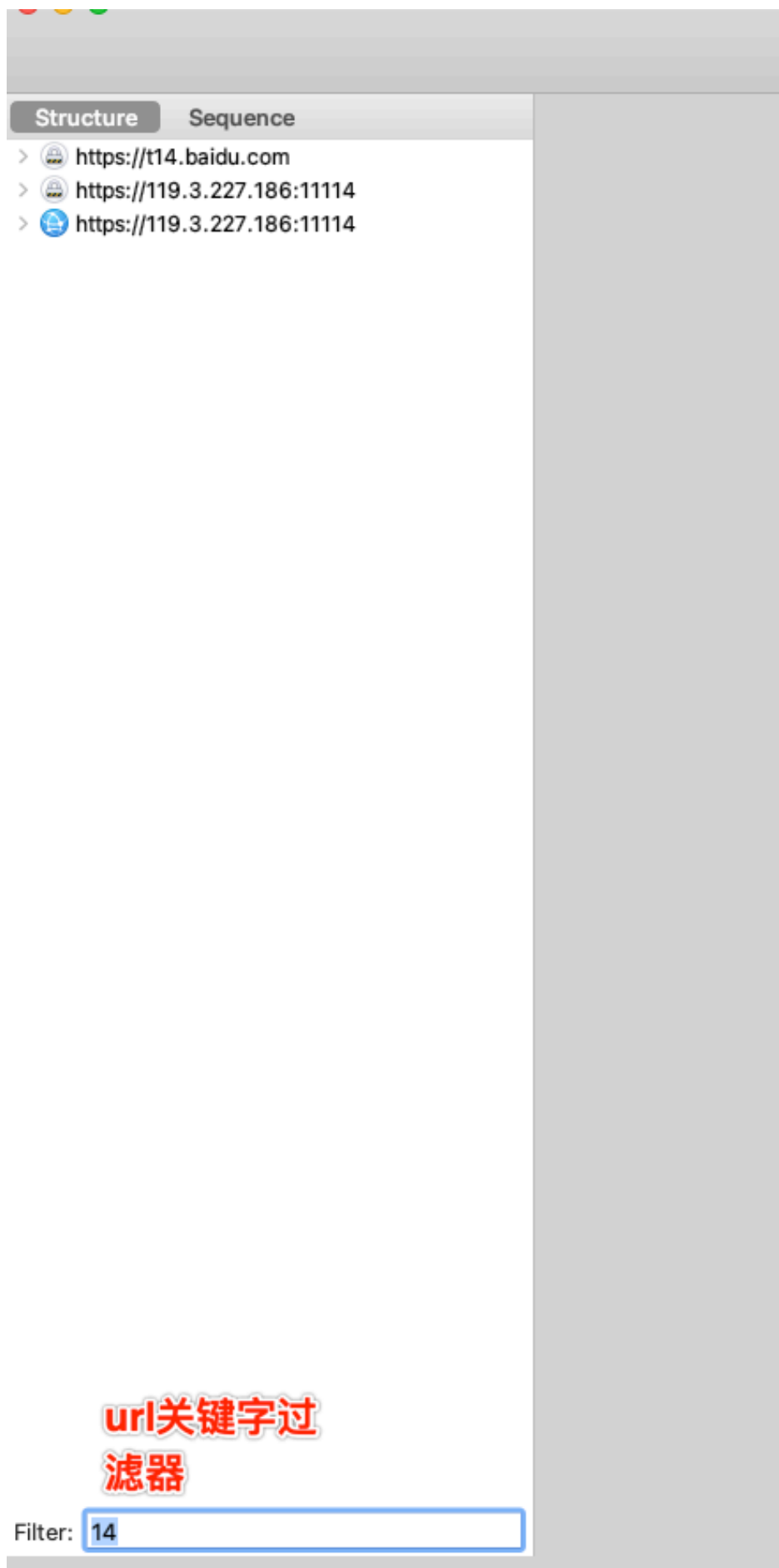


如此一来，就解决了黑客劫持的问题，因为即使黑客劫持后的给浏览器即使返回了证书也无法通过校验，同时浏览器也会提示错误信息。

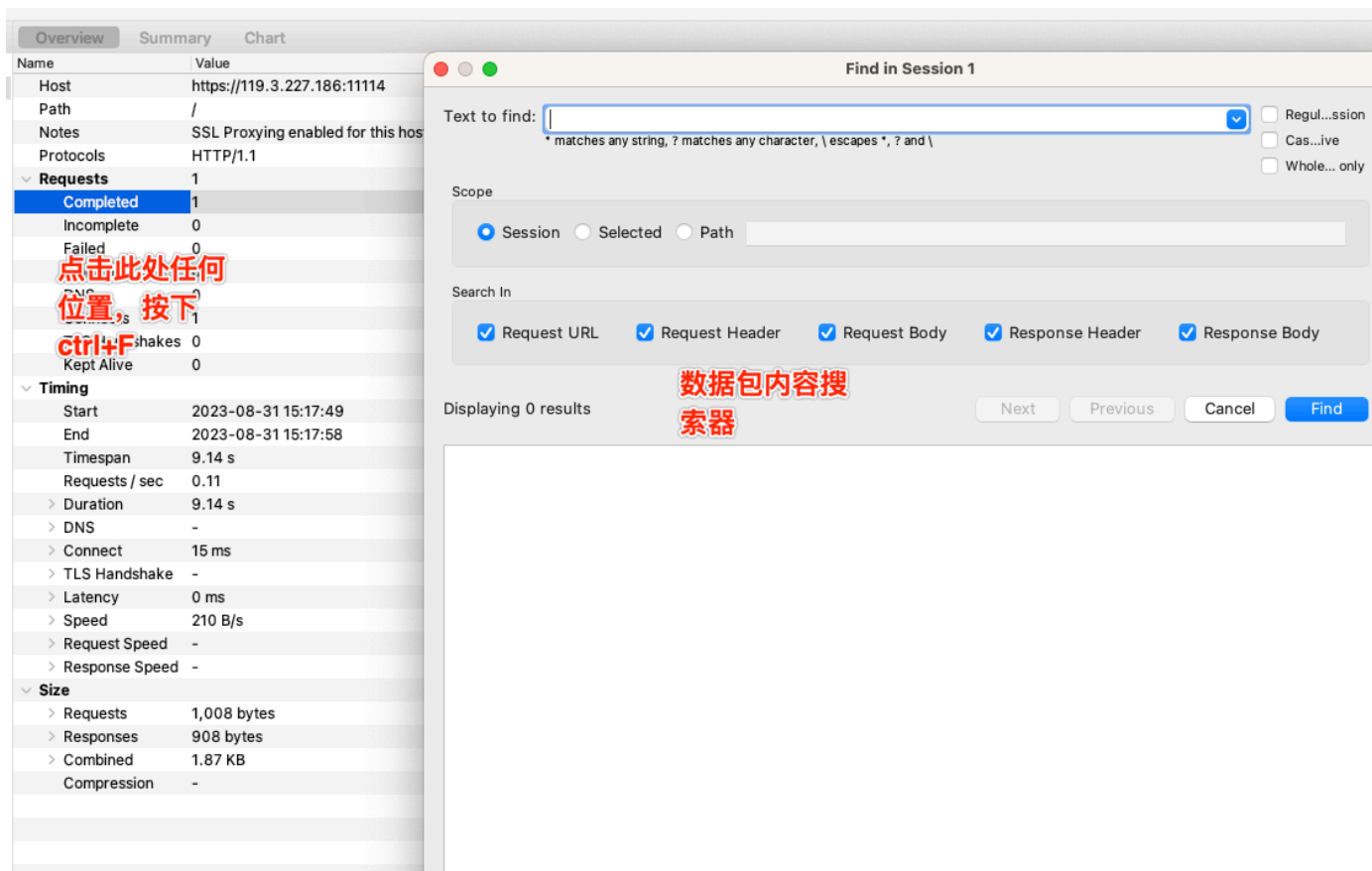
https可以保证数据安全，但由过程需要反复加密解密所有访问速度会有所下降（鱼和熊掌不能兼得）。

小技巧

过滤器：

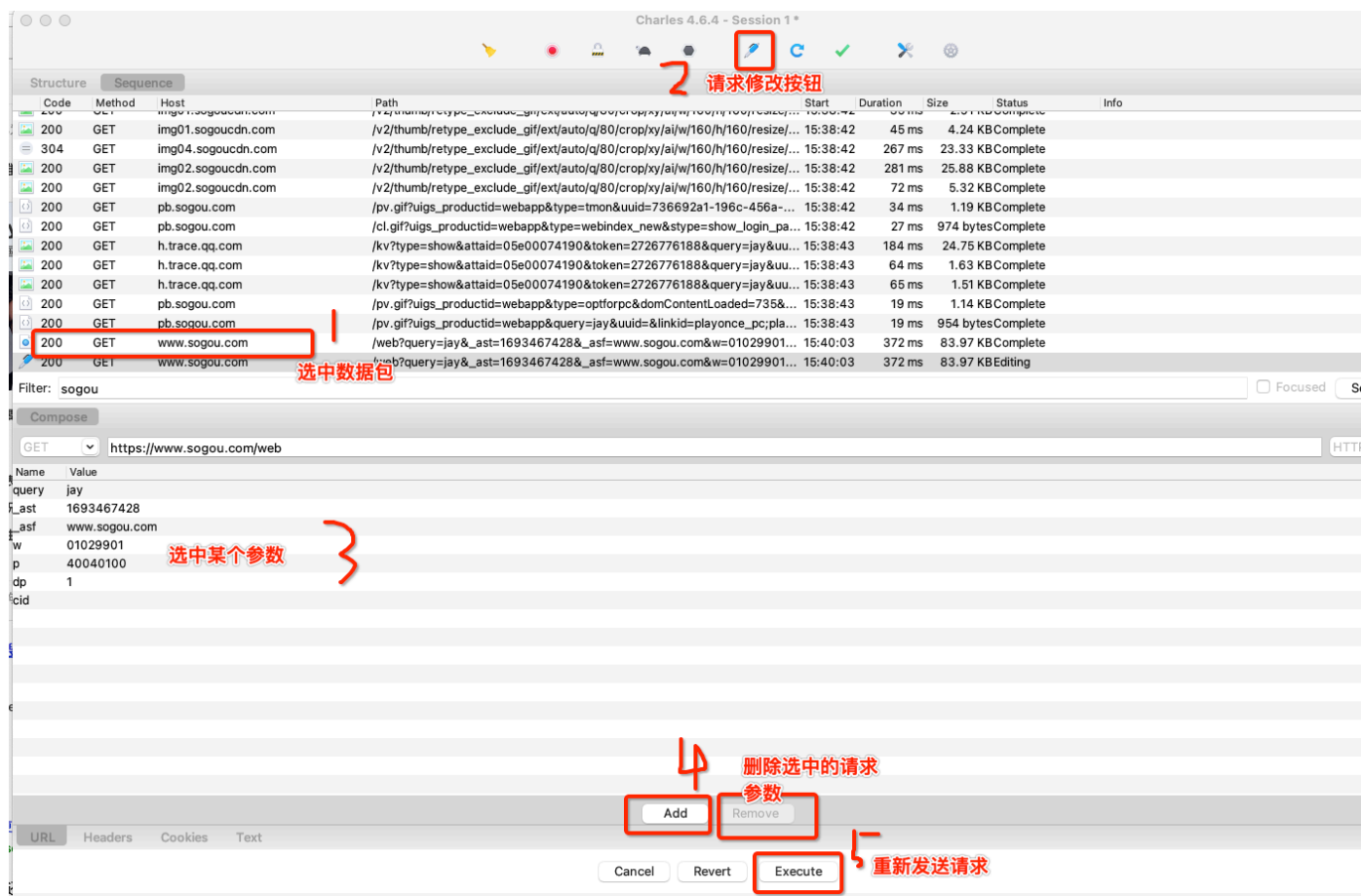


数据包内容搜索：



请求修改：用来验证哪些请求参数是必要的

请求sogou关键字搜索，然后在数据包内容搜索框中，搜索显示页面中的文字内容，定位到指定数据包，然后，选中该数据包，按下Compose键，进行请求参数修改，然后重新发送请求，查看删除参数后，响应数据是否依然正常。



案例

- 抓取微信小程序---实习僧中，python关键字对应的岗位搜索结果

#实习僧 python招聘信息抓取

```
import requests
```

#如果加了verify=False这个关键字参数，使用requests模块发送请求的时候会给你弹出一个警告

InsecureRequestWarning，警告你当前的请求可能不安全，可以使用如下代码忽略该警告

```
import urllib3
```

```
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

```
headers = {  
    'user-agent': 'Mozilla/5.0 (Macintosh; Intel  
Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,  
like Gecko) Chrome/98.0.4758.102 Safari/537.36  
MicroMessenger/6.8.0(0x16080000) NetType/WIFI  
MiniProgramEnv/Mac MacWechat/WMPF XWEB/30817',  
}
```

```
url =  
'https://apigateway.shixiseng.com/api/interns/v  
2.0/interns/wxz/search/v3?  
city=%E5%85%A8%E5%9B%BD&k=Python&intention=&deg  
ree=&internship_duration=&days_per_week=&paymen  
t_per_day=&emp_chance=&area=&scale=&category=&i  
po=&nature=&t=0&p=1&target=intern'
```

#目前各大网站基本有自己的ca证书，但是不排除有的网站为了节约网站建设开销并没有购买ca证书。又因为requests模块在发送网络请求的时候，默认会验证ca证书。如果当前网站没有ca证书，那么就会报出SSLError错误，则使用verify参数赋值False可以在请求的时候不验证网站的ca证书。

```
ret =  
requests.get(url,headers=headers,verify=False).  
json()  
print(ret)
```

- 抓取美女轩公众号里的【美女精选】

```
import requests
import re
import urllib3
import json
from lxml import etree

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

headers = {
    'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36 MicroMessenger/6.8.0(0x16080000) NetType/WIFI MiniProgramEnv/Mac MacWechat/WMPF XWEB/30817',
}
```

```
url = 'https://mp.weixin.qq.com/mp/profile_ext?
action=home&__biz=Mzg3Nzc2OTQzOA==&uin=MTM1ODMy
ODkwNQ%3D%3D&key=16ff41fc38234ef85714c38b97b06d
4054d7aaad452dcdaf5581c33d2104f5e1630ff6ee48da4
6b153f323af36a90b5e59a7129b6b28fb1c791f4a59a12e
e787ad2ca7cf19ee565396f08b773ac694d06ea340cbe0d
62d386a153ecfcdd84eaf742b071f75c5a234dc5d31204a
1b4854d1930225947bebe72cc59476490e0e22&devicety
pe=iMac+MacBookPro17%2C1+OSX+OSX+13.5+build(22G
74)&version=13080310&lang=zh_CN&nettype=WIFI&a8
scene=0&fontScale=100&acctmode=0&pass_ticket=km
2622VtVeH2MRYrt4EbChPo59EMr4MvwCDuSRJa%2F3w8GTl
BsbamxA475CNyQt3m'
```

```
page_text =
```

```
requests.get(url,headers=headers,verify=False).
text
```

```
#替换";为空串
```

```
ret = re.findall("var msgList =
'(.*)'",page_text)[0]
```

```
#将";转换成引号
```

```
import html
```

```
ret = html.unescape(ret)
```

```
ret = json.loads(ret)
```

```
for dic in ret['list']:
```

```
    title = dic['app_msg_ext_info']
```

```
['multi_app_msg_item_list'][0]['title']
```

```
detail_url = dic['app_msg_ext_info']  
['multi_app_msg_item_list'][0]['content_url']  
detail_page_text =  
requests.get(detail_url,headers=headers,verify=  
False).text
```

#图片链接保存在img标签的data-src属性值中

```
tree = etree.HTML(detail_page_text)  
img_list = tree.xpath('//img/@data-src')  
print(img_list)  
break
```