测试目标网站：<http://www.pycsgl.gov.cn/list.php?class=17>

正常网站我们请求url会返回给我们网页数据内容等，这个网站却返回了只包含js代码的html页面 如下:

< script type = "text/javascript" >

function stringToHex(str) {

var val = "";

for (var i = 0; i < str.length; i++) {

if (val == "") val = str.charCodeAt(i).toString(16);

else val += str.charCodeAt(i).toString(16);

}

return val;

}

function YunSuoAutoJump(){  
 var width =screen.width;  
 var height=screen.height;  
 var screendate = width + "," + height;  
 var curlocation = window.location.href;  
 if(-1 == curlocation.indexOf("security\_verify\_")){  
 document.cookie="srcurl=" + stringToHex(window.location.href) + ";path=/;";  
 }  
 self.location = "/list.php?class=17&security\_verify\_data=" + stringToHex(screendate);  
}

< /script>

<script>setTimeout("YunSuoAutoJump()", 50);</script >

分析返回的js分成两个部分。第一部分，定义了stringToHex和YunSuoAutoJump两个函数。第二部分，50毫秒后执行YunSuoAutoJump这个函数

YunSuoAutoJump这个函数功能是添加一个cookie并去请求一个构造的url，可以从document.cookie 和 self.location这里看出。stringToHex这个函数的共能其实就是字符串的转换（转换成16进制），

通过浏览器 调试网站的请求如下：

1. 我们在浏览器上输入请求目标网址，返回包涵如上js代码的页面，响应设置cookies【security\_session\_verify】
2. 执行第一步返回的js函数YunSuoAutoJump() 并头部加上名为【srcurl】的cookies 值再次请求 http://www.pycsgl.gov.cn/list.php?class=17&security\_verify\_data=xxx

得到一个响应cookie 【ecurity\_session\_mid\_verify】

1. 通过第二步的返回js内容可知,再次请求目标网站并且移除名称为【srcurl】的cookies，头部请求cookies 为【security\_session\_verify】 和【ecurity\_session\_mid\_verify】返回我们想要的数据

通过上述分析改网站的请求多了两个步骤，分别是对请求的url 编码+设置cookies 再发起特定的url请求得到某个cookies 通过第一步和第二步得到的cookies 值再次请求目标网址得到数据

**破解思路：**

1、用python 重写js函数

**def** stringToHex(string):  
 length = len(string)  
 hex\_string = str()  
 **for** i **in** range(length):  
 hex\_string += hex(ord(string[i]))[2:]  
 **return** hex\_string  
  
**def** get\_cookie(url):  
 hex\_string = stringToHex(url)  
 cookie = {**"srcurl"**: hex\_string}  
 **return** cookie

**def** get\_response\_cookies(cookies):  
 cook = {}  
 **for** cookie **in** cookies:  
 cook[cookie.name] = cookie.value  
 **return** cook

一个用于字符串转换，一个用于获取cookie，一个是发送请求是用的cookies 格式

1. 通过CookieJar 获取相应cookies

from scrapy.http.cookies import CookieJar # 该模块继承自内置的http.cookiejar,操作类似 # 实例化一个cookiejar对象

cookie\_jar = CookieJar()

3、部分代码如下

**def** handle\_next(self, response):  
 logger.info(**"请求 start url：%s"** % response.url)  
 url = response.meta.get(**'url'**)  
 cookiejar.extract\_cookies(response, response.request)  
 cookies = get\_response\_cookies(cookiejar)  
 logger.info(**"获取第一个cookie ：%s"** % cookies)  
 c = get\_cookie(self.url)  
 cookies.update(c)  
 **yield** Request(url=next\_url, callback=self.handle\_next\_t, dont\_filter=**True**,meta={**'url'**:url},cookies=cookies) #

**def** handle\_next\_t(self, response):  
 logger.info(**"请求中间url：%s"** % response.url)  
 cookiejar.extract\_cookies(response, response.request)  
 cookies = get\_response\_cookies(cookiejar)  
 logger.info(**"获取最终的cookie：%s"** % cookies)  
 url = response.meta.get(**'url'**)  
 **yield** Request(url=url, callback=self.parse, dont\_filter=**True**, cookies=cookies)  
  
**def** parse(self, response):  
 logger.info(**"再次请求 start url：%s"** % response.url)

# 获取最终数据