JS逆向反爬手段总结

JS逆向知识总结(查了很多资料，掉了很多头发)

一、JS逆向-反爬虫常见的四种方式

1.JS写cookie

浏览器一段JS生成一个（或多个）cookie再带着这个cookie做二次请求。服务器那边收到这个cookie就认为你的访问是通过浏览器过来的合法访问。

2.JS加密ajax请求参数

写爬虫抓某个网页里面的数据，发现网页源代码里面没有我们要的数据，那就有点麻烦了。那些数据往往是ajax请求得到的。
但是也不用怕，按F12打开Network窗口，刷新网页看看加载这个网页都下载了哪些URL，我们要的数据就在某个URL请求的结果里面。这类URL在Chrome的Network里面的类型大多是XHR。通过观察它们的“Response”就可以发现我们要的数据。
然而事情往往不是这么顺利，这个URL包含很多参数，某个参数是一串看上去无意义的字符串。这个字符串很可能是JS通过一个加密算法得到的，服务器也会通过同样的算法进行验证，验证通过了才认为你这是从浏览器来的请求。我们可以把这个URL拷贝到地址栏，把那个参数随便改个字母，访问一下看看是不是能得到正确的结果，由此来验证它是否是很重要的加密参数。
对于这样的加密参数，对策是通过debug JS来找到对应的JS加密算法。其中关键的是在Chrome里面设置“XHR/fetch Breakpoints”。

3.JS反调试（反debug）

前面我们都用到了Chrome 的F12去查看网页加载的过程，或者是调试JS的运行过程。这种方法用多了，网站就加了反调试的策略，只有我们打开F12，就会暂停在一个“debugger”代码行，无论怎样都跳不出去。它看起来像下面这样：

```
(function anonymous() {
        debugger
    })
```

不管我们点击多少次继续运行，它一直在这个“debugger”这里，每次都会多出一个VMxx的标签，观察“Call Stack”发现它好像陷入了一个函数的递归调用。这个“debugger”让我们无法调试JS。但是关掉F12窗口，网页就正常加载了。
解决这种JS反调试的方法我们称之为“反-反调试”，其策略是：通过“Call Stack”找到把我们带入死循环的函数，重新定义它。
这样的函数几乎没有任何其它功能只是给我们设置的陷阱。我们可以把这个函数在“Console”里面重新定义，比如把它重新定义为空函数，这样再运行它时就什么都不做，也就不会把我们带人陷阱。在这个函数调用的地方打个“Breakpoint”。因为我们已经在陷阱里面了，所以要刷新页面，JS的运行应该停止在设置的断点处，此时该函数尚未运行，我们在Console里面重新定义它，继续运行就可以跳过该陷阱。

4.JS发送鼠标点击事件

JS会响应链接被点击的事件，在打开链接前，先访问cl.gif，把当前的信息发送给服务器，然后再打开被点击的链接。服务器收到被点击链接的请求，会看看之前是不是已经通过demo.gif把对应信息发过来，如果发过来了就认为是合法的浏览器访

问，给出正常的网页内容。
因为requests没有鼠标事件响应就没有访问cl.gif的过程就直接访问链接，服务器就拒绝服务。

二、常见的JS混淆方法

1.js压缩：从源代码中删除不必要的字符，使代码看起来简单整洁，这种技术也被称为代码压缩和最小化。

破解方法代码格式化：

```
http://tool.oschina.net/codeformat/js/
```

2.Base62混淆：最明显的特征是生成的代码以eval(function(p,a,c,k,e,r)) 开头

这类混淆的关键思想在于将需要执行的代码进行一次编码，在执行的时候还原出浏览器可执行的合法的脚本
破解方法-浏览器
1打开 谷歌 或者 火狐 浏览器
2. 按 F12 打开控制台
3. 把代码复制进去
4. 删除开头 eval 这 4 个字母
5. 按 回车键

3.公钥加密关键字段

原理：
1.用户访问客户端，客户端向服务器请求获取一个 RSA 公钥以及键值 key ，存储在 本地
2.用户在本地公钥失效前发起登录请求，则使用已有公钥对用户密码进行加密；若已过期则执行 1 后再 加密
3.客户端将密文与 key 一起传回 后台4.后台通过 key 找到缓存里面的私钥，对密文进行解密
破解方法：

```
from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1v1_5
from Crypto.PublicKey import RSA
```

4.变量名混淆

原理：
字符串字面量混淆：首先提取全部的字符串，在全局作用域创建一个字符串数组，同时转义字符增大阅读难度，然后将字符串出现的地方替换成为数组元素的引用变量名混淆：不同于压缩器的缩短命名，此处使用了下划线加数字的格式，变量之间区分度很低，相比单个字母更难以阅读成员运算符混淆：将点运算符替换为字符串下标形式，然后对字符串进行混淆删除多余的空白字符：减小文件体积，这是所有压缩器都会做的事。
破解方法IDE 、解密工具、浏览器：

```
http://jsnice.org
```

```
2 http://js.51tools.info
```

个人课程，需要可以购买。