链滴

蘑菇街采集无水印资源（mw-sign 值计算）

作者：724555508
原文链接：https://ld246.com/article/1594022847457
来源网站：链滴

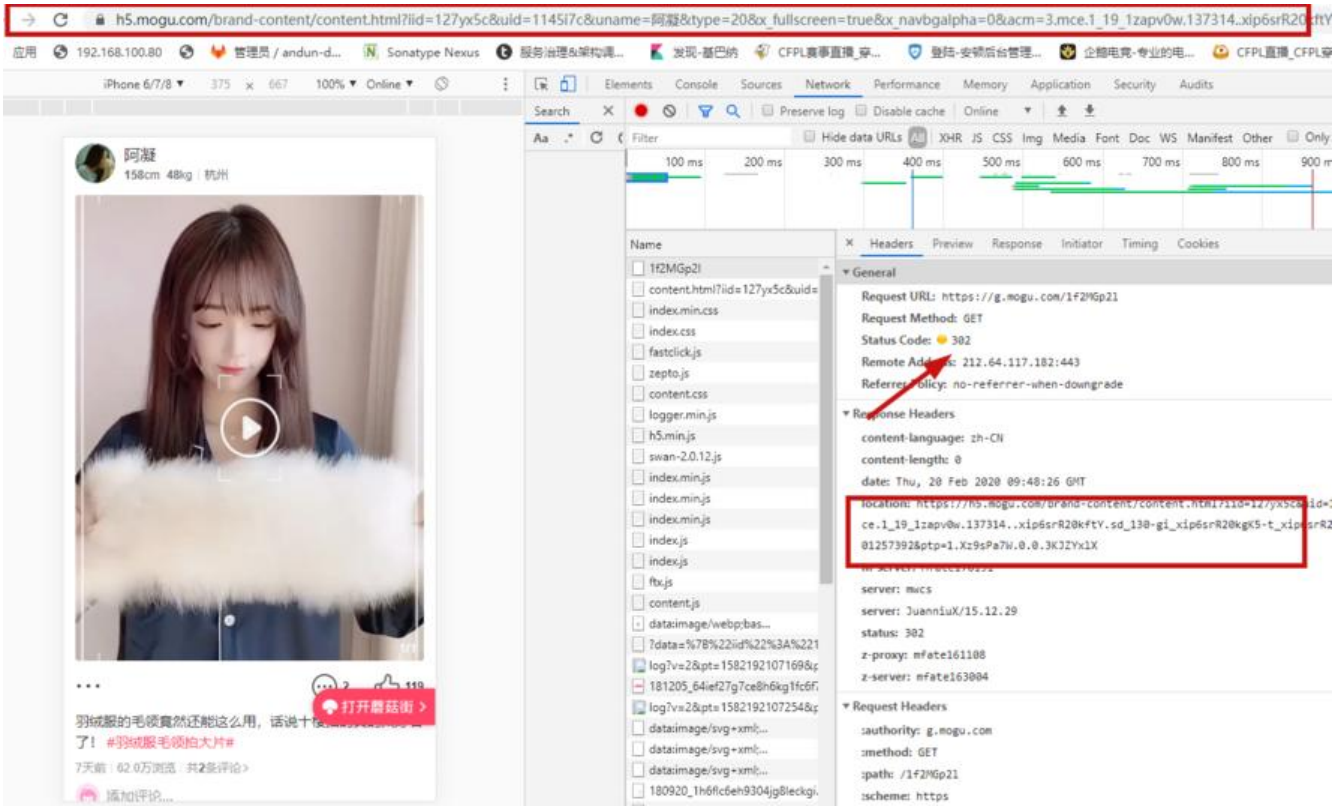
一。需求描述

通过“蘑菇街”App分享出来的链接获取无水印视频。**该内容只供学习交流，不可用于非法用途。**

二。分析接口

示例地址：https://g.mogu.com/1f2MGp2l

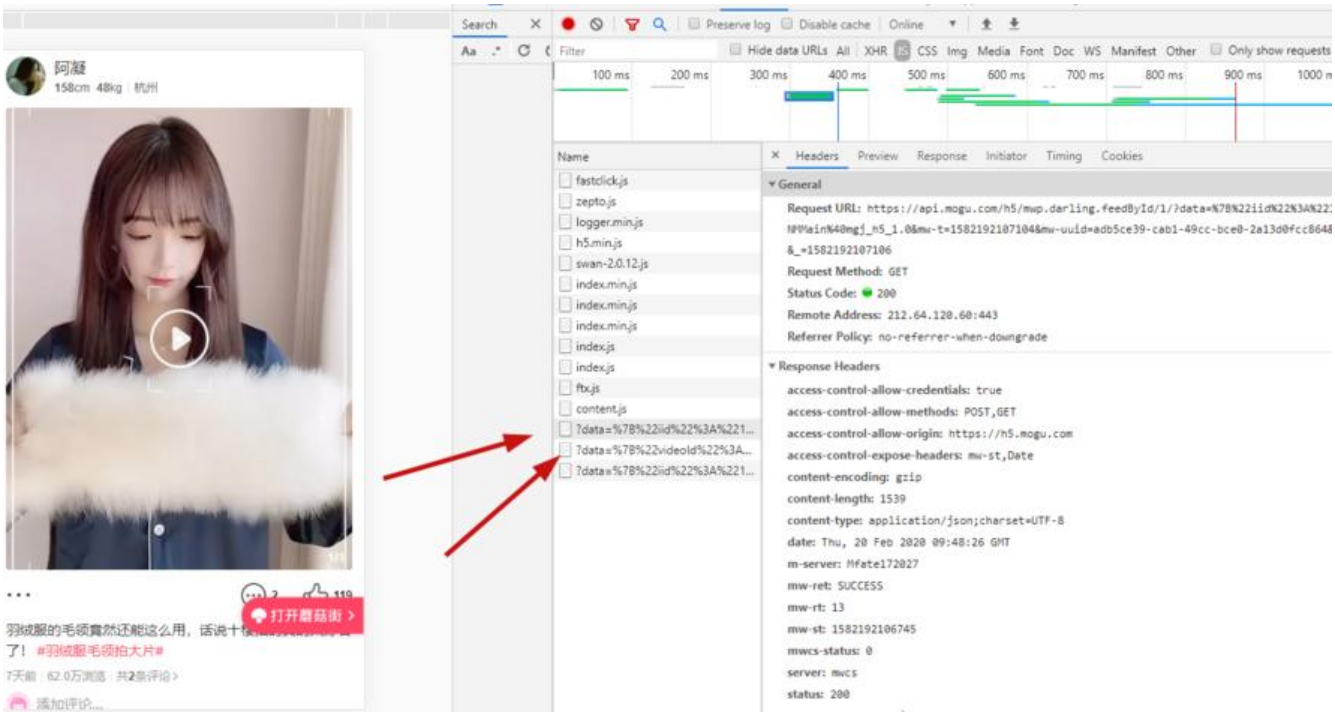
直接请求移动端的，这里就不示例web了，接口太多



原文链接：蘑菇街采集无水印资源（mw-sign 值计算）

如图，地址做了重定向，在ResponseHeader中拿到真实地址[location]

在js中找到了请求的接口



第一个接口是根据iid获取视频信息的。第二个接口是根据videoId获取视频播放路径的
(其实第一个接口也有返回视频url， 但是清晰度不够高，所有需要在第二个接口中取)

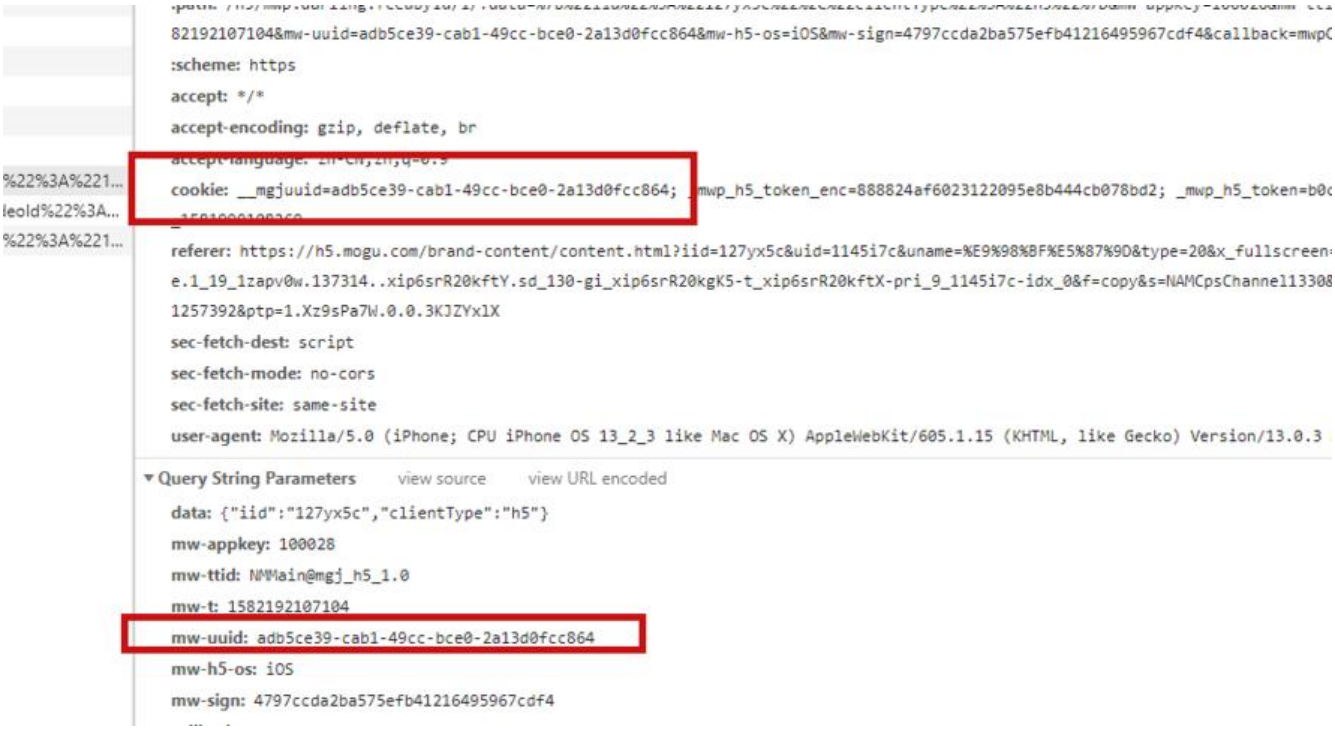
分析以下请求参数

param	是否必须	说明
data	是	数据json（包括id 和 cli ntType）
mw-appkey	是	100028
mw-ttid	是	NMMain@mgj_h5_ .0
mw-t	是	时间戳
mw-uuid	是	从cookie中获取
mw-h5-os	是	iOS
mw-sign	是	调用js中方法z获取
callback	否	回调

其中“data”是可以手动拼出来的数据，"mw-appkey","mw-ttid","mw-h5-os" 都是写死的参数值。"mw-t"为时间戳，剩下"mw-uuid" 和 "mw-sign"

三。获取mw-uuid

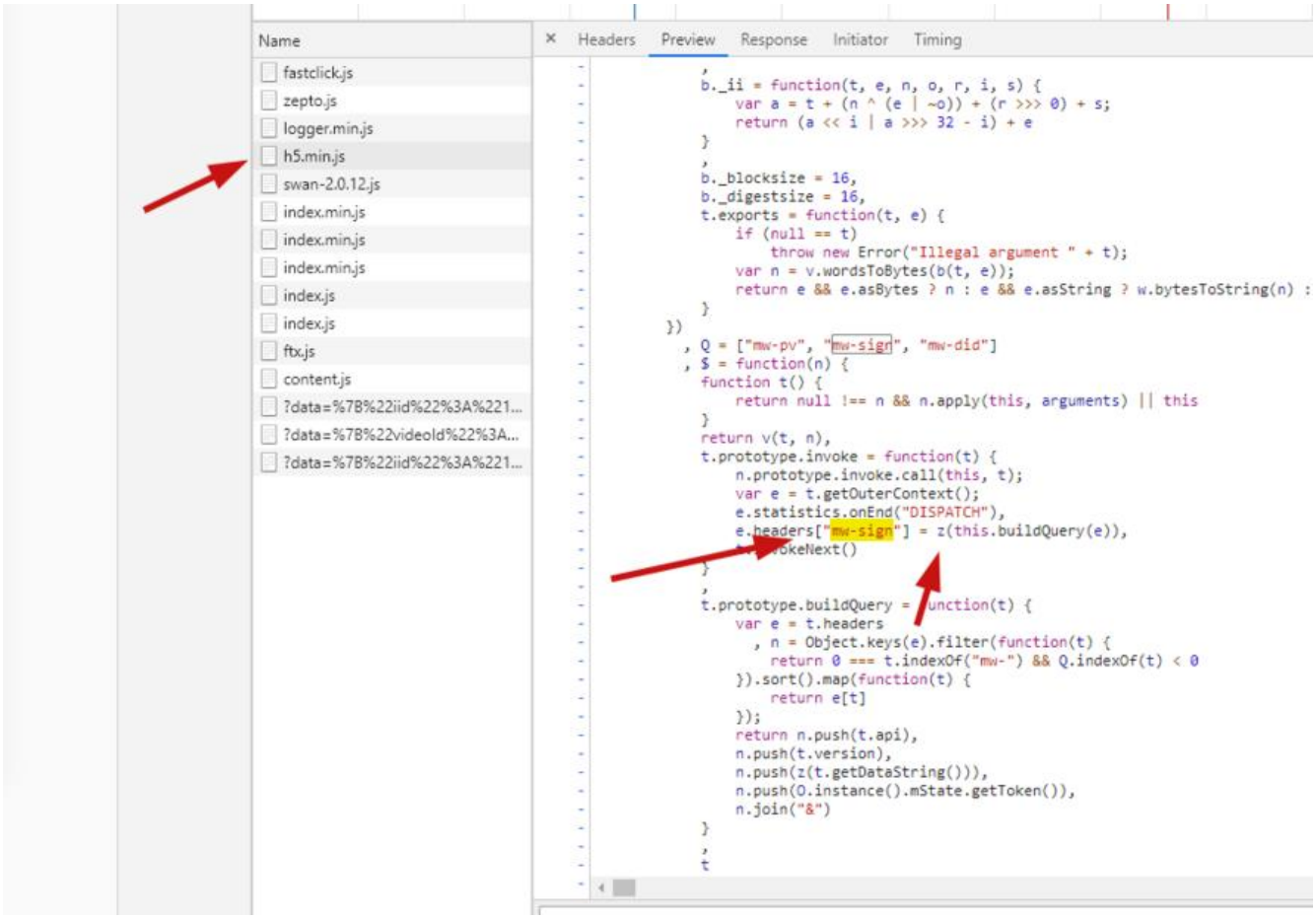
这里发现uuid跟cookie中的一致！！！



如何获取?

通过请求 https://list.mogujie.com 在cookie中获取uuid (一开始我也找不到，别人说从这儿获取hahaha)

四。获取mw-sign

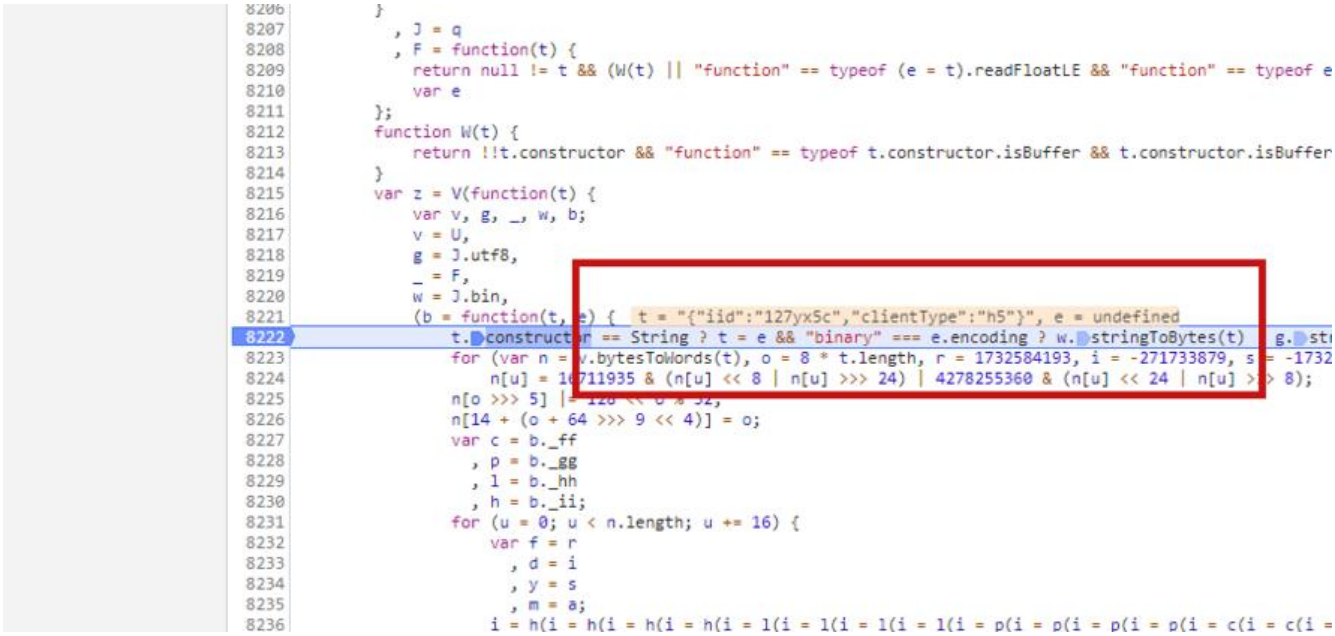


原文链接：蘑菇街采集无水印资源（mw-sign 值计算）

在js中搜索 mw-sign， 最终锁定在了这个地方，先buildQuery， 后调用了z方法，在buildQuery也调用了一次z方法。

现在是已经知道了实现方式所以可以直接找到位置，第一次分析的时候非常困难，可以尝试debugge分析z方法的入参

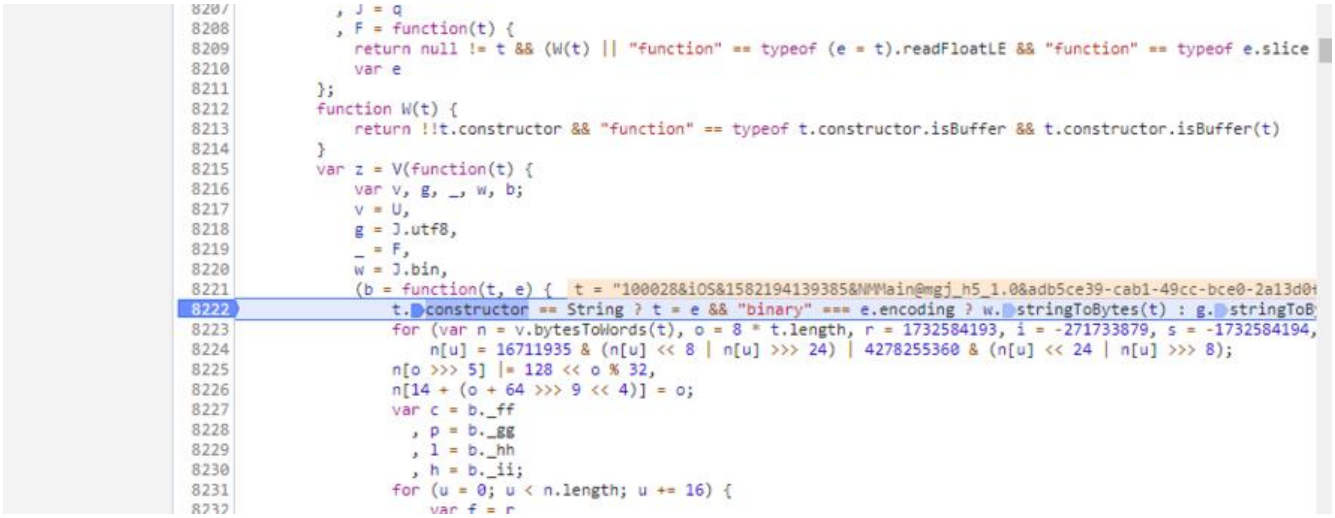
第一次调用z方法，入参为{"iid":"" , "clientType":"h5"}，有没有觉得很眼熟，这个就是刚刚分析请参数中的data



```
8206        }
8207          , J = q
8208          , F = function(t) {
8209            return null != t && (W(t) || "function" == typeof (e = t).readFloatLE && "function" == typeof e
8210            var e
8211        };
8212        function W(t) {
8213            return !!t.constructor && "function" == typeof t.constructor.isBuffer && t.constructor.isBuffer
8214        }
8215        var z = V(function(t) {
8216            var v, g, _, w, b;
8217            v = U,
8218            g = J.utf8,
8219            _ = F,
8220            w = J.bin,
8221            (b = function(t, e) {  t = "{"iid":"127yx5c","clientType":"h5"}", e = undefined
8222                t.constructor == String ? t = e && "binary" === e.encoding ? w.stringToBytes(t)  g.st
8223                for (var n = v.bytesToWords(t), o = 8 * t.length, r = 1732584193, i = -271733879, s = -1732
8224                    n[u] = 16711935 & (n[u] << 8 | n[u] >>> 24) | 4278255360 & (n[u] << 24 | n[u] >>> 8);
8225                n[o >>> 5] |= 128 << o % 32,
8226                n[14 + (o + 64 >>> 9 << 4)] = o;
8227                var c = b._ff
8228                  , p = b._gg
8229                  , l = b._hh
8230                  , h = b._ii;
8231                for (u = 0; u < n.length; u += 16) {
8232                    var f = r
8233                      , d = i
8234                      , y = s
8235                      , m = a;
8236                    i = h(i = h(i = h(i = h(i = l(i = l(i = l(i = l(i = p(i = p(i = p(i = p(i = c(i = c(i =
```

第二次调用z方法，入参为

100028&IOS&1564034676371&NMMain@mgj_pc_1.0&f65f8c1e-6286-4b4a-a917-5061e23855b&mwp.darling.feedById&1&b9cab4ab7f543491e2c4f6c556711345&39a9ae72d3faec64f17166036f84edd_1564026637963



```
8207          , J = q
8208          , F = function(t) {
8209            return null != t && (W(t) || "function" == typeof (e = t).readFloatLE && "function" == typeof e.slice
8210            var e
8211        };
8212        function W(t) {
8213            return !!t.constructor && "function" == typeof t.constructor.isBuffer && t.constructor.isBuffer(t)
8214        }
8215        var z = V(function(t) {
8216            var v, g, _, w, b;
8217            v = U,
8218            g = J.utf8,
8219            _ = F,
8220            w = J.bin,
8221            (b = function(t, e) {  t = "100028&iOS&1582194139385&NMMain@mgj_h5_1.0&adb5ce39-cab1-49cc-bce0-2a13d0
8222                t.constructor == String ? t = e && "binary" === e.encoding ? w.stringToBytes(t) : g.stringToB
8223                for (var n = v.bytesToWords(t), o = 8 * t.length, r = 1732584193, i = -271733879, s = -1732584194,
8224                    n[u] = 16711935 & (n[u] << 8 | n[u] >>> 24) | 4278255360 & (n[u] << 24 | n[u] >>> 8);
8225                n[o >>> 5] |= 128 << o % 32,
8226                n[14 + (o + 64 >>> 9 << 4)] = o;
8227                var c = b._ff
8228                  , p = b._gg
8229                  , l = b._hh
8230                  , h = b._ii;
8231                for (u = 0; u < n.length; u += 16) {
8232                    var f = r
```

按&截开，最后发现每个参数代表的是

"mw-appkey"，“mw-h5-os”，“mw-t”，“mw-ttid”，“mw-uuid”,以及部分请求地址(“mwp.darl

ng.feedById/1”)使用"&"拼接而成

b9cab4ab7f543491e2c4f6c556711345 这个值是第一次调用z方法的结果

39a9ae72d3faec64f157166036f84edd_1564026637963 这个是cookie中的_mwp_h5_token

如何获取mw-sign？ 第二次调用z方法返回的结果就是mw-sign

五。获取_mwp_h5_token

现在只查这个token了，直接说怎么获取吧

拼接好参数 调用api 携带mw-uuid（mw-sign 可以不带，因为我们这个时候并没有获取到mw-sign）会返回以下结果

{"api":"mwp.darling.feedById","v":"1","ret":"FAIL_SYS_TOKEN_NEED_RENEW","token":"6aaeb8f9371b395bfc417c0948f1e2c_1582195277778","encToken":"636565ab0ff6e4d5579a246e2f7bab7","needHeaderCookie":false}

这里这个token就是_mwp_h5_token， encToken就是_mwp_h5_token_enc，这两个值都有用，为一会儿再次请求的时候需要带上cookie

拿到token之后按照第二次调用z方法执行一次，获取出来的就是mw-sign了

六。最后请求获取数据

拼接好所有参数再次请求一次api（必须带上cookie，三个都要带），即可获取到响应数据。然后在边找自己用的信息即可

七。注意

1. 不同的接口请求地址不同，会影响第二次z方法加密
2. uuid / token 之类的数据会失效，具体能保持多久不知道，目前24小时是没问题的，可以存入缓。