爱奇艺视频 H5 解析分析过程

在正文开始之前先贴下关于实现这一过程的项目地址：

Github: **https://github.com/ZSAIm/iqiyi-parser**

说明：

这解析方式是基于爱奇艺 H5 播放方式的。

准备工具：

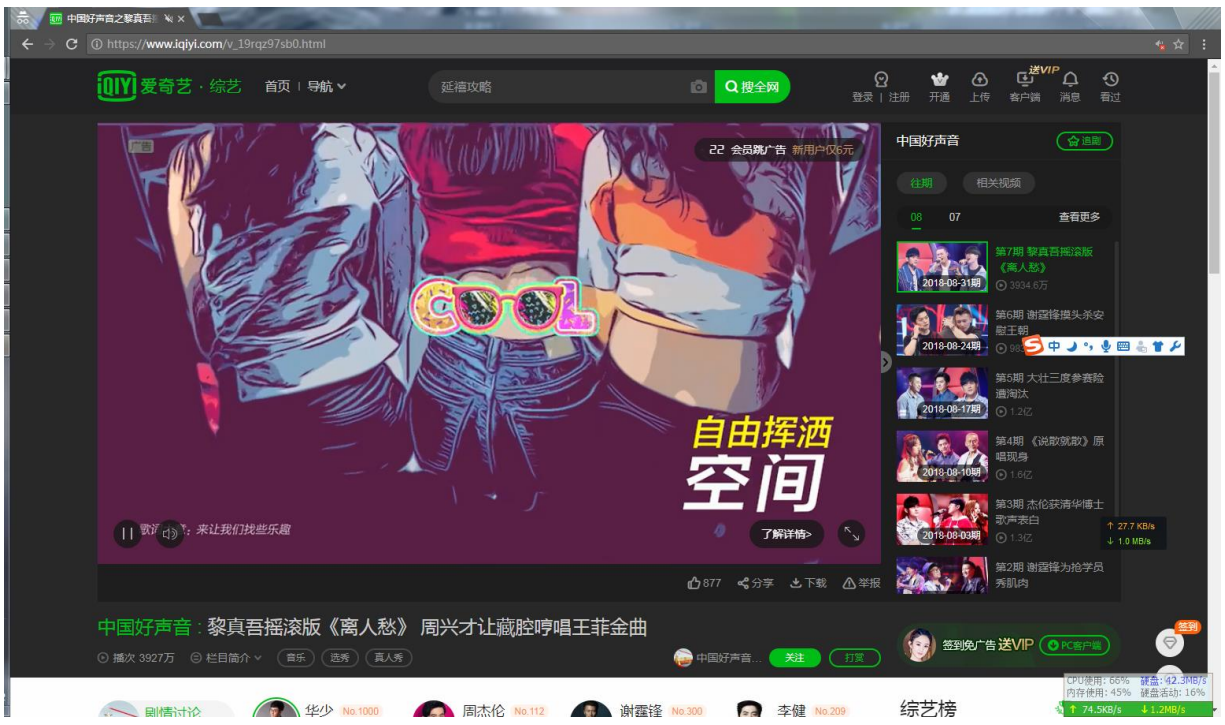
1) Fiddler（抓包工具）
2) 谷歌浏览器
3) Notepad++（文本编辑工具）（可选）
4) Webstorm（JS 调试工具）
5) Postman（HTTP 请求工具）
6) UltraCompare（文本比较）（可选）

正文：

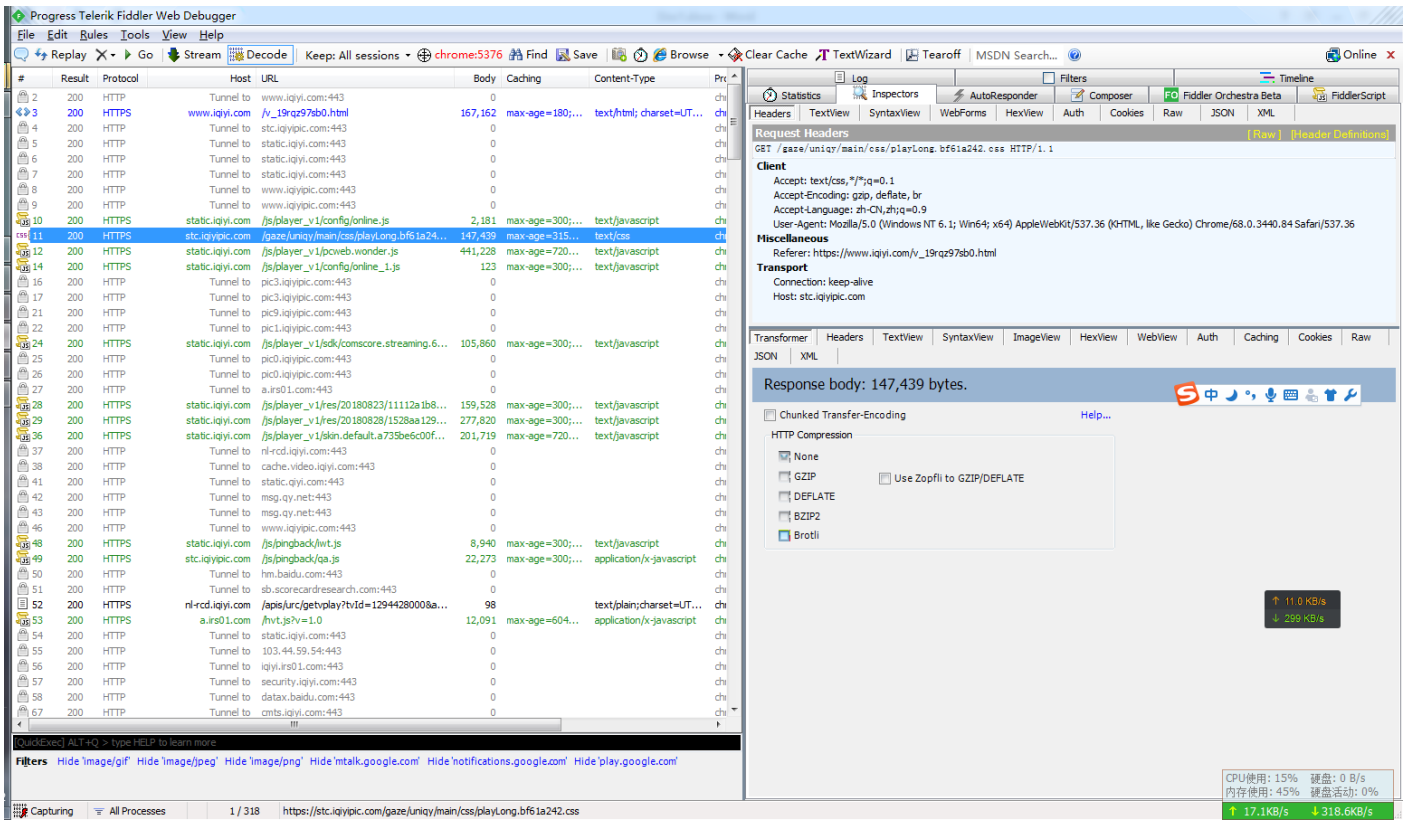
（由于论坛图片大小的限制，所以对图片进行了适当的压缩处理，如果需要高清图，请看 Github，路径/里面的 PDF 文件）

在使用谷歌浏览器之前，为了避免各种缓存的影响，务必先清除 Cache，Cookie……

打开其中一个爱奇艺的视频。（我以后面地址为例：https://www.iqiyi.com/v_19rqz97sb0.html）



使用 fiddler 进行抓包，然后把对分析无用的图像格式文件、css 之类的过滤掉。



等待视频开始缓存，为了保证已经加载视频，就等待广告过完之后再进行分析数据。

……

接下来就要开始分析需要的数据，当然首先要做的就是先再 fiddler 里面找到捕获的视频资源，

可以通过一下两种方式找到所需视频资源：

1）Body 项，可以通过找数据长度比较长的项。

2）Content-Type 项，文件流格式一般使用 application/octet-stream 类型。

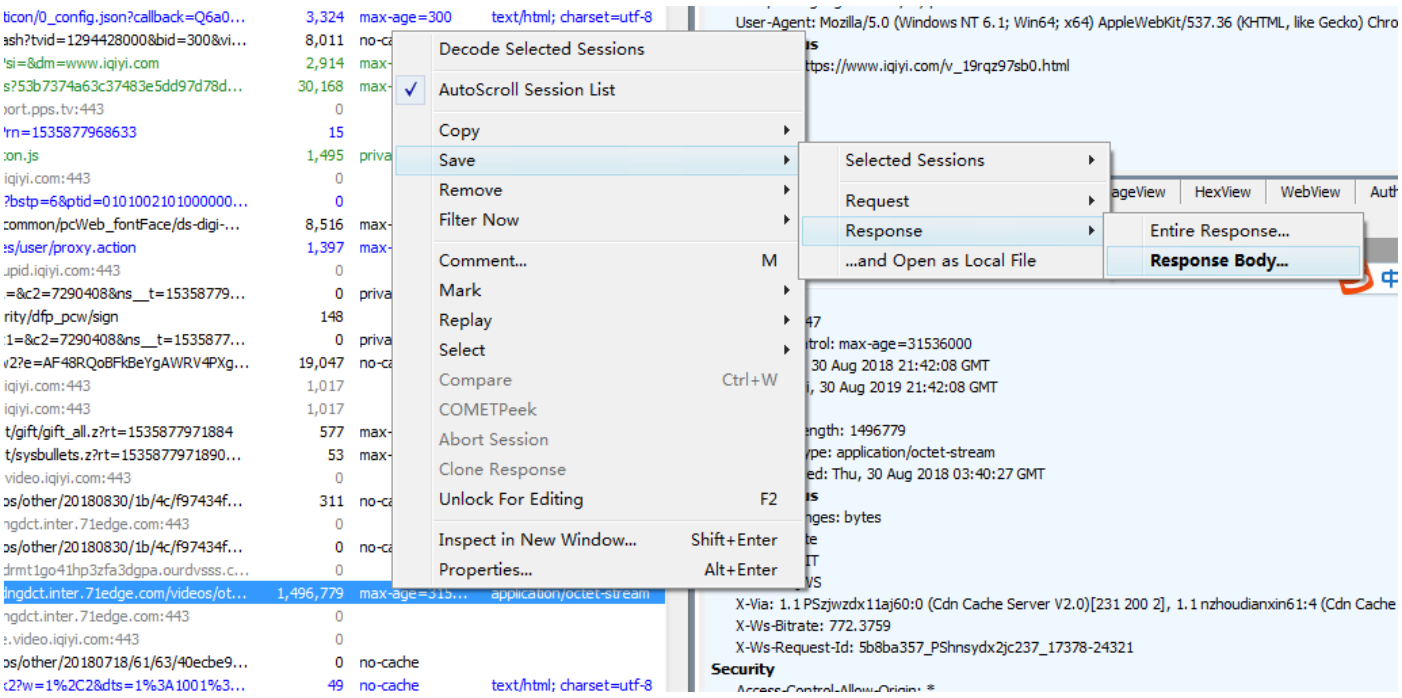
综合以上方式会比较好。

以下是抓包所得结果。（视频开始播放后，就可以停止继续抓包了，因为这东西挺占后台的。）

#	Result	Protocol	Host	URL	Body	Caching	Content-Type
58	200	HTTP	Tunnel to	datax.baidu.com:443	0		
67	200	HTTP	Tunnel to	cmts.iqiyi.com:443	0		
75	200	HTTPS	static.qiyi.com	/ext/common/pcw-v4-font/iconfont.woff	20,484	max-age=360...	application/octet-stream
79	200	HTTPS	msg.qy.net	/core?bstp=6&ptid=0101002101000000...	0		text/html
80	200	HTTPS	static.iqiyi.com	/ext/common/h5dpcfg.json?rn=1535877...	4,249	max-age=300;...	application/octet-stream
81	200	HTTPS	cmts.iqiyi.com	/emoticon/ep_config.json?callback=Q50...	10,502	max-age=300	text/html; charset=utf-8
82	200	HTTPS	iqiyi.irs01.com	/irt?_jwt_id=&_jwt_UA=UA-iqiyi-000001...	45	private,no-stor...	text/javascript
83	200	HTTPS	security.iqiyi.com	/static/cook/v1/cooksdk.js	133,066	max-age=300;...	text/javascript
84	200	HTTPS	cmts.iqiyi.com	/emoticon/0_config.json?callback=Q6a0...	3,324	max-age=300	text/html; charset=utf-8
85	200	HTTPS	cache.video.iqiyi.com	/jp/dash?tvid=1294428000&bid=300&vi...	8,011	no-cache	application/json; chars...
86	200	HTTPS	datax.baidu.com	/x.js?si=&dm=www.iqiyi.com	2,914	max-age=0, m...	application/javascript
87	200	HTTPS	hm.baidu.com	/hm.js?53b7374a63c37483e5dd97d78d...	30,168	max-age=0, m...	application/javascript
88	200	HTTP	Tunnel to	passport.pps.tv:443	0		
89	200	HTTPS	103.44.59.54	/3e8?rn=1535877968633	15		text/html
90	200	HTTPS	sb.scorecardresear...	/beacon.js	1,495	private, no-tra...	application/x-javascript
94	200	HTTP	Tunnel to	cook.iqiyi.com:443	0		
95	200	HTTPS	msg.qy.net	/core?bstp=6&ptid=0101002101000000...	0		text/html
98	200	HTTPS	static.qiyi.com	/ext/common/pcWeb_fontFace/ds-digi-...	8,516	max-age=360...	application/octet-stream
99	200	HTTPS	passport.pps.tv	/pages/user/proxy.action	1,397	max-age=315...	text/html; charset=utf-8
100	200	HTTP	Tunnel to	t7z.cupid.iqiyi.com:443	0		
102	302	HTTPS	sb.scorecardresear...	/b?c1=&c2=7290408&ns__t=15358779...	0	private, no-cac...	
103	200	HTTPS	cook.iqiyi.com	/security/dfp_pcw/sign	148		application/json;chars...
105	204	HTTPS	sb.scorecardresear...	/b2?c1=&c2=7290408&ns__t=1535877...	0	private, no-cac...	
106	200	HTTPS	t7z.cupid.iqiyi.com	/show2?e=AF48RQoBFkBeYgAWRV4PXg...	19,047	no-cache	text/plain; charset=utf-8
109	200	HTTP	Tunnel to	cmts.iqiyi.com:443	1,017		
110	200	HTTP	Tunnel to	cmts.iqiyi.com:443	1,017		
111	200	HTTPS	cmts.iqiyi.com	/bullet/gift/gift_all.z?rt=1535877971884	577	max-age=300;...	application/octet-stream
112	200	HTTPS	cmts.iqiyi.com	/bullet/sysbullets.z?rt=1535877971890...	53	max-age=300;...	application/octet-stream
114	200	HTTP	Tunnel to	data.video.iqiyi.com:443	0		
116	200	HTTPS	data.video.iqiyi.com	/videos/other/20180830/1b/4c/f97434f...	311	no-cache	text/plain
117	200	HTTP	Tunnel to	wscdngdct.inter.71edge.com:443	0		
119	302	HTTPS	wscdngdct.inter.71...	/videos/other/20180830/1b/4c/f97434f...	0	no-cache	
120	200	HTTP	Tunnel to	1ge3drmt1go41hp3zfa3dgpa.ourdvsss.c...	0		
121	200	HTTPS	1ge3drmt1go41hp3...	/wscdngdct.inter.71edge.com/videos/ot...	1,496,779	max-age=315...	application/octet-stream
122	200	HTTP	Tunnel to	wscdngdct.inter.71edge.com:443	0		
124	200	HTTP	Tunnel to	cache.video.iqiyi.com:443	0		
125	302	HTTPS	wscdngdct.inter.71...	/videos/other/20180718/61/63/40ecbe9...	0	no-cache	
126	200	HTTPS	t7z.cupid.iqiyi.com	/track2?w=1%2C2&dts=1%3A1001%3...	49	no-cache	text/html; charset=utf-8
127	200	HTTP	Tunnel to	t7z.cupid.iqiyi.com:443	943		
128	200	HTTPS	cache.video.iqiyi.com	/jp/vi/1294428000/5e54f1fec36034f675...	6,525	no-cache	application/json; chars...
129	200	HTTP	Tunnel to	t7z.cupid.iqiyi.com:443	0		

根据以上说的方法 1 找到下面的项

这是找到的一个看起来像是视频的文件，所以你可以尝试把响应的数据（Body）保存起来看看是不是你要的视频文件。（当然这可能无法播放）



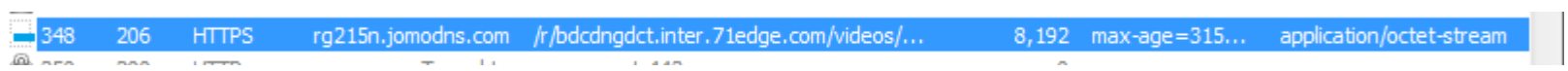
播放结果如下。

显然这不是所要的视频资源。

所以你可以不断尝试，直到找到你想要的视频缓存。

……

然后你就能找到以下这个项



可惜的是，这个文件却是无法播放的，但这并不代表他不属于这个视频资源的。但是如果找不到这个，也没问题的，你会找到下一片段的视频资源。

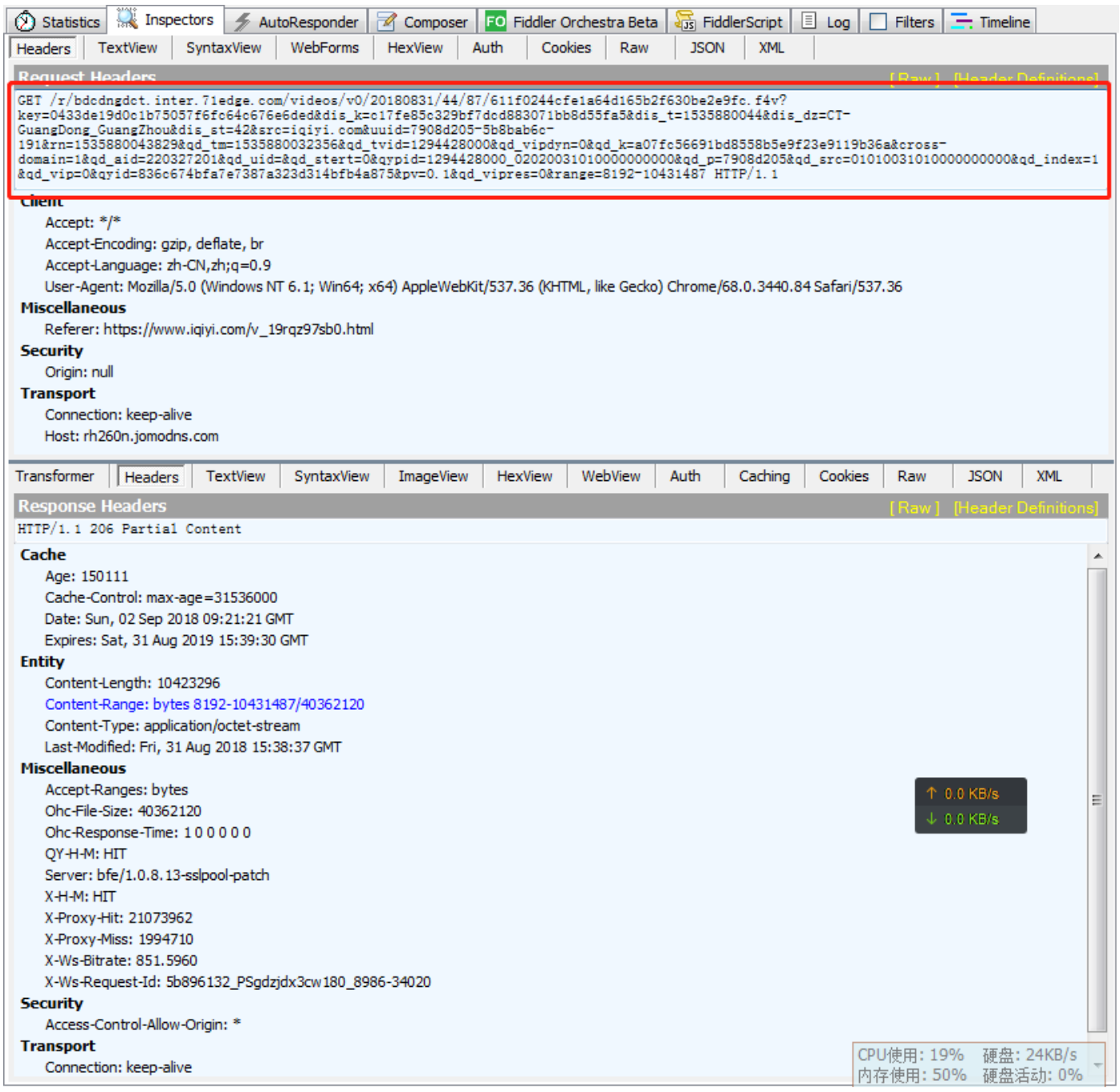
#	Result	Protocol	Host	URL	Body	Caching	Content-Type
327	200	HTTPS	pcw-api.iqiyi.com	/video/video/hotplaytimes/1207215800,...	417		application/json; chars...
328	200	HTTPS	v-7909dfcf.71edge...	/videos/other/20180816/63/ff/98e2034...	1,693,916	max-age=315...	application/octet-stream
331	200	HTTPS	data.video.iqiyi.com	/videos/v0/20180831/44/87/611f0244cf...	681	no-cache	text/plain
332	200	HTTP	Tunnel to	h-t-65e3135c.video.iqiyi.com:443	0		
333	200	HTTP	Tunnel to	bdcdngdct.inter.71edge.com:443	0		
334	101	HTTPS	h-t-65e3135c.video...	/ws	0		
335	302	HTTPS	bdcdngdct.inter.71...	/videos/v0/20180831/44/87/611f0244cf...	0	no-cache	text/html
336	200	HTTP	Tunnel to	rh260n.jomodns.com:443	766		
337	206	HTTPS	rh260n.jomodns.com	/r/bdcdngdct.inter.71edge.com/videos/...	8,192	max-age=315...	application/octet-stream
338	200	HTTP	Tunnel to	sb.scorecardresearch.com:443	0		
341	200	HTTP	Tunnel to	www.googleapis.com:443	0		
343	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8
346	302	HTTP	ckm.iqiyi.com	/pixel	1	no-cache	
347	302	HTTP	397c0.admaster.co...	/i/a111576,b2778301,c1118,i0,m202,8a...	0	private, no-cac...	text/html
348	200	HTTP	Tunnel to	msg.qy.net:443	0		
351	302	HTTP	ckm.iqiyi.com	/pixel?qiyi_nid=71000080	1	no-cache	
359	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8
362	200	HTTP	Tunnel to	msg.qy.net:443	780		
369	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8
371	302	HTTP	ckm.iqiyi.com	/pixel	1	no-cache	
372	302	HTTP	ad.doubleclick.net	/ddm/trackimp/N5050.2207IQIYI.COM/B...	0	no-cache, mus...	text/html; charset=UT...
373	200	HTTP	Tunnel to	msg.qy.net:443	0		
374	200	HTTP	Tunnel to	msg.qy.net:443	780		
377	302	HTTP	cm.ipinyou.com	/qiyi/cms.gif?qiyi_uid=7462b59d2be0c5...	0	no-cache; Expi...	
379	302	HTTP	ipinyou.cm.admaste...	/ipinyou/?tid=1277&type=1&uid=I92HL...	0	private, no-cac...	text/html
381	302	HTTPS	bdcdngdct.inter.71...	/videos/v0/20180831/44/87/611f0244cf...	0	no-cache	text/html
382	206	HTTPS	rh260n.jomodns.com	/r/bdcdngdct.inter.71edge.com/videos/...	10,423,296	max-age=315...	application/octet-stream
386	-	HTTP	Tunnel to	www.googleapis.com:443	-1		
390	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8
392	302	HTTP	ckm.iqiyi.com	/pixel	1	no-cache	
393	302	HTTP	397c0.admaster.co...	/i/a113356,b2800251,c1118,i0,m202,8a...	0	private, no-cac...	text/html
394	200	HTTP	Tunnel to	msg.qy.net:443	780		
397	302	HTTP	c.yes.youku.com	/cm.gif?dspid=11210	154	no-cache; Expi...	text/html
405	200	HTTPS	t7z.cupid.iqiyi.com	/track2?w=8&dts=8%3A1001%3A&nr=...	48	no-cache	text/html; charset=utf-8
406	200	HTTP	Tunnel to	iqiyi.irs01.com:443	777		
407	200	HTTPS	msg.qy.net	/core?bstp=6&ptid=0101002101000000...	0		text/html
408	200	HTTP	Tunnel to	msg.qy.net:443	0		
409	200	HTTPS	iqiyi.irs01.com	/hvt?title=%E4%B8%AD%E5%9B%BD...	35	private,no-stor...	text/javascript
410	200	HTTP	Tunnel to	msg.qy.net:443	0		
412	200	HTTP	Tunnel to	sb.scorecardresearch.com:443	0		
415	200	HTTPS	msg.qy.net	/core?bstp=6&ptid=0101002101000000...	0		text/html

当我们知道图片的下面一项的时候，你就很快能找到剩余的片段了。

为什么？因为你可以发现这两个请求的 URL 路径一样，而且下面的哪一个片段是不完整的视频，所以不可否认的是，爱奇艺把视频片段分割成了很多片段。

什么？你觉得牵强？对，确实是这样的。但是不如这么想，如果你觉得这不是可疑的视频，那么当你不能在 Fiddler 里面找到除了这些片段的可疑播放的视频的时候你就不会这么想了，因为不管怎么样，你要相信 fiddler 已经把视频捕获到了。而且视频肯定也是存在的，不然你怎么能再页面取观看呢对吧。

既然这样，那么我们就分析 382 这一项。要干什么呢？当然首先是看下这个地址。



这恐怖的 GET 请求，一大堆符号。

GET /r/bdcdngdct.inter.71edge.com/videos/v0/20180831/44/87/611f0244cfe1a64d165b2f630be2e9fc.f4v?key=0433de19d0c1b75057f6fc64c676e6ded&dis_k=c17fe85c329bf7dcd883071bb8d55fa5&dis_t=1535880044&dis_dz=CT-GuangDong_GuangZhou&dis_st=42&src=iqiyi.com&uuid=7908d205-5b8bab6c-191&rn=1535880043829&qd_tm=1535880032356&qd_tvid=1294428000&qd_vipdyn=0&qd_k=a07fc56691bd8558b5e9f23e9119b36a&cross-domain=1&qd_aid=220327201&qd_uid=&qd_stert=0&qypid=1294428000_020200310100000000000&qd_p=7908d205&qd_src=01010031010000000000&qd_index=1&qd_vip=0&qyid=836c674bfa7e7387a323d314bfb4a875&pv=0.1&qd_vipres=0&range=8192-10431487 HTTP/1.1

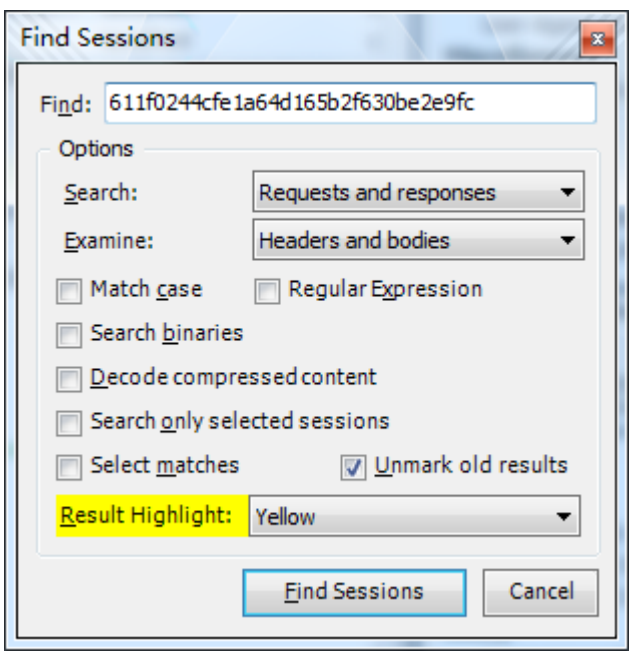
我们要做的就是构造这一串符号，并且是有依据的构造。

从前面往后看，寻找最后一层目录，得到服务器上的文件是这个

611f0244cfe1a64d165b2f630be2e9fc.f4v?key=0433de19d0c1b75057f6fc64c676e6ded&dis_k=c17fe85c329bf7dcd883071bb8d55fa5&dis_t=1535880044&dis_dz=CT-GuangDong_GuangZhou&dis_st=42&src=iqiyi.com&uuid=7908d205-5b8bab6c-191&rn=1535880043829&qd_tm=1535880032356&qd_tvid=1294428000&qd_vipdyn=0&qd_k=a07fc56691bd8558b5e9f23e9119b36a&cross-domain=1&qd_aid=220327201&qd_uid=&qd_stert=0&qypid=1294428000_020200310100000000000&qd_p=7908d205&qd_src=01010031010000000000&qd_index=1&qd_vip=0&qyid=836c674bfa7e7387a323d314bfb4a875&pv=0.1&qd_vipres=0&range=8192-10431487

为了构造他，肯定是先找到这串字符最先出现的地方了。先忽略?号后面的参数，不如先找到 611f0244cfe1a64d165b2f630be2e9fc（因为一眼看去他是最突出的）

这是哪里得到的吧。这就要用到 fiddler 强大的搜索工具了。

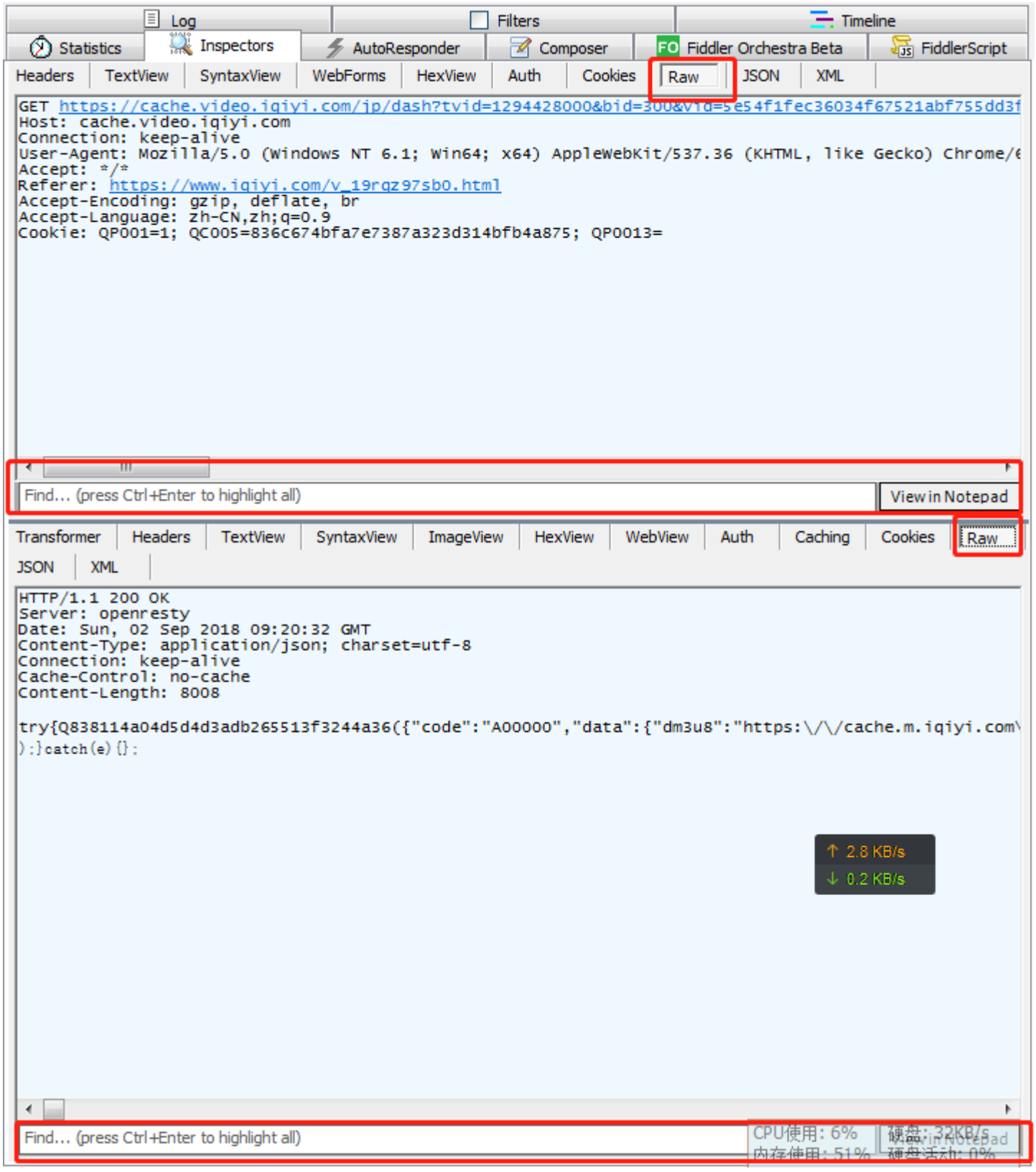


#	Result	Protocol	Host	URL	Body	Caching	Content-Type	P
322	200	HTTPS	v-7909dfcf.71edge...	/videos/other/20180820/c8/81/1ef7a5a...	3,762,793	max-age=315...	application/octet-stream	c
324	200	HTTP	Tunnel to	control-i.iqiyi.com:443	0			c
325	200	HTTPS	msg.qy.net	/b?t=218pf=1&p=10&p1=101&block=8...	0		text/html	c
326	200	HTTPS	control-i.iqiyi.com	/control/content_config?business=paop...	198		text/html;charset=utf-8	c
327	200	HTTPS	pcw-api.iqiyi.com	/video/video/hotplaytimes/1207215800,...	417		application/json; chars...	c
328	200	HTTPS	v-7909dfcf.71edge...	/videos/other/20180816/63/ff/98e2034...	1,693,916	max-age=315...	application/octet-stream	c
331	200	HTTPS	data.video.iqiyi.com	/videos/v0/20180831/44/87/611f0244cf...	681	no-cache	text/plain	c
332	200	HTTP	Tunnel to	h-t-65e3135c.video.iqiyi.com:443	0			c
333	200	HTTP	Tunnel to	bdcdngdct.inter.71edge.com:443	0			c
334	101	HTTPS	h-t-65e3135c.video...	/ws	0			c
335	302	HTTPS	bdcdngdct.inter.71...	/videos/v0/20180831/44/87/611f0244cf...	0	no-cache	text/html	c
336	200	HTTP	Tunnel to	rh260n.jomodns.com:443	766			c
337	206	HTTPS	rh260n.jomodns.com	/r/bdcdngdct.inter.71edge.com/videos/...	8,192	max-age=315...	application/octet-stream	c
338	200	HTTP	Tunnel to	sb.scorecardresearch.com:443	0			c
341	200	HTTP	Tunnel to	www.googleapis.com:443	0			c
343	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8	c
346	302	HTTP	ckm.iqiyi.com	/pixel	1	no-cache		c
347	302	HTTP	397c0.admaster.co...	/i/a111576,b2778301,c1118,i0,m202,8a...	0	private, no-cac...	text/html	c
348	200	HTTP	Tunnel to	msg.qy.net:443	0			c
351	302	HTTP	ckm.iqiyi.com	/pixel?qiyi_nid=71000080	1	no-cache		c
359	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8	c
362	200	HTTP	Tunnel to	msg.qy.net:443	780			c
369	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8	c
371	302	HTTP	ckm.iqiyi.com	/pixel	1	no-cache		c
372	302	HTTP	ad.doubleclick.net	/ddm/trackimp/N5050.2207IQIYI.COM/B...	0	no-cache, mus...	text/html; charset=UT...	c
373	200	HTTP	Tunnel to	msg.qy.net:443	0			c
374	200	HTTP	Tunnel to	msg.qy.net:443	780			c
377	302	HTTP	cm.ipinyou.com	/qiyi/cms.gif?qiyi_uid=7462b59d2be0c5...	0	no-cache; Expi...		c
379	302	HTTP	ipinyou.cm.admaste...	/ipinyou/?tid=1277&type=1&uid=I92HL...	0	private, no-cac...	text/html	c
381	302	HTTPS	bdcdngdct.inter.71...	/videos/v0/20180831/44/87/611f0244cf...	0	no-cache	text/html	c
382	206	HTTPS	rh260n.jomodns.com	/r/bdcdngdct.inter.71edge.com/videos/...	10,423,296	max-age=315...	application/octet-stream	c
386	200	HTTP	Tunnel to	www.googleapis.com:443	0			c
390	200	HTTPS	t7z.cupid.iqiyi.com	/track2?f=8e5d53f73426b4787a20f0b3...	48	no-cache	text/html; charset=utf-8	c
392	302	HTTP	ckm.iqiyi.com	/pixel	1	no-cache		c
393	302	HTTP	397c0.admaster.co...	/i/a113356,b2800251,c1118,i0,m202,8a...	0	private, no-cac...	text/html	c
394	200	HTTP	Tunnel to	msg.qy.net:443	780			c
397	302	HTTP	c.yes.youku.com	/cm.gif?dspid=11210	154	no-cache; Expi...	text/html	c
405	200	HTTPS	t7z.cupid.iqiyi.com	/track2?w=8&dts=8%3A1001%3A&nr=...	48	no-cache	text/html; charset=utf-8	c
406	200	HTTP	Tunnel to	iqiyi.irs01.com:443	777			c
407	200	HTTPS	msg.qy.net	/core?bstp=6&ptid=0101002101000000...	0		text/html	c
408	200	HTTP	Tunnel to	msg.qy.net:443	0			c

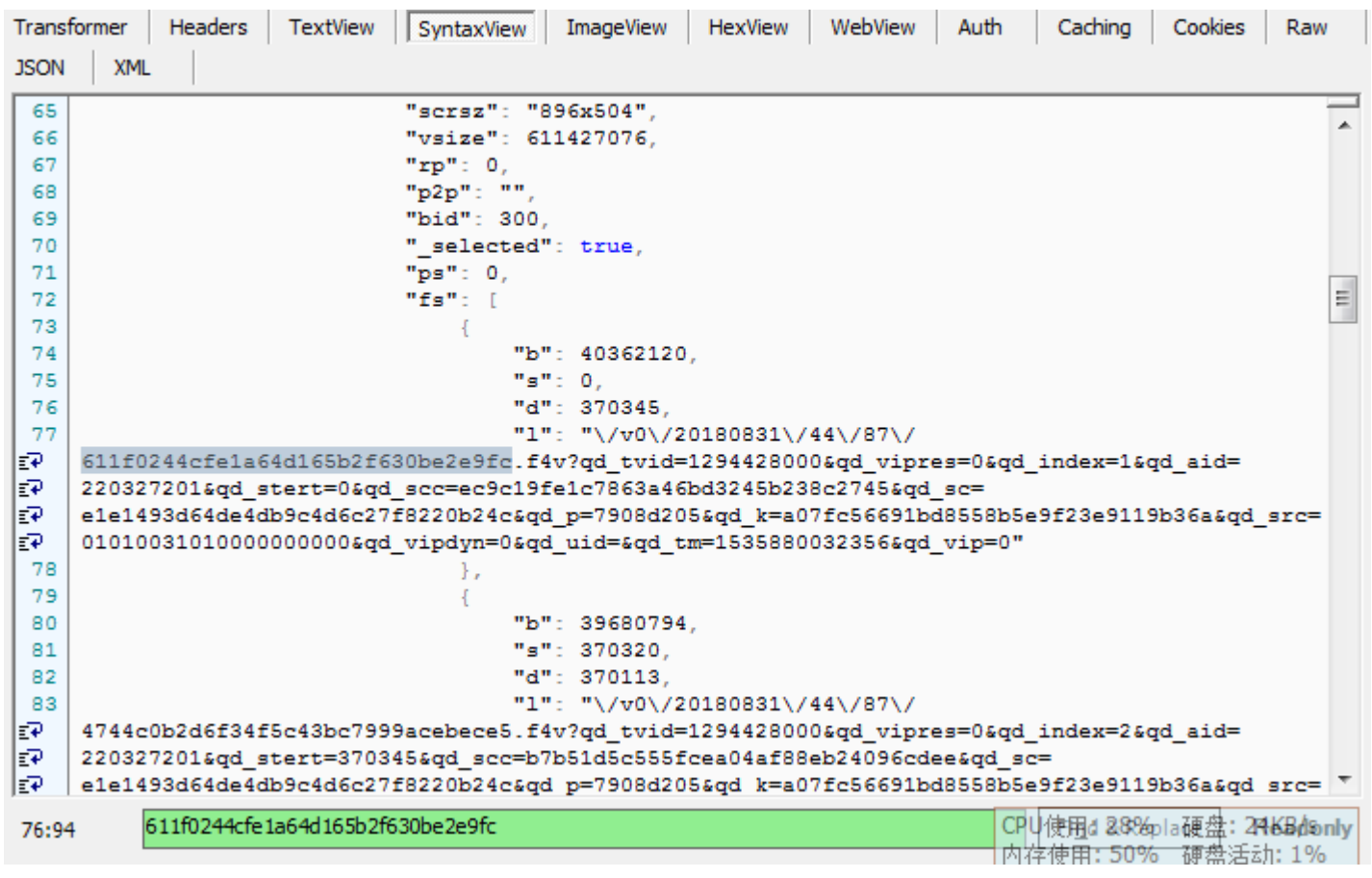
我们要找的就是从前往后寻找最早出现这串字符的项。

52	200	HTTPS	cache.video.iqiyi.com	/jp/dash?tvid=1294428000&bid=300&vi...	8,008	no-cache	application/json; chars...	c

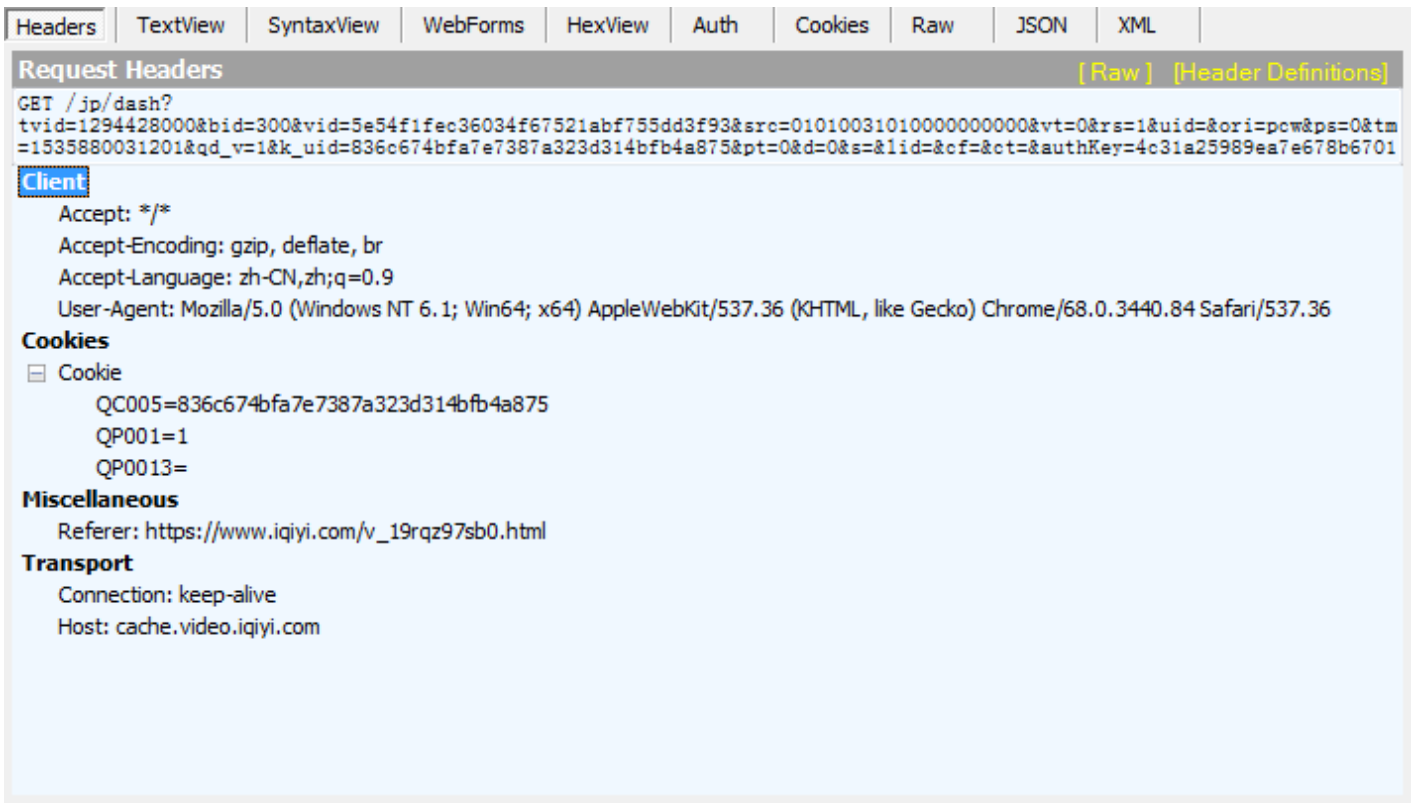
然后进一步寻找这一串字符出现在哪里。（也就是在一下红框里面各自寻找这一串字符）



搜索结果如下，在响应中



那么既然这一串字符出现在这里，那么必然要先知道这一项的 URL，并且构造他。



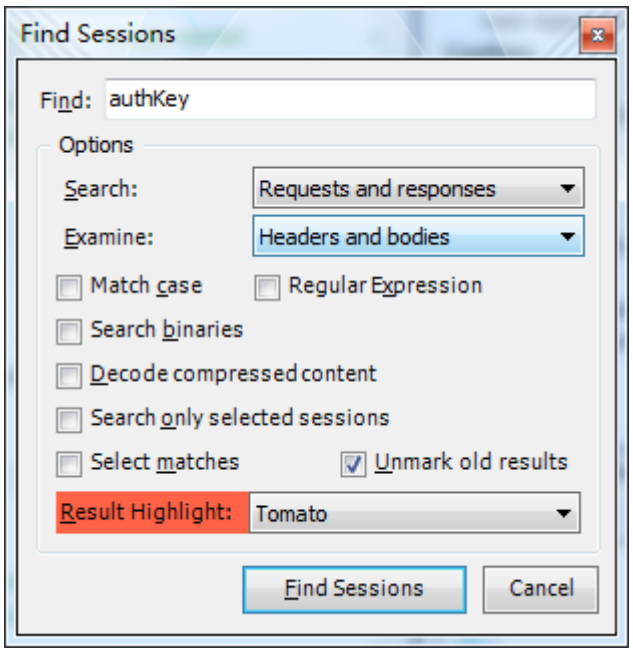
GET /jp/dash?tvid=1294428000&bid=300&vid=5e54f1fec36034f67521abf755dd3f93&src=0101003101000000000&vt=0&rs=1&uid=&ori=pcw&ps=0&tm=1535880031201&qd_v=1&k_uid=836c674bfa7e7387a323d314bfb4a875&pt=0&d=0&s=&lid=&cf=&ct=&authKey=4c31a25989ea7e678b670125e6ee5acf&k_tag=1&ost=0&ppt=0&dfp=&locale=zh_cn&prio=%7B%22ff%22%3A%22f4v%22%2C%22code%22%3A2%7D&pck=&k_err_retries=0&k_ft1=549755813888&bop=%7B%22version%22%3A%227.0%22%2C%22dfp%22%3A%22%22%7D&callback=Q838114a04d5d4d3adb265513f3244a36&ut=0&vf=a07fc56691bd8558b5e9f23e9119b36a HTTP/1.1

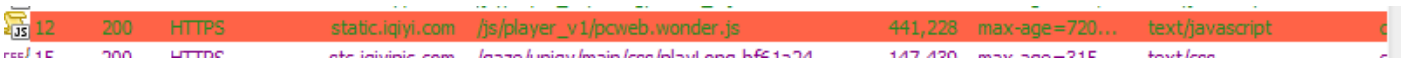
又是一长串字符了。

整理参数如下

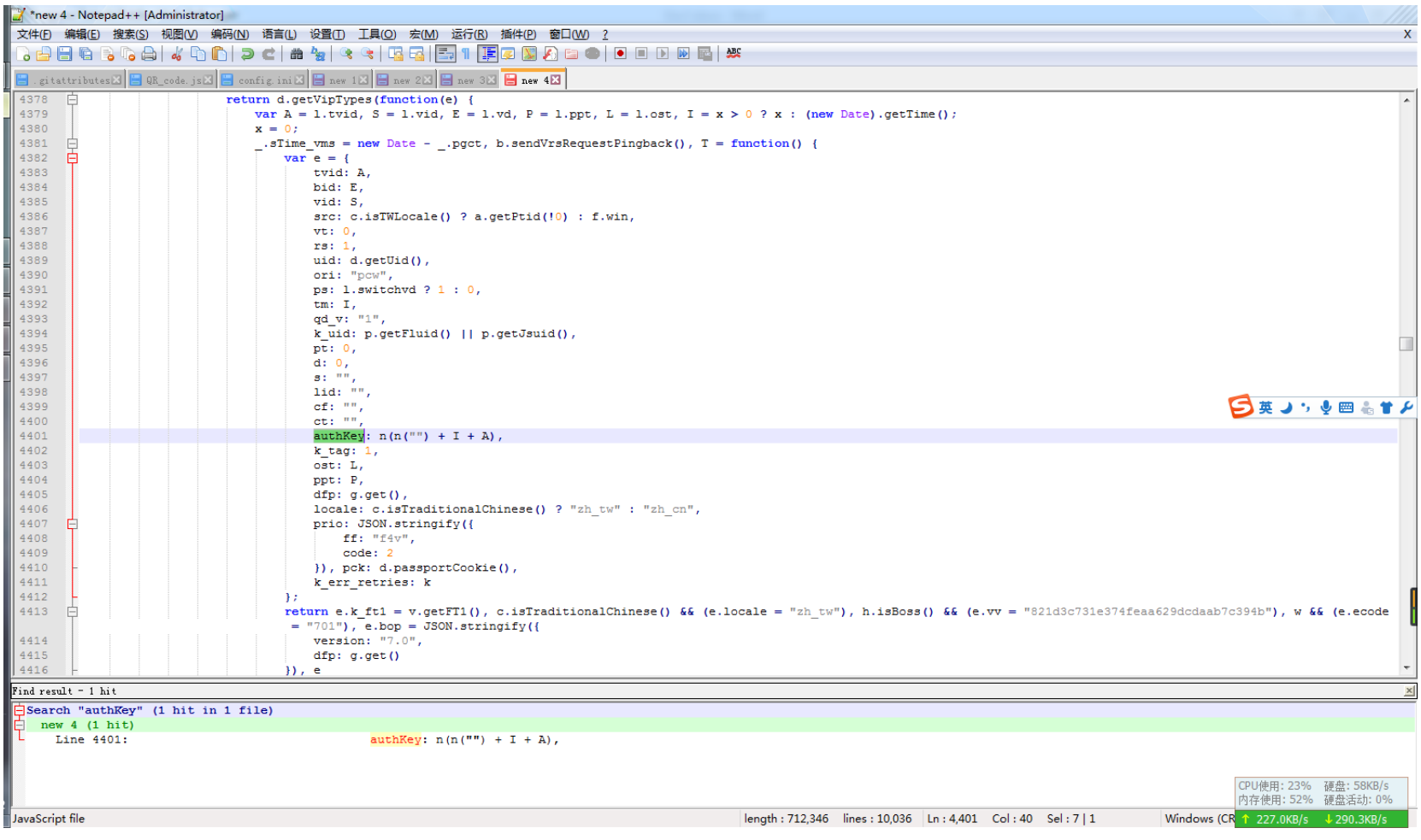
tvid=1294428000
&bid=300
&vid=5e54f1fec36034f67521abf755dd3f93
&src=01010031010000000000
&vt=0
&rs=1
&uid=
&ori=pcw
&ps=0
&tm=1535880031201
&qd_v=1
&k_uid=836c674bfa7e7387a323d314bfb4a875
&pt=0
&d=0
&s=
&lid=
&cf=
&ct=
&authKey=4c31a25989ea7e678b670125e6ee5acf
&k_tag=1
&ost=0
&ppt=0
&dfp=
&locale=zh_cn
&prio=%7B%22ff%22%3A%22f4v%22%2C%22code%22%3A2%7D
&pck=
&k_err_retries=0
&k_ft1=549755813888
&bop=%7B%22version%22%3A%227.0%22%2C%22dfp%22%3A%22%22%7D
&callback=Q838114a04d5d4d3adb265513f3244a36
&ut=0
&vf=a07fc56691bd8558b5e9f23e9119b36a

看英文也可以大概的知道 authKey 可能就是加密方式。所以你要在 fiddler 搜索 authKey，（其实我这么做是为了找到构造这个链接所对应的 js）





这是最早出现 authKey 的地方，为了方便分析，把他复制到 notepad++里面分析。



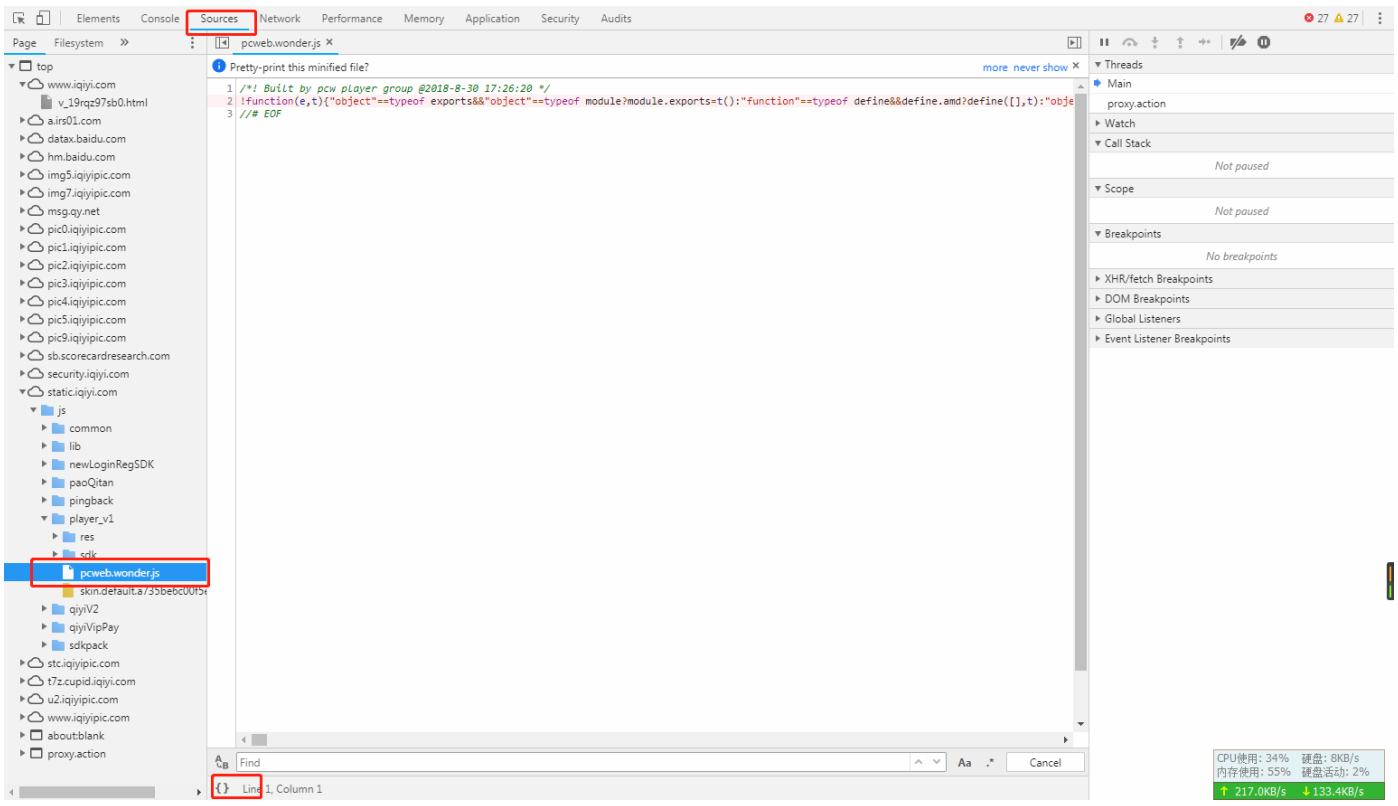
只找到一个结果。

而且回头看一下这些参数是不是很熟悉呢？大部分的参数他都已经给出来了。

authKey: n(n("") + I + A),

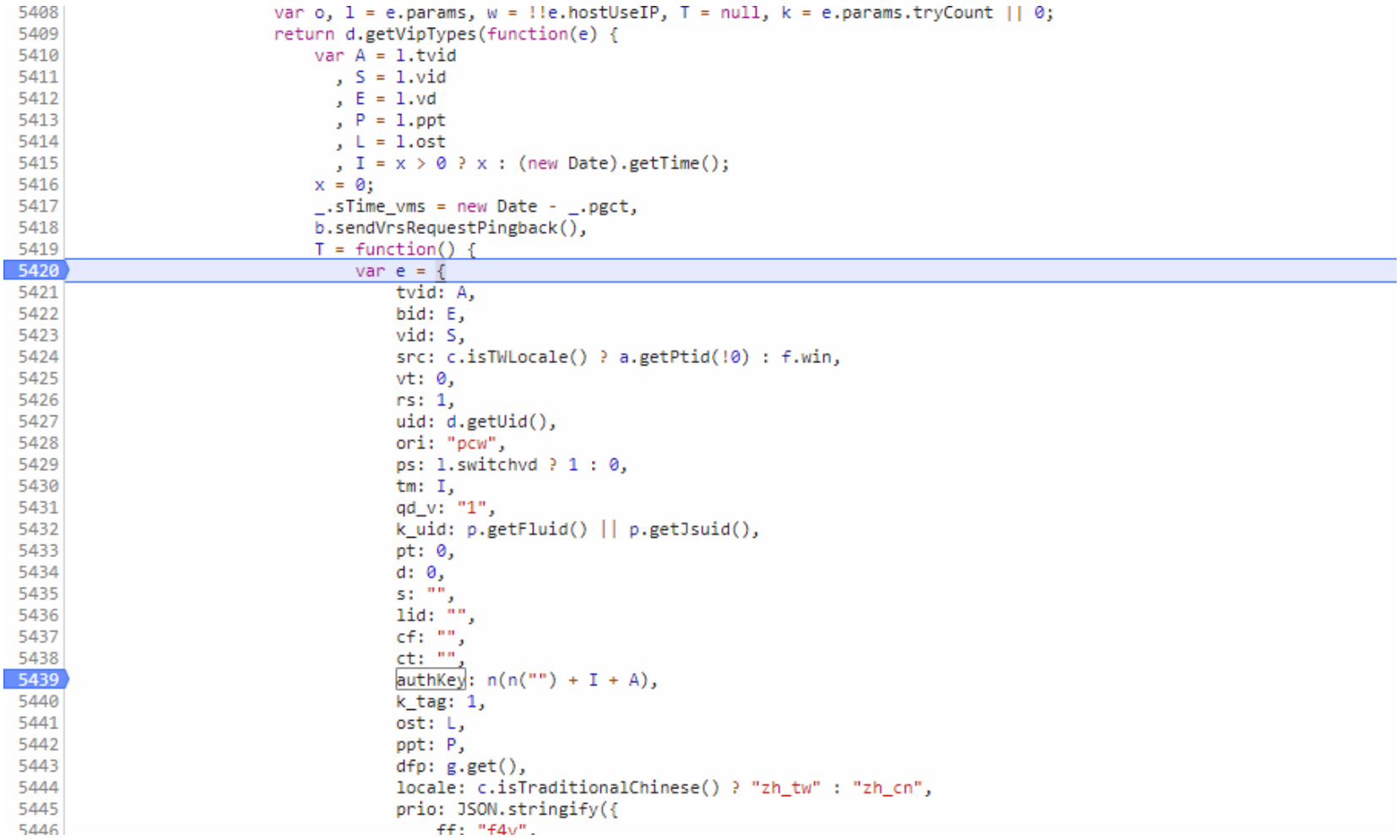
当然这里面都是 Js 加密的结果，所以我们只要找到函数 n 和 A 和 I 的值就能构造出 authKey 了。当然我们不可能吧这个 js 代码分析一遍，所以我们就要借助谷歌浏览器的调试工具了。

谷歌浏览器按 F12 弹出工具。并且在 Sources 找到这个 js 文件，并把他格式化左下角的{}，（为了方便分析）



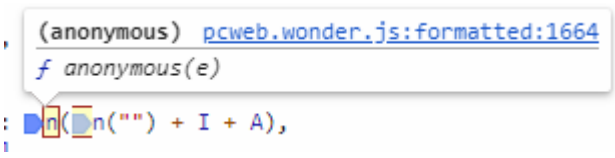
找到 authKey，并给他做断点。当然如果你能够顾及的话，你可以同时分析多个参数。

然后你要刷新页面（可以直接按 F5 刷新），让他停在断点处。

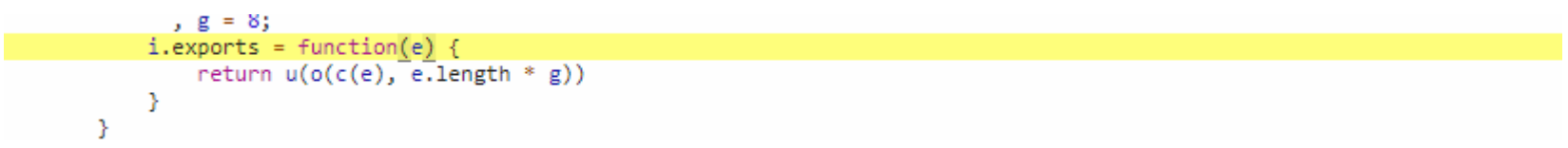


```
5408                var o, l = e.params, w = !!e.hostUseIP, T = null, k = e.params.tryCount || 0;
5409                return d.getVipTypes(function(e) {
5410                    var A = l.tvid
5411                      , S = l.vid
5412                      , E = l.vd
5413                      , P = l.ppt
5414                      , L = l.ost
5415                      , I = x > 0 ? x : (new Date).getTime();
5416                    x = 0;
5417                    _.sTime_vms = new Date - _.pgct,
5418                    b.sendVrsRequestPingback(),
5419                    T = function() {
5420                        var e = {
5421                            tvid: A,
5422                            bid: E,
5423                            vid: S,
5424                            src: c.isTWLocale() ? a.getPtid(!0) : f.win,
5425                            vt: 0,
5426                            rs: 1,
5427                            uid: d.getUid(),
5428                            ori: "pcw",
5429                            ps: l.switchvd ? 1 : 0,
5430                            tm: I,
5431                            qd_v: "1",
5432                            k_uid: p.getFluid() || p.getJsuid(),
5433                            pt: 0,
5434                            d: 0,
5435                            s: "",
5436                            lid: "",
5437                            cf: "",
5438                            ct: "",
5439                            authKey: n(n("") + I + A),
5440                            k_tag: 1,
5441                            ost: L,
5442                            ppt: P,
5443                            dfp: g.get(),
5444                            locale: c.isTraditionalChinese() ? "zh_tw" : "zh_cn",
5445                            prio: JSON.stringify({
5446                                ff: "f4v",
```

刷新后停在 e 处，你也可以直接让指针触碰函数 n，然后你就能看到



你可以直接点进去看是哪一个函数，显示是这个函数



```
                , g = 8;
            i.exports = function(e) {
                return u(o(c(e), e.length * g))
            }
        }
```

看来这个 n 是调用了一串这些函数返回的结果。

u(o(c(e), e.length * g))

继续往前面找函数 c()，因为停留在这个，再了解作用域这些东西可以知道，这个函数 c 就是在这里面，下图

```
            return a(i ^ (t | ~o), e, t, n, r, s)
        }
        function p(e, t) {
            var i = (65535 & e) + (65535 & t);
            return (e >> 16) + (t >> 16) + (i >> 16) << 16 | 65535 & i
        }
        function l(e, t) {
            return e << t | e >>> 32 - t
        }
        function c(e) {
            for (var t = Array(), i = (1 << g) - 1, o = 0; o < e.length * g; o += g)
                t[o >> 5] |= (e.charCodeAt(o / g) & i) << o % 32;
            return t
        }
        function u(e) {
            for (var t = f ? "0123456789ABCDEF" : "0123456789abcdef", i = "", o = 0; o < 4 * e.length; o++)
                i += t.charAt(e[o >> 2] >> o % 4 * 8 + 4 & 15) + t.charAt(e[o >> 2] >> o % 4 * 8 & 15);
            return i
        }
        var f = 0
          , g = 8;
        i.exports = function(e) {
            return u(o(c(e), e.length * g))
        }
    }
    .call(t, i, t, e)) && (e.exports = o)
}
, function(e, t, i) {
```

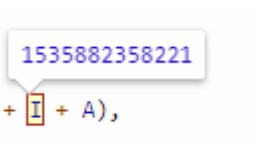
看来函数 c 是最底层了（也就是说没有再往下调用的函数了（自定义））

下面到函数 o，再往前看就能找到函数 o 了。

```
}
, function(e, t, i) {
    var o;
    void 0 !== (o = function(e, t, i) {
        function o(e, t) {
            e[t >> 5] |= 128 << t % 32,
            e[14 + (t + 64 >>> 9 << 4)] = t;
            for (var i = 1732584193, o = -271733879, a = -1732584194, l = 271733878, c = 0; c < e.length; c += 16) {
                var u = i
                  , f = o
                  , g = a
                  , _ = l;
                i = n(i, o, a, l, e[c + 0], 7, -680876936),
                l = n(l, i, o, a, e[c + 1], 12, -389564586),
                a = n(a, l, i, o, e[c + 2], 17, 606105819),
                o = n(o, a, l, i, e[c + 3], 22, -1044525330),
                i = n(i, o, a, l, e[c + 4], 7, -176418897),
                l = n(l, i, o, a, e[c + 5], 12, 1200080426),
                a = n(a, l, i, o, e[c + 6], 17, -1473231341),
                o = n(o, a, l, i, e[c + 7], 22, -45705983),
                i = n(i, o, a, l, e[c + 8], 7, 1770035416),
                l = n(l, i, o, a, e[c + 9], 12, -1958414417),
                a = n(a, l, i, o, e[c + 10], 17, -42063),
                o = n(o, a, l, i, e[c + 11], 22, -1990404162),
                i = n(i, o, a, l, e[c + 12], 7, 1804603682),
                l = n(l, i, o, a, e[c + 13], 12, -40341101),
                a = n(a, l, i, o, e[c + 14], 17, -1502002290),
                o = n(o, a, l, i, e[c + 15], 22, 1236535329),
                i = r(i, o, a, l, e[c + 1], 5, -165796510),
                l = r(l, i, o, a, e[c + 6], 9, -1069501632),
                a = r(a, l, i, o, e[c + 11], 14, 643717713),
                o = r(o, a, l, i, e[c + 0], 20, -373897302),
                i = r(i, o, a, l, e[c + 5], 5, -701558691),
                l = r(l, i, o, a, e[c + 10], 9, 38016083),
                a = r(a, l, i, o, e[c + 15], 14, -660478335),
                o = r(o, a, l, i, e[c + 4], 20, -405537848),
                i = r(i, o, a, l, e[c + 9], 5, 568446438),
                l = r(l, i, o, a, e[c + 14], 9, -1019803690),
                a = r(a, l, i, o, e[c + 3], 14, -187363961),
                o = r(o, a, l, i, e[c + 8], 20, 1163531501),
                i = r(i, o, a, l, e[c + 13], 5, -1444681467),
                l = r(l, i, o, a, e[c + 2], 9, -51403784),
```

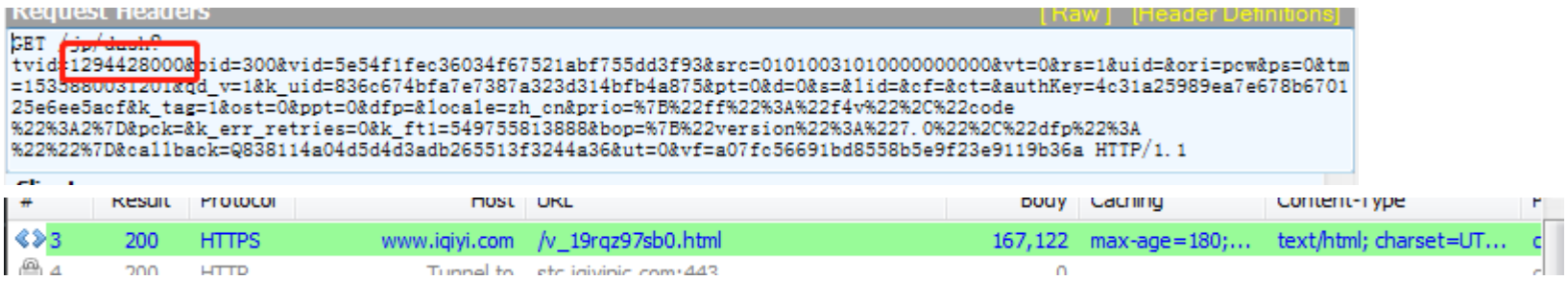
最后就是 u 了，在上上图就能看到函数 u 了，所以，这个参数 authKey 就可以构造了，现在要做的就是要找到调用这些参数所传递的参数。，就是 I 和 A

```
    b.sendVrsRequestPingback(),
    T = function() {
        var e = f, e = undefined
            tvid: A,
            bid: E,
            vid: S,
            src: c.isTWLocale() ? a.getPtid(!0) : f.win,
            vt: 0,
            rs: 1,
            uid: d.getUid(),
            ori: "pcw",
            ps: l.switchvd ? 1 : 0,
            tm: I,
            qd_v: "1",
            k_uid: p.getFluid() || p.getJsuid(),
            pt: 0,
            d: 0,
            s: "",
            lid: "",
            cf: "",
            ct: "",
            authKey: n(n("") + I + A),
            k_tag: 1,
            ost: L,
            ppt: P,
            dfp: g.get(),
            locale: c.isTraditionalChinese() ? "zh_tw" : "zh_cn",
            prio: JSON.stringify({
                ff: "f4v",
                code: 2
            }),
            pck: d.passportCookie(),
            k_err_retries: k
    };
    return e.k_ft1 = v.getFT1(),  e = undefined
```

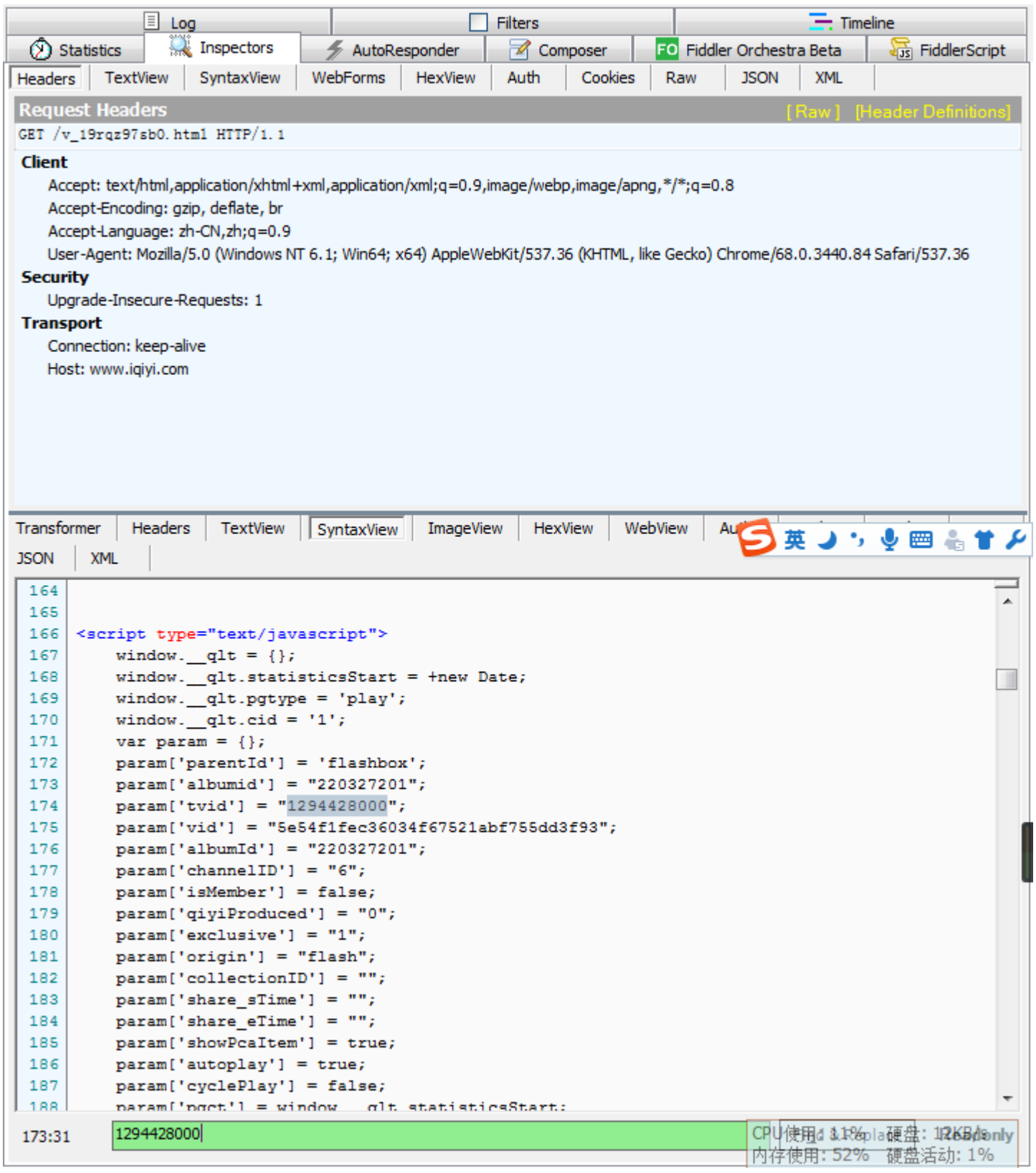


看参数名字 tm 大概可以知道这是一个时间戳，而且查看 I 也发现 ，所以这很大概率就是时间戳。

而 A 就是参数 tvid 的值。我们先返回去 fiddler 看下这个 tvid 在前面的链接有没有提到过。



发现第一个链接就有这串数字。



所以我们就可以从这个链接得到 A 了，这个链接就是爱奇艺视频的请求地址。

下一步

&k_uid=836c674bfa7e7387a323d314bfb4a875

在进行这一步之前不妨先看下这还缺什么

```
var e = {
    tvid: A,
    bid: E,
    vid: S,
    src: c.isTWLocale() ? a.getPtid(!0) : f.win,
    vt: 0,
    rs: 1,
    uid: d.getUid(),
    ori: "pcw",
    ps: l.switchvd ? 1 : 0,
    tm: I,
    qd_v: "1",
    k_uid: p.getFluid() || p.getJsuid(),
    pt: 0,
    d: 0,
    s: "",
    lid: "",
    cf: "",
    ct: "",
    authKey: n(n("") + I + A),
    k_tag: 1,
    ost: L,
    ppt: P,
    dfp: g.get(),
    locale: c.isTraditionalChinese() ? "zh_tw" : "zh_cn",
    prio: JSON.stringify({
        ff: "f4v",
        code: 2
    }),
    pck: d.passportCookie(),
    k_err_retries: k
};
```

通过对比发现

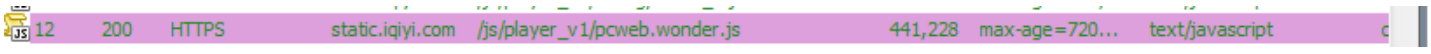
```
tvid=1294428000
bid=300
vid=5e54f1fec36034f67521abf755dd3f93
src=01010031010000000000
vt=0
rs=1
uid=
ori=pcw
ps=0
tm=1535880031201
qd_v=1
k_uid=836c674bfa7e7387a323d314bfb4a875
pt=0
d=0
s=
lid=
cf=
ct=
authKey=4c31a25989ea7e678b670125e6ee5acf
k_tag=1
ost=0
ppt=0
dfp=
locale=zh_cn
prio=%7B%22ff%22%3A%22f4v%22%2C%22code%22%3A2%7D
pck=
k_err_retries=0
k_ft1=549755813888
bop=%7B%22version%22%3A%227.0%22%2C%22dfp%22%3A%22%22%7D
callback=Q838114a04d5d4d3adb265513f3244a36
ut=0
vf=a07fc56691bd8558b5e9f23e9119b36a
```

就下面蓝点那五个参数是上面没有的。所以前面哪一些我就不讲了（因为可以通过一样的道理分析 js 就能得到参数的依据），接下来我要讲的就是获得以下五个参数。

k_ft1=549755813888

首先是这个，要做的当然就是先找一下后面的那一串数字在 fiddler 在更前一点的链接里面有没有找到辣。

可惜的是没有找到。。那么我们只能去寻找他的参数名了“k_ft1”



```
                                cf: "",
                                ct: "",
                                authKey: n(n("") + I + A),
                                k_tag: 1,
                                ost: L,
                                ppt: P,
                                dfp: g.get(),
                                locale: c.isTraditionalChinese()
                                prio: JSON.stringify({
                                    ff: "f4v",
                                    code: 2
                                }), pck: d.passportCookie(),
                                k_err_retries: k
                            };
                            return e.k_ft1 = v.getFT1(), c.isTraditionalChinese() && (e.
locale = "zh_tw"), h.isBoss() && (e.vv = "821d3c731e374feaa629dcdaab7c394b"), w && (e.
ecode = "701"), e.bop = JSON.stringify({
                                version: "7.0",
                                dfp: g.get()
                            }), e
                        }();
                        var F = function(a) {
                            o = s.jsonp({
                                url: a,
                                params: T,
                                memory: !0,
```

```
                    d: 0,
                    s: "",
                    lid: "",
                    cf: "",
                    ct: "",
                    authKey: n(n("") + I + A),
                    k_tag: 1,
                    ost: L,
                    ppt: P,
                    dfp: g.get(),
                    locale: c.isTraditionalChinese() ? "zh_tw" : "zh_cn",
                    prio: JSON.stringify({
                        ff: "f4v",
                        code: 2
                    }), pck: d.passportCookie(),
                    k_err_retries: k
                };
                return e.k_ft1 = v.getFT1(), c.isTraditionalChinese() && (e.locale = "zh_tw"), h.isBoss() && (e.vv = "821d3c731e374feaa629dcdaab7c394b"), w && (e.ecode
                 = "701"), e.bop = JSON.stringify({
                    version: "7.0",
                    dfp: g.get()
                }), e
            }();
            var F = function(a) {
                o = s.jsonp({
                    url: a,
                    params: T,
                    memory: !0,
                    timeout: 5e3,
                    beforeSend: function(t) {
                        var i = r.parse(t.url).host;
                        e.forEach(function(e) {
                            t.url += "&ut=" + e
                        });
                        var o = "iloveiqiyi";
                        try {
                            var a = t.url.replace(new RegExp("^(http|https)://" + i, "ig"), "");
```

事实上你不一定看到这些就马上用谷歌浏览器的开发者工具取调试他，你可以在 notepad++里面看一下，找一下这个函数，比如这里面这些函数都是没有经过压缩函数名，这样可以直接取寻找这个函数，然后大概分析一下他的含义，也就是他是干什么的。

```
Search "getFT1" (2 hits in 1 file)
  new 5 (2 hits)
    Line 4413:                 return e.k_ft1 = v.getFT1(), c.isTraditionalChinese() && (e.locale = "zh_tw"), h.isBoss() && (e.vv = "821d3c731e374feaa629dcdaab7c394b"), w && (e.e
    Line 7942:             getFT1: o,
Search "k_ft1" (1 hit in 1 file)
Search "authKey" (1 hit in 1 file)
```

```
                2: {}
            };
            i.exports = {
                getFT1: o,
                getFT2: a,
                hasBit: s,
                openBit: n,
                closeBit: r
            }
        }
        .call(t, i, t, e)) && (e.exports = o)
    },
```

```
},
function(e, t, i) {
    var o;
    void 0 !== (o = function(e, t, i) {
        function o() {
            for (var e = [], t = 1; t <= 64; t++)
                e.push(d[1][t] ? 1 : 0);
            return parseInt(e.reverse().join(""), 2)
        }
        function a() {}
        function n(e, t) {
            d[e][t] = !0
        }
        function r(e, t) {
            d[e][t] = !1
        }
        function s(e, t) {
            return d[e][t]
        }
        var d = {
            1: {
                3: !1,
                37: !1,
                40: !0,
                42: !1
            },
            2: {}
        };
        i.exports = {
            getFT1: o,
            getFT2: a,
            hasBit: s,
            openBit: n,
            closeBit: r
        }
    }
```

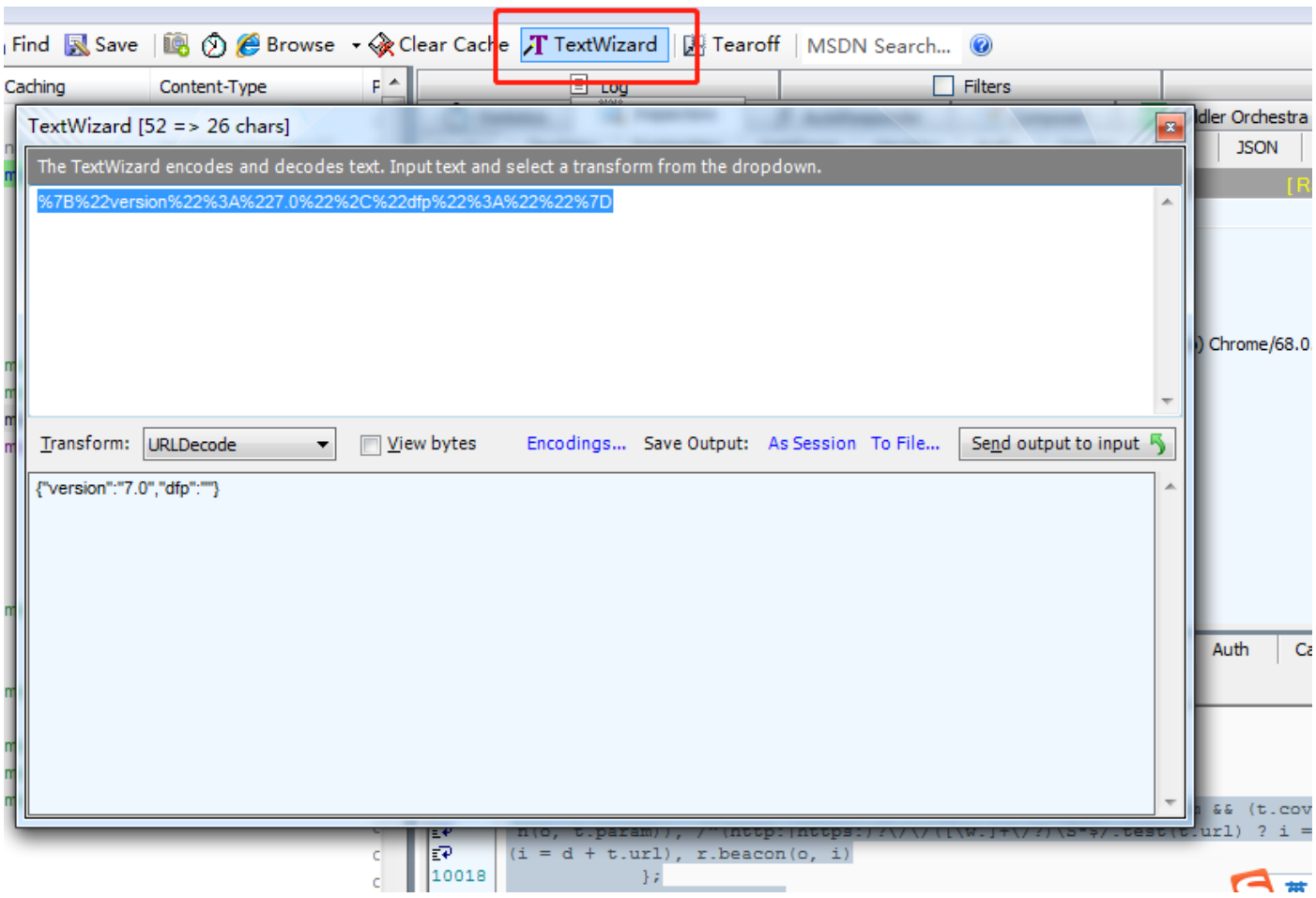
所以这个参数解决掉了。

第二个参数：

bop=%7B%22version%22%3A%227.0%22%2C%22dfp%22%3A%22%22%7D

同样的方法，先搜索参数值，但是注意的是这里面的参数值是通过 urlencode 的，所以先解码再寻找。

可以使用 fiddler 里面的 textwizard 工具来解码



发现并没有这个，但是你可以搜索里面的 version 或者 dfp，为什么呢？因为这很有可能是类似的 js 片段。

```
{
  'version' : e.version,
  'dfp' : e.dfp
}
```

，我建议搜索 dfp。但是说道理，我们也可以直接搜索参数名 bop 而不纠结与参数值。

然后有意思的是，我们想太多了，却发现其实都在这里。

```
4381                    _.sTime_vms = new Date - _.pgct, b.sendVrsRequestPingback(), T = function() {
4382                        var e = {
4383                            tvid: A,
4384                            bid: E,
4385                            vid: S,
4386                            src: c.isTWLocale() ? a.getPtid(!0) : f.win,
4387                            vt: 0,
4388                            rs: 1,
4389                            uid: d.getUid(),
4390                            ori: "pcw",
4391                            ps: l.switchvd ? 1 : 0,
4392                            tm: I,
4393                            qd_v: "1",
4394                            k_uid: p.getFluid() || p.getJsuid(),
4395                            pt: 0,
4396                            d: 0,
4397                            s: "",
4398                            lid: "",
4399                            cf: "",
4400                            ct: "",
4401                            authKey: n(n("") + I + A),
4402                            k_tag: 1,
4403                            ost: L,
4404                            ppt: P,
4405                            dfp: g.get(),
4406                            locale: c.isTraditionalChinese() ? "zh_tw" : "zh_cn",
4407                            prio: JSON.stringify({
4408                                ff: "f4v",
4409                                code: 2
4410                            }), pck: d.passportCookie(),
4411                            k_err_retries: k
4412                        };
4413                        return e.k_ft1 = v.getFT1(), c.isTraditionalChinese() && (e.locale = "zh_tw"), h.isBoss() && (e.vv = "821d3c731e374feaa629dcdaab7c394b"), w && (e.ecode
                         = "701"), e.bop = JSON.stringify({
4414                            version: "7.0",
4415                            dfp: g.get()
4416                        }), e
```

所以这就完了，事实上这里面就有几个函数了。

那么剩下的 callback vf 可以通过

而 callback 大家可以使用谷歌浏览器的开发者工具调试一下，看一下什么时候获得了 callback。（图忘了截，所以就当做是练习好了），

下面是 vf

```
                    };
                    return e.k_ft1 = v.getFT1(), c.isTraditionalChinese() && (e.locale = "zh_tw"), h.isBoss() && (e.vv = "821d3c731e374feaa629dcda
                     = "701"), e.bop = JSON.stringify({
                        version: "7.0",
                        dfp: g.get()
                    }), e
                }();
                var F = function(a) {
                    o = s.jsonp({
                        url: a,
                        params: T,
                        memory: !0,
                        timeout: 5e3,
                        beforeSend: function(t) {
                            var i = r.parse(t.url).host;
                            e.forEach(function(e) {
                                t.url += "&ut=" + e
                            });
                            var o = "iloveiqiyi";
                            try {
                                var a = t.url.replace(new RegExp("^(http|https)://" + i, "ig"), "");
                                w && (a = a.replace("/3ea/420a8433732a6c99d1eae98fea69e55d", "")), o = u.cmd5x(a)
                            }
                            catch (e) {
                                y.error("cmd5x: " + (e.message ? e.message : e))
                            }
                            return t.url += "&vf=" + o, y.log("load movieInfo from vrs, request,params: url = " + t.url), t
                        },
                        success: function(e) {
                            if (y.log("dash success raw json data->" + JSON.stringify(e)), _.usedTime_vms = new Date - _.pgct - _.sTime_vms, b.sen
                            usedTime_vms), e && e.hasOwnProperty("code"))
                                "A00000" === e.code ? t(e) : ("A00020" === e.code && e.tm && (x = e.tm), i(e));
                            else {
                                var o = {};
                                o.code = "P00002", i(o)
                            }
```

然后大家可以吧刚才那些参数整理一下就行了。

这是一个漫长的工作，所以，我就不帮你们整理了。

到现在，我们已经构造出来了

GET /jp/dash?tvid=1294428000&bid=300&vid=5e54f1fec36034f67521abf755dd3f93&src=01010031010000000000&vt=0&rs=1&uid=&ori=pcw&ps=0&tm=1535880031201&qd_v=1&k_uid=836c674bfa7e7387a323d314bfb4a875&pt=0&d=0&s=&lid=&cf=&ct=&authKey=4c31a25989ea7e678b670125e6ee5acf&k_tag=1&ost=0&ppt=0&dfp=&locale=zh_cn&prio=%7B%22ff%22%3A%22f4v%22%2C%22code%22%3A2%7D&pck=&k_err_retries=0&k_ft1=549755813888&bop=%7B%22version%22%3A%227.0%22%2C%22dfp%22%3A%22%22%7D&callback=Q838114a04d5d4d3adb265513f3244a36&ut=0&vf=a07fc56691bd8558b5e9f23e9119b36a HTTP/1.1

这样，我们就能得到

一开始寻找的字符串：

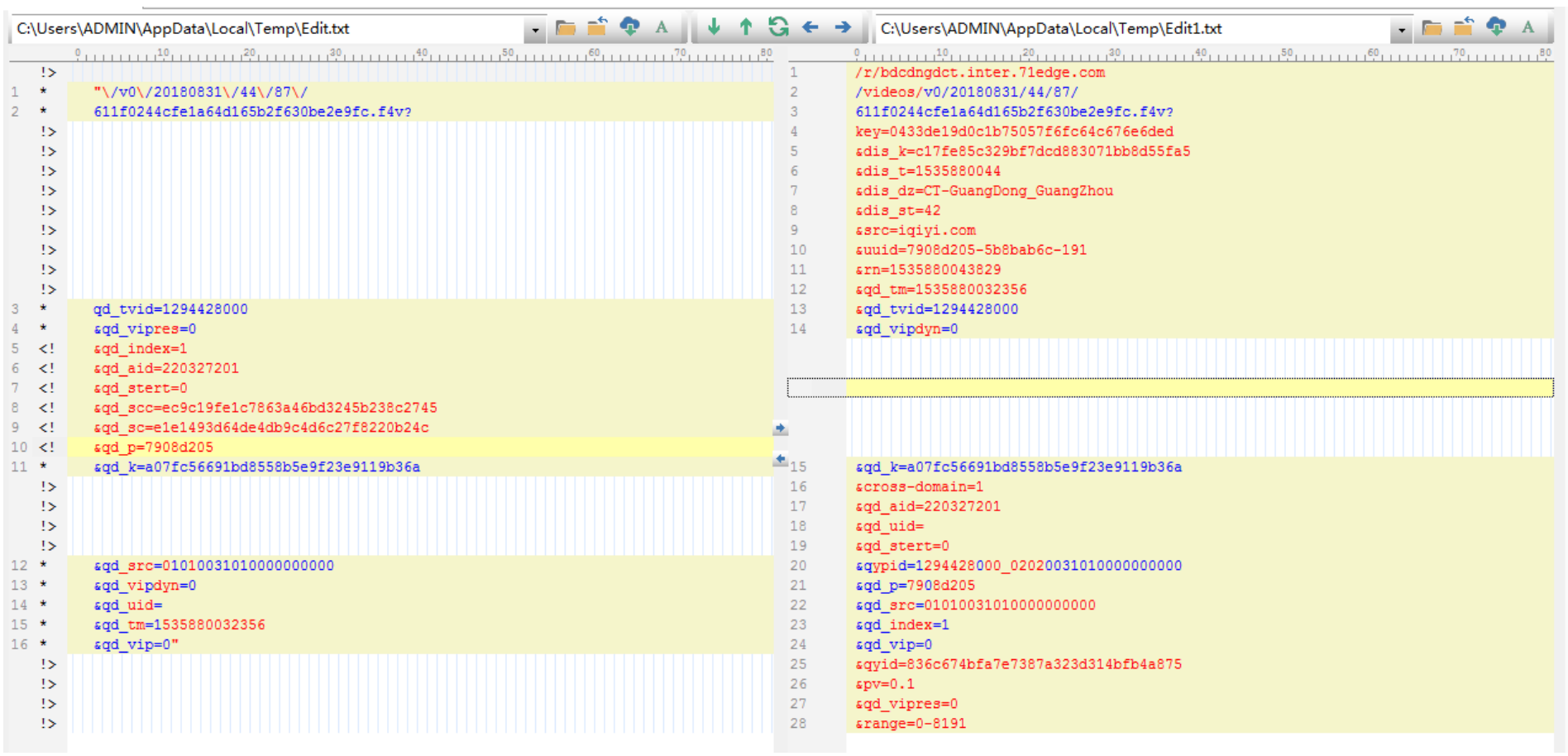
611f0244cfe1a64d165b2f630be2e9fc

```
"p2p": "",
"bid": 300,
"_selected": true,
"ps": 0,
"fs": [
    {
        "b": 40362120,
        "s": 0,
        "d": 370345,
        "l":
        "\/v0\/20180831\/44\/87\/611f0244cfe1a64d165b2f630be2e9fc.f4v?qd_tvid=1294428000&qd_vipres=0&qd_index=1&qd_aid=220327201&qd_stert=0&qd_scc=ec9c19fe1c
        7863a46bd3245b238c2745&qd_sc=e1e1493d64de4db9c4d6c27f8220b24c&qd_p=7908d205&qd_k=a07fc56691bd8558b5e9f23e9119b36a&qd_src=01010031010000000000&qd_vipd
        yn=0&qd_uid=&qd_tm=1535880032356&qd_vip=0"
    },
    {
```


而要知道的是，我们的目标就是下面这个链接，上面的链接只是为了寻找字符串　　611f0244cfe1a64d165b2f630be2e9fc

GET /r/bdcdngdct.inter.71edge.com/videos/v0/20180831/44/87/611f0244cfe1a64d165b2f630be2e9fc.f4v?key=0433de19d0c1b75057f6fc64c676e6ded&dis_k=c17fe85c329bf7dcd883071bb8d55fa5&dis_t=1535880044&dis_dz=CT-GuangDong_GuangZhou&dis_st=42&src=iqiyi.com&uuid=7908d205-5b8bab6c-191&rn=1535880043829&qd_tm=1535880032356&qd_tvid=1294428000&qd_vipdyn=0&qd_k=a07fc56691bd8558b5e9f23e9119b36a&cross-domain=1&qd_aid=220327201&qd_uid=&qd_stert=0&qypid=1294428000_02020031010000000000&qd_p=7908d205&qd_src=01010031010000000000&qd_index=1&qd_vip=0&qyid=836c674bfa7e7387a323d314bfb4a875&pv=0.1&qd_vipres=0&range=0-8191 HTTP/1.1

通过简单观察发现，那个链接不仅得到了字符串 611f0244cfe1a64d165b2f630be2e9fc，还得到了很多参数。不如比较一下看还缺什么。

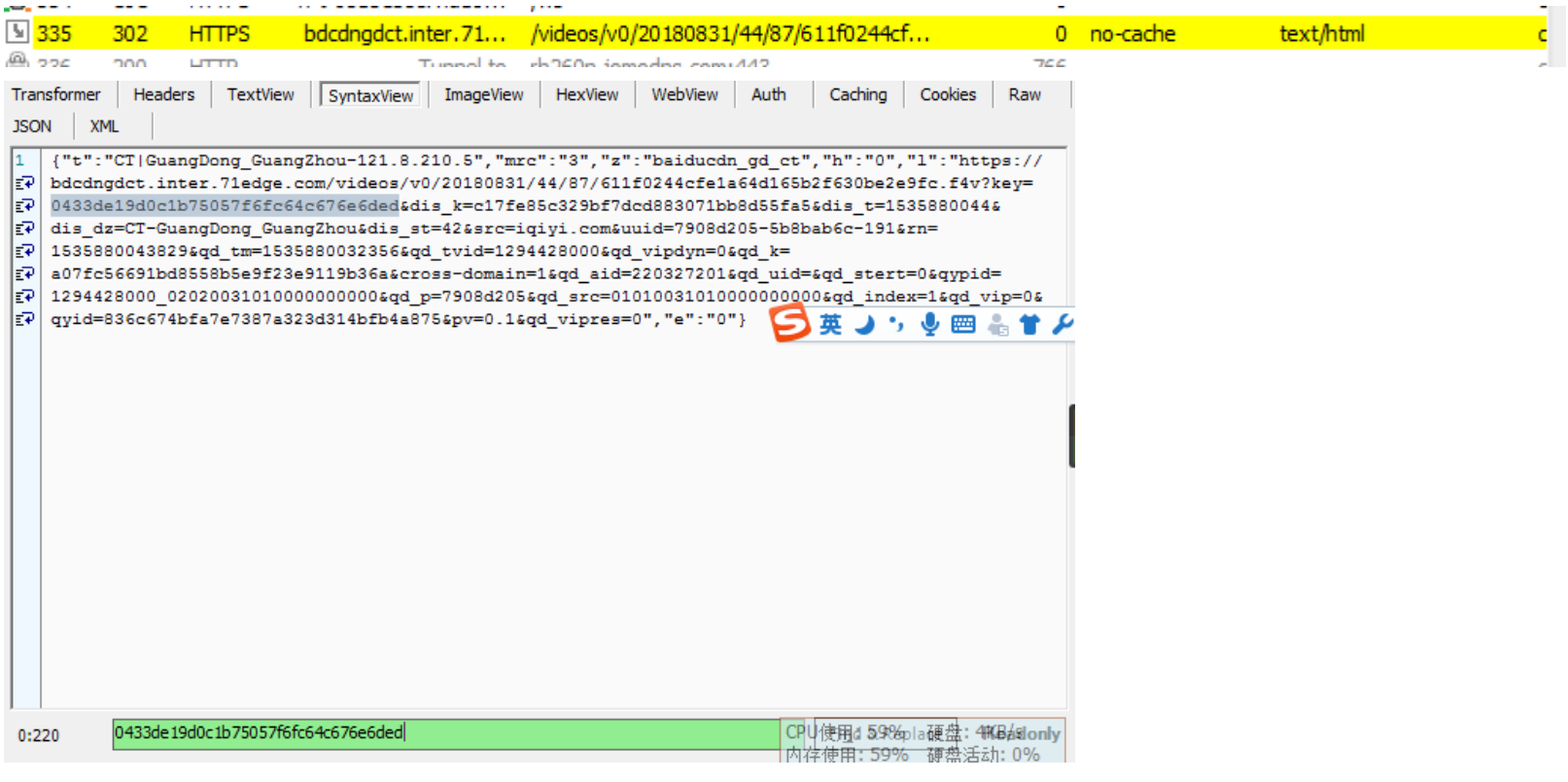


通过对比发现还确实缺了不少。

不如先从 key 开始找

key=0433de19d0c1b75057f6fc64c676e6ded

寻找 0433de19d0c1b75057f6fc64c676e6ded，得到下面这一项



发现里面的链接就是很完整的好不好。

https://bdcdngdct.inter.71edge.com/videos/v0/20180831/44/87/611f0244cfe1a64d165b2f630be2e9fc.f4v?key=0433de19d0c1b75057f6fc64c676e6ded&dis_k=c17fe85c329bf7dcd883071bb8d55fa5&dis_t=1535880044&dis_dz=CT-GuangDong_GuangZhou&dis_st=42&src=iqiyi.com&uuid=7908d205-5b8bab6c-191&rn=1535880043829&qd_tm=1535880032356&qd_tvid=1294428000&qd_vipdyn=0&qd_k=a07fc56691bd8558b5e9f23e9119b36a&cross-domain=1&qd_aid=220327201&qd_uid=&qd_stert=0&qypid=1294428000_02020031010000000000&qd_p=7908d205&qd_src=01010031010000000000&qd_index=1&qd_vip=0&qyid=836c674bfa7e7387a323d314bfb4a875&pv=0.1&qd_vipres=0

所以我们要的就是要构造下面

GET /videos/v0/20180831/44/87/611f0244cfe1a64d165b2f630be2e9fc.f4v?qd_tvid=1294428000&qd_vipres=0&qd_index=1&qd_aid=220327201&qd_stert=0&qd_scc=ec9c19fe1c7863a46bd3245b238c2745&qd_sc=e1e1493d64de4db9c4d6c27f8220b24c&qd_p=7908d205&qd_k=a07fc56691bd8558b5e9f23e9119b36a&qd_src=01010031010000000000&qd_vipdyn=0&qd_uid=&qd_tm=1535880032356&qd_vip=0&cross-domain=1&qyid=836c674bfa7e7387a323d314bfb4a875&qypid=1294428000_02020031010000000000&qypid=1294428000_02020031010000000000&rn=1535880043829&pv=0.1&cross-domain=1 HTTP/1.1

分开参数

/videos/v0/20180831/44/87/
611f0244cfe1a64d165b2f630be2e9fc.f4v?
qd_tvid=1294428000
&qd_vipres=0
&qd_index=1
&qd_aid=220327201
&qd_stert=0
&qd_scc=ec9c19fe1c7863a46bd3245b238c2745
&qd_sc=e1e1493d64de4db9c4d6c27f8220b24c
&qd_p=7908d205
&qd_k=a07fc56691bd8558b5e9f23e9119b36a
&qd_src=01010031010000000000
&qd_vipdyn=0
&qd_uid=
&qd_tm=1535880032356
&qd_vip=0
&cross-domain=1
&qyid=836c674bfa7e7387a323d314bfb4a875
&qypid=1294428000_02020031010000000000
&qypid=1294428000_02020031010000000000
&rn=1535880043829
&pv=0.1
&cross-domain=1

这一次发现和之前的参数就就差不多了，比较一下

看来就剩下后面的几个参数了。

当然你也可以一个参数一个参数的找依据，但是有时候这是不必要的，你可以通过在不同的爱奇艺视频网页请求抓包，把一些不变的量你就当然是常量就行了。当然这也有弊端

比如说这样你就无法了解更深一层的含义。或者这些参数有可能就是破解 vip 限制的关键。（讲道理这些参数是什么意义我就没研究，所以这个破解 vip 我是随便说的，就是为了表达这样一个意思，就是这些参数对仅仅解析这些视频可能是无关重要的，但是如果你不仅仅只要这些，这些就很重要了）。

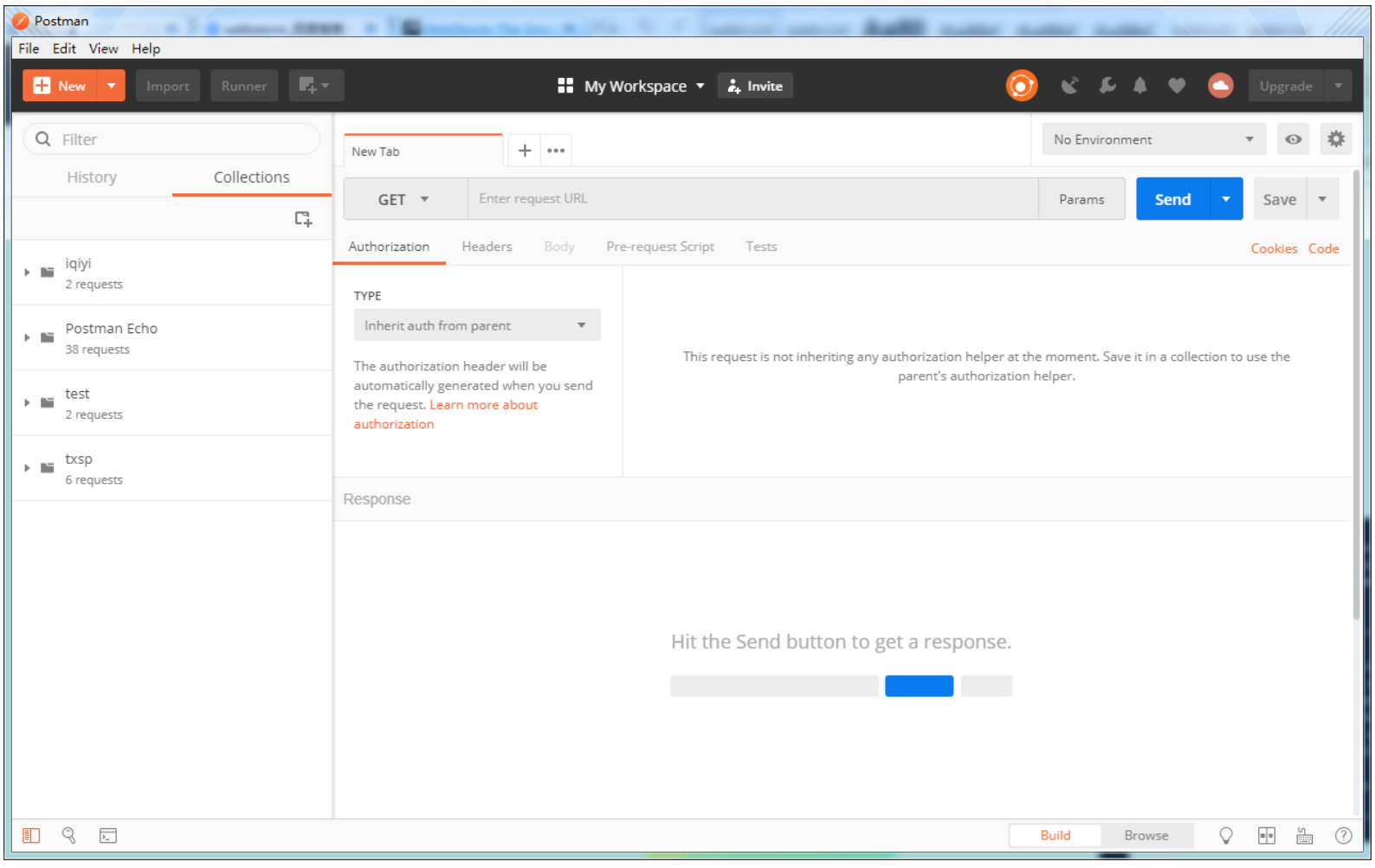
其实后面的几个参数我不需要讲，因为这个前面的一些参数值有交叉。比如&qyid=836c674bfa7e7387a323d314bfb4a875 就是 cookie 的 QC005。

所以后面就大概讲一下如何验证你构造出来的结果对不对。

后面就是要搬出 webstorm 了，

这个软件可以帮助你运行 js 脚本，这样你就能运行上面的函数得到结果。

然后你可以使用 postman 可以帮助你提交数据。



至于这些软件如何使用我就不讲了。（自己可以去网上查教程资料什么的）

以上。

以上的分析，我使用 python 实现了这一过程，下面 github 地址可以找到这一项目。

Github: https://github.com/ZSAIm/iqiyi-parser

有兴趣的小伙伴可以参与进来改善这个软件，还有我 Github 里面的其他项目。（Github 右上角的星号☆）！！