第四届“安洵杯”网络安全挑战赛MISC-Writeup

原创

末 初 于 2021-11-29 22:54:57 发布 4504 收藏 4

分类专栏： CTF_MISC_Writeup 文章标签： 2021安洵杯MISC

文章目录

```
题目附件请自取
链接：https://pan.baidu.com/s/13TwadE6DenseIuRUNZlCKg
提取码：rrpe
```

应该算是签到

B站搜索直接搜索这个BV号

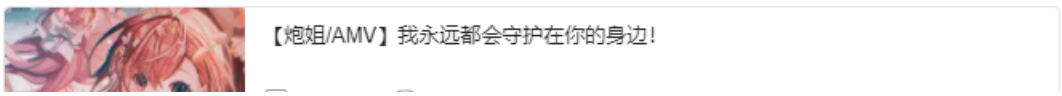
直接页面`Ctrl+F`没找出来
搜索引擎找一下有没有通过API查弹幕的方法：https://www.bilibili.com/read/cv7923601

至此，你就可以获得视频对应的弹幕XML文件了。

弹幕文件获取（历史弹幕）（新方法）[当前可用]

前置条件：通过上述方式获取CID，已经登录B站

注意：此方式可能会出现部分无法解析的数据（例如时间、颜色等）



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

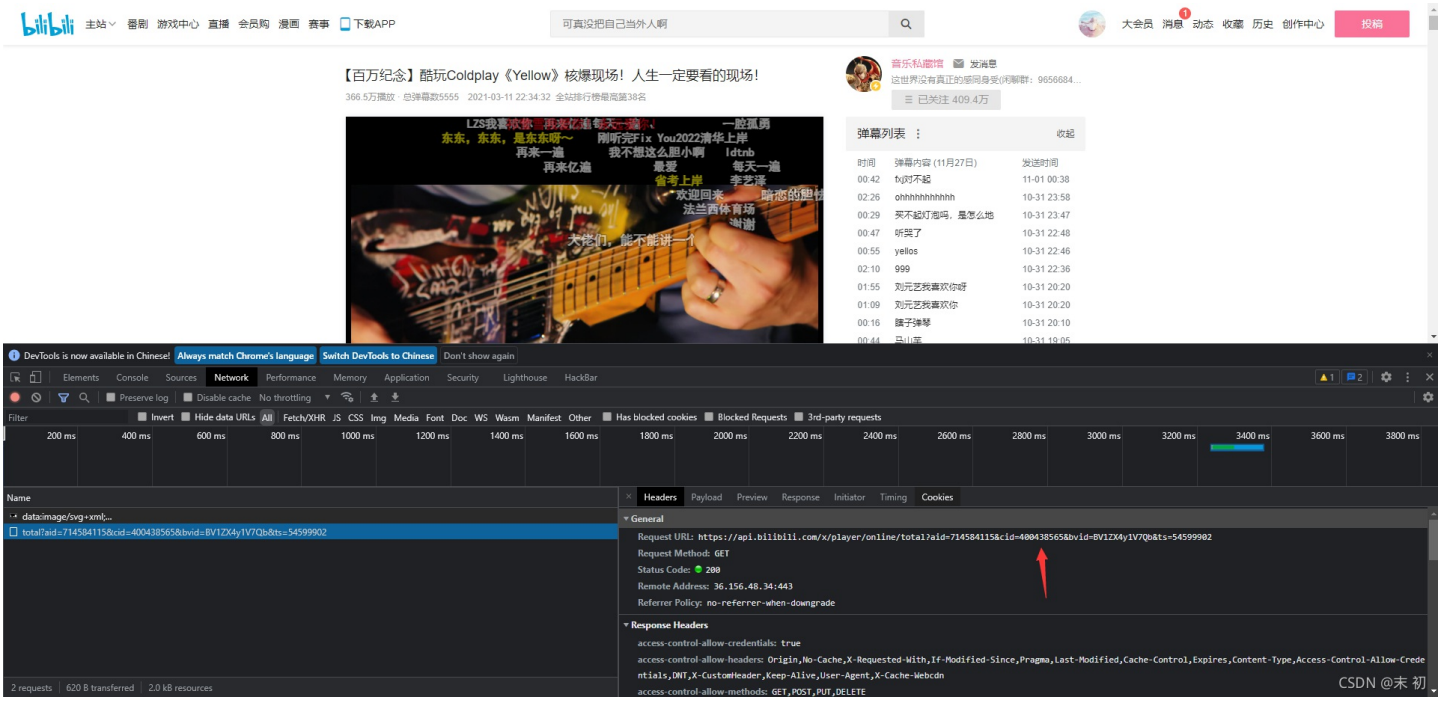
```
https://api.bilibili.com/x/v2/dm/web/history/seg.so?type=1&oid={cid}&date={date}
```

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~

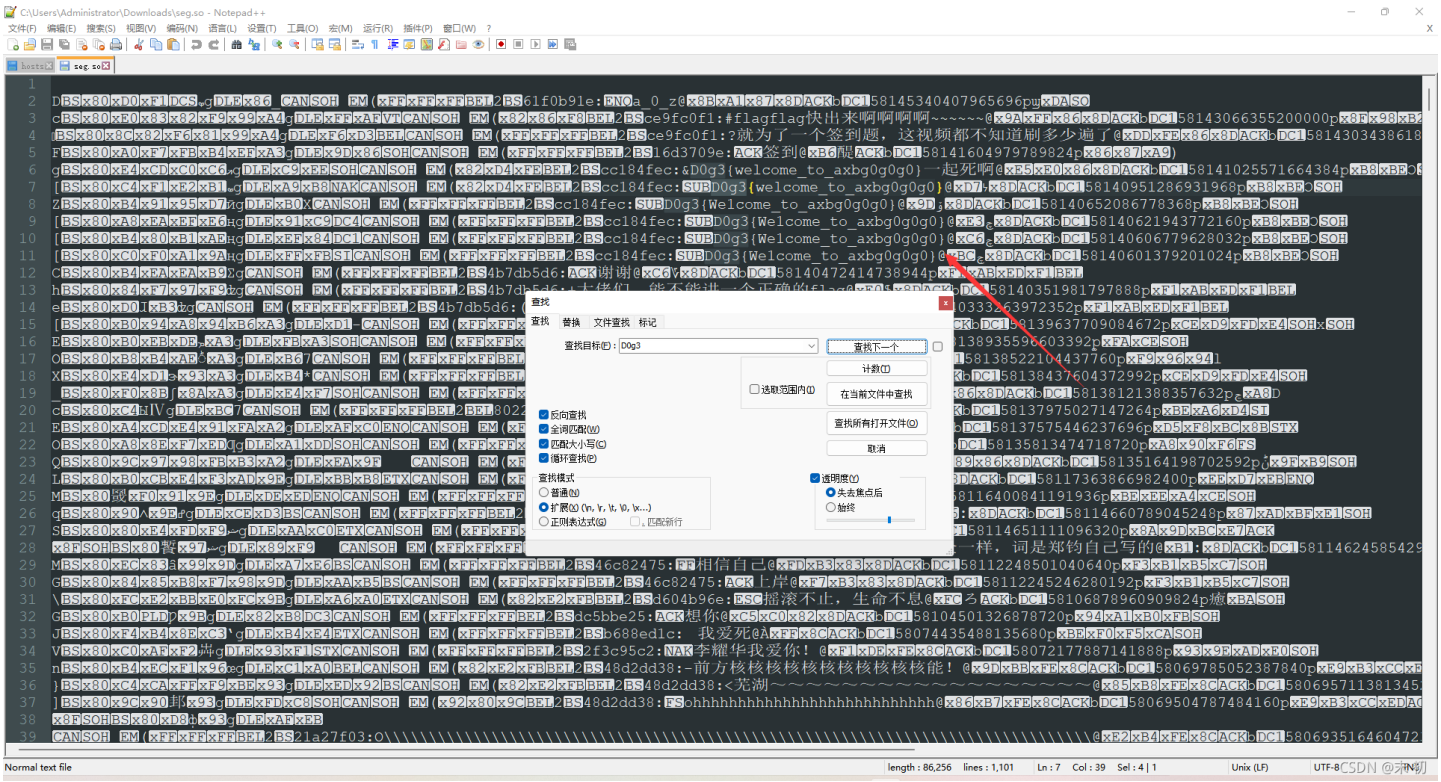
F12点击`Network`，找到这个视频的cid



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



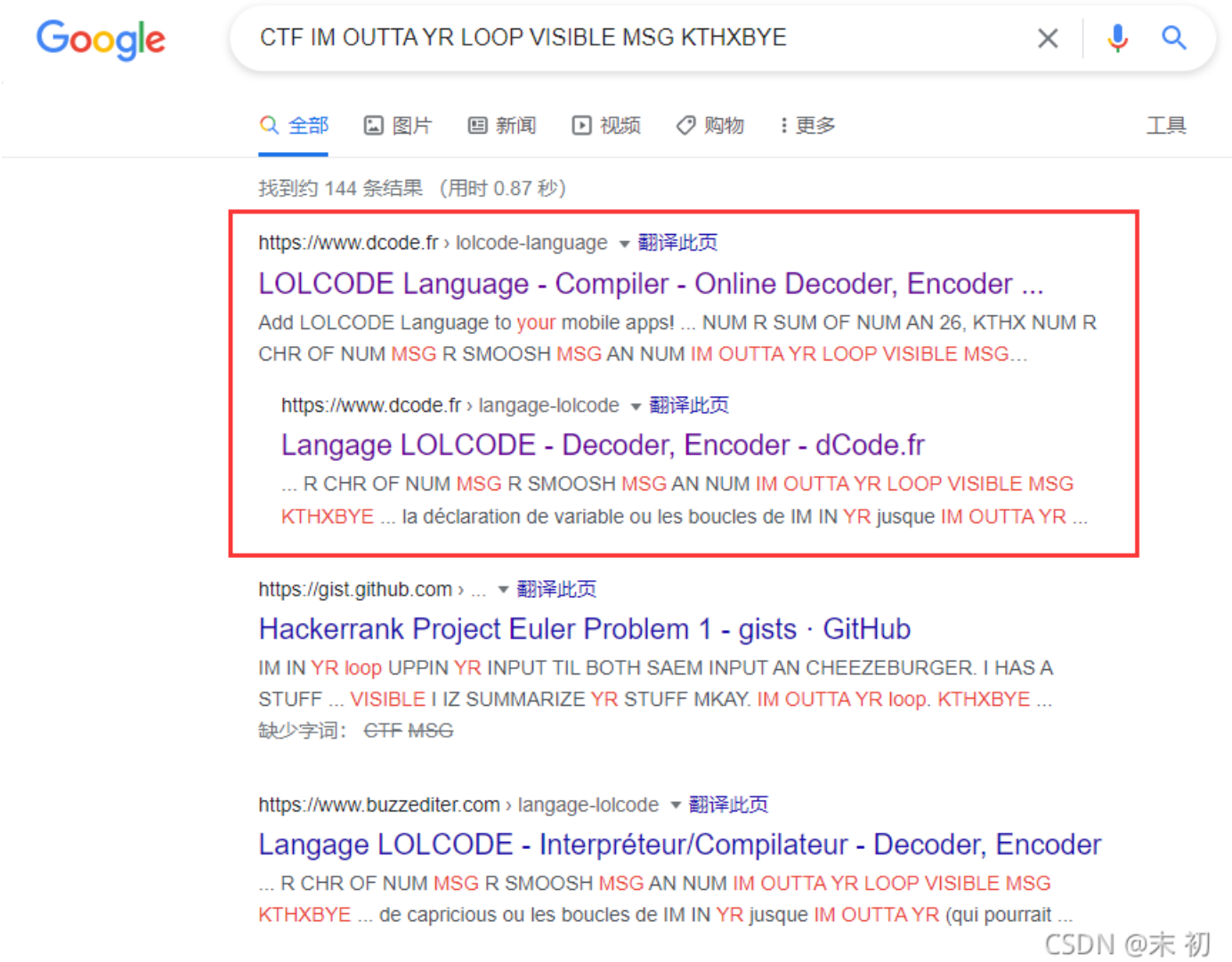
```
D0g3{We1come_to_axbg0g0g0}
```

CyzCC_loves_LOL


```
D0g3_LOLteampassword
```

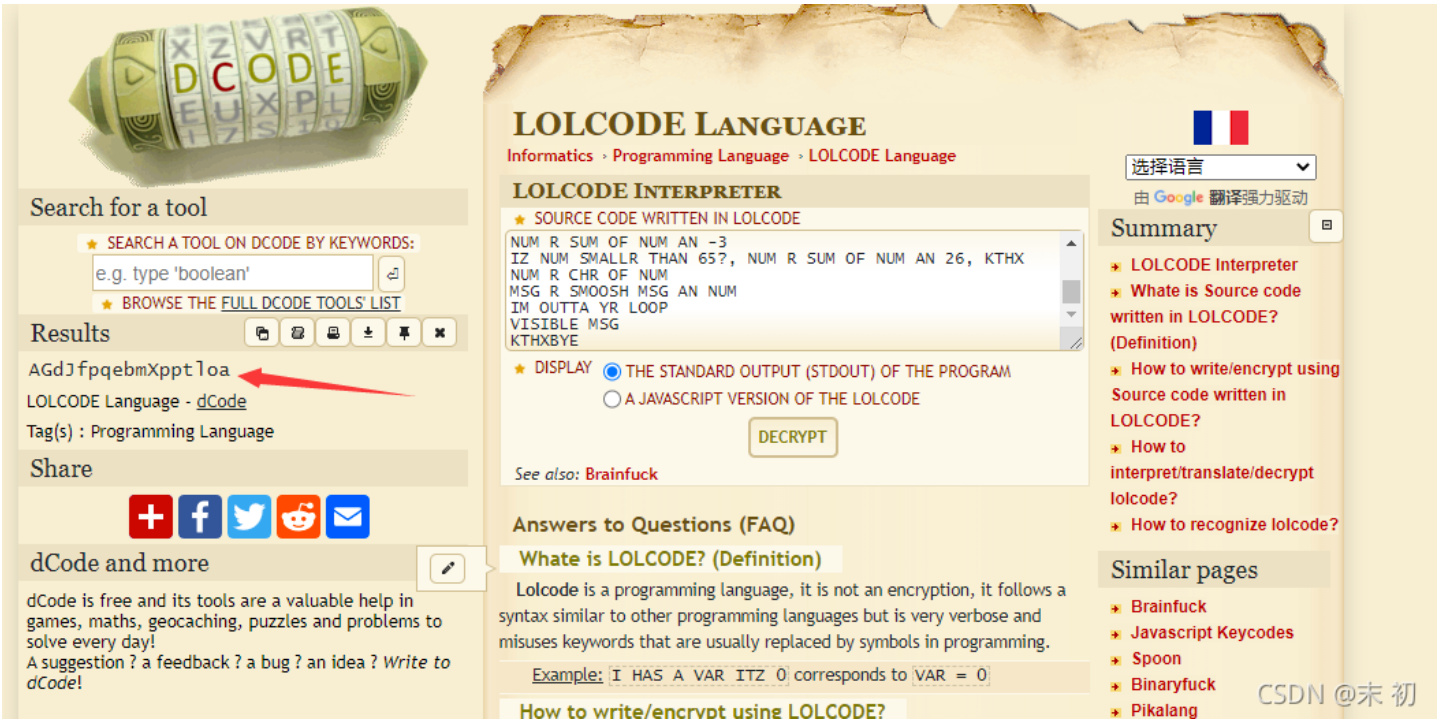
```
HAI D0g3 code
I HAS A CODE ITZ "D0g3isthepAssword"
I HAS A MSG ITZ ""
I HAS A COUNTER ITZ 0
I HAS A NUM
IM IN YR LOOP UPPIN YR COUNTER WILE COUNTER SMALLR THAN LEN OF CODE
I HAS A C ITZ CODE!COUNTER
NUM R ORD OF C
NUM R SUM OF NUM AN -3
IZ NUM SMALLR THAN 65?, NUM R SUM OF NUM AN 26, KTHX
NUM R CHR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

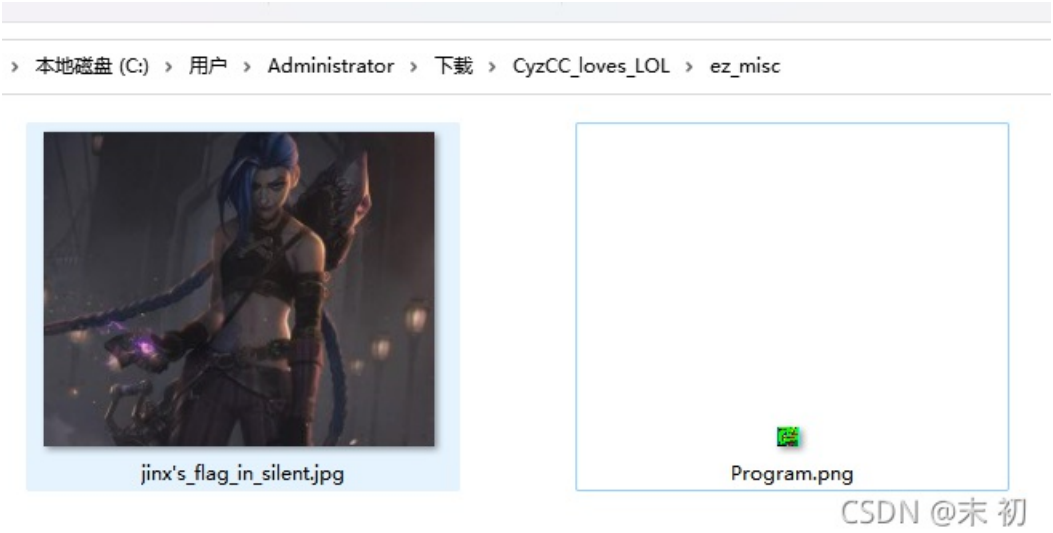
看不懂什么东西，猜测某种编码，搜索引擎找一下



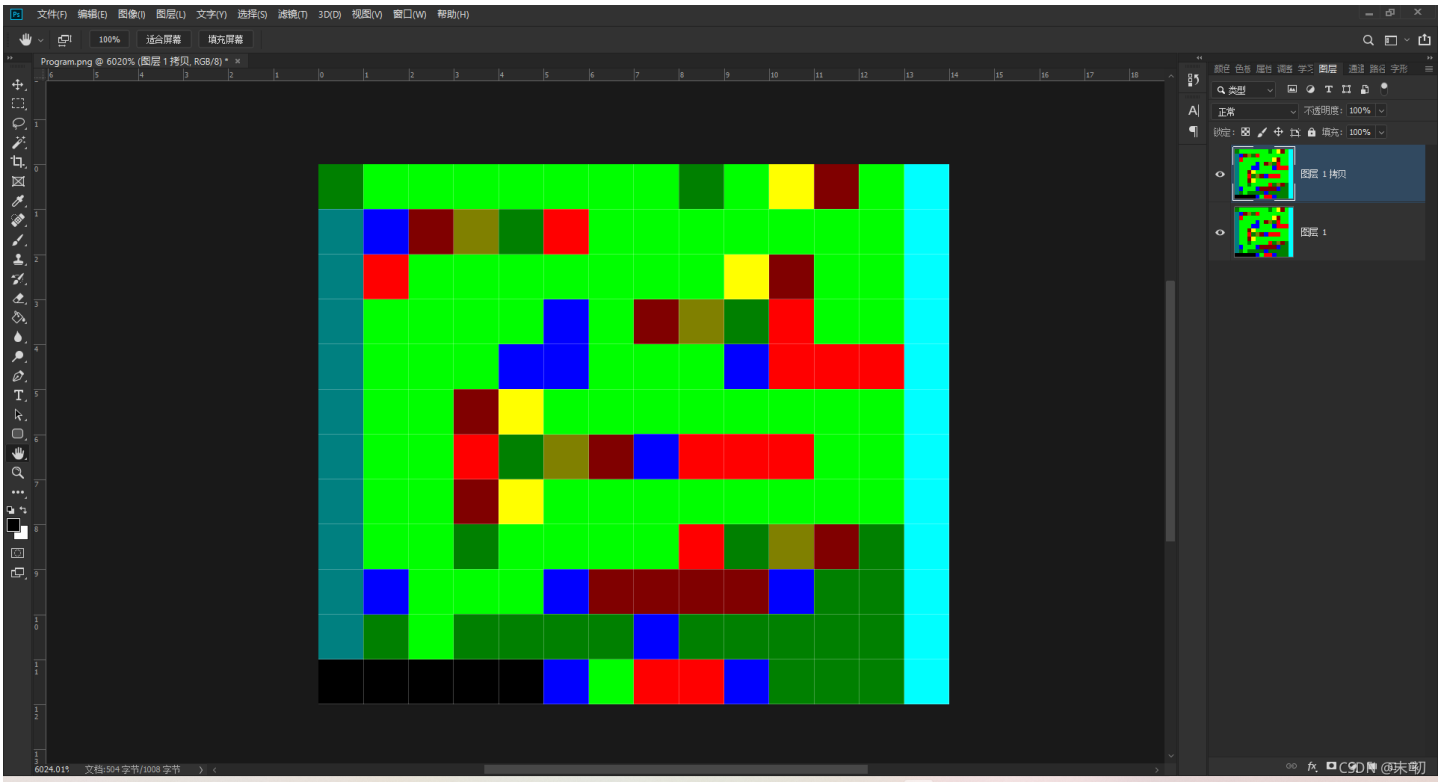
- lolcode-language：https://www.dcode.fr/lolcode-language

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





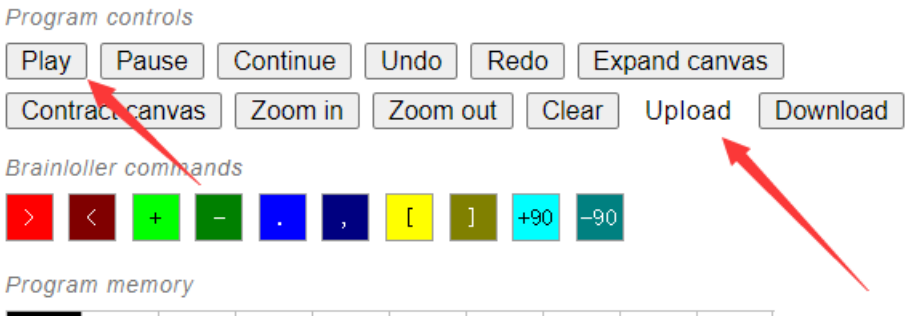
`Program.png` 根据名称提示一开始以为是 `npiet`，尝试直接编译发现不对

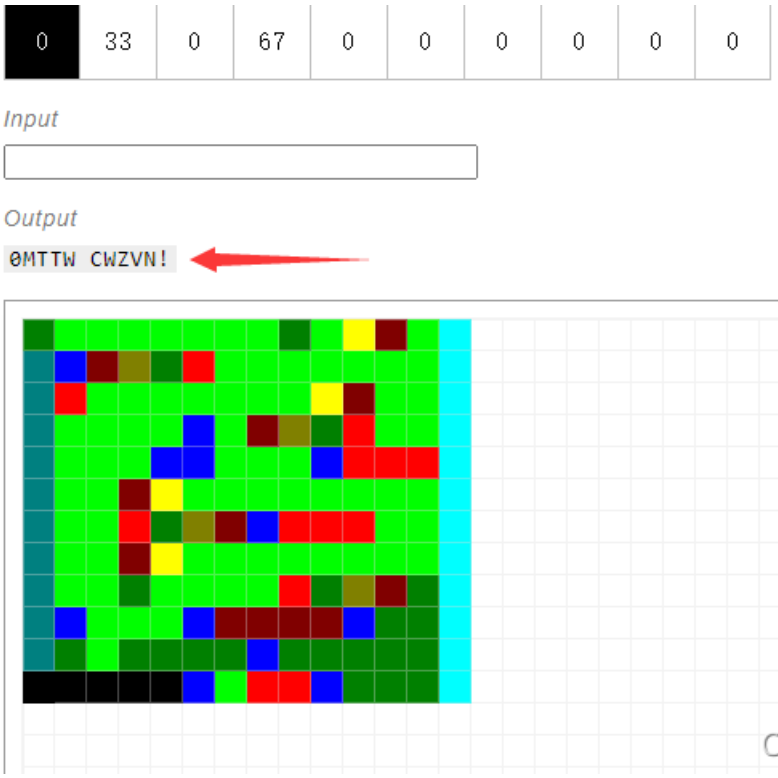


后来经过查阅资料才发现 `Brainfuck` 也有一种用像素颜色表示的语言：`Brainloller`

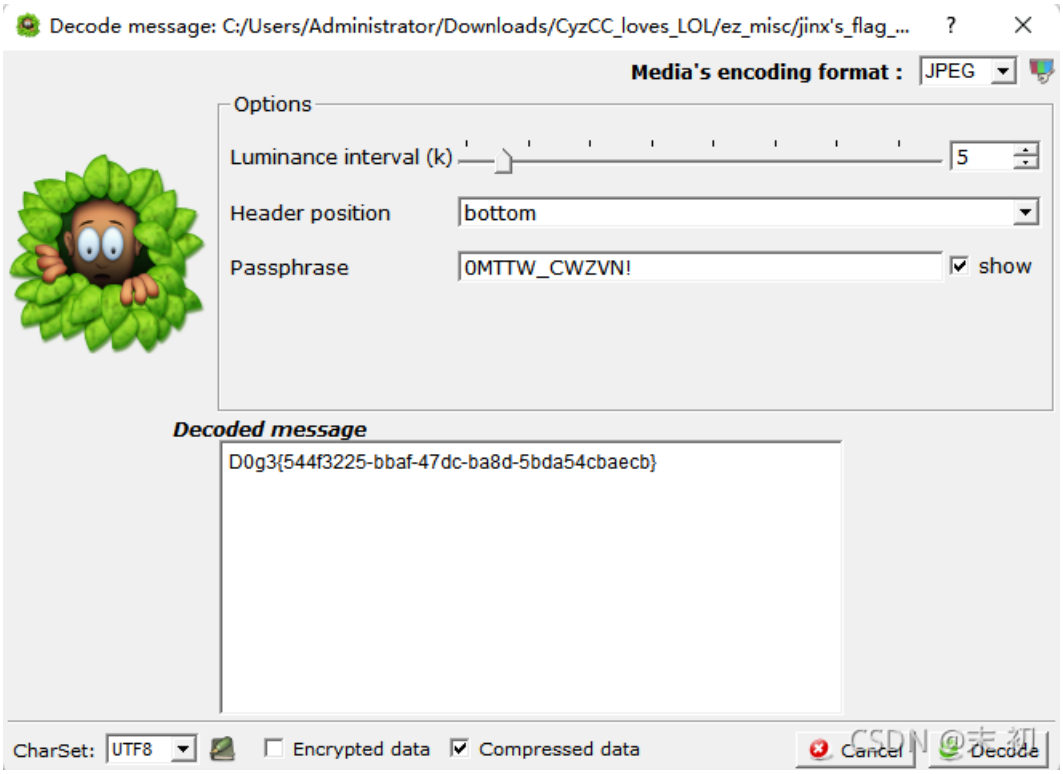
- https://minond.xyz/brainloller/

上传之后点击 `Play`，得到密码：`0MTTW CWZVN!`



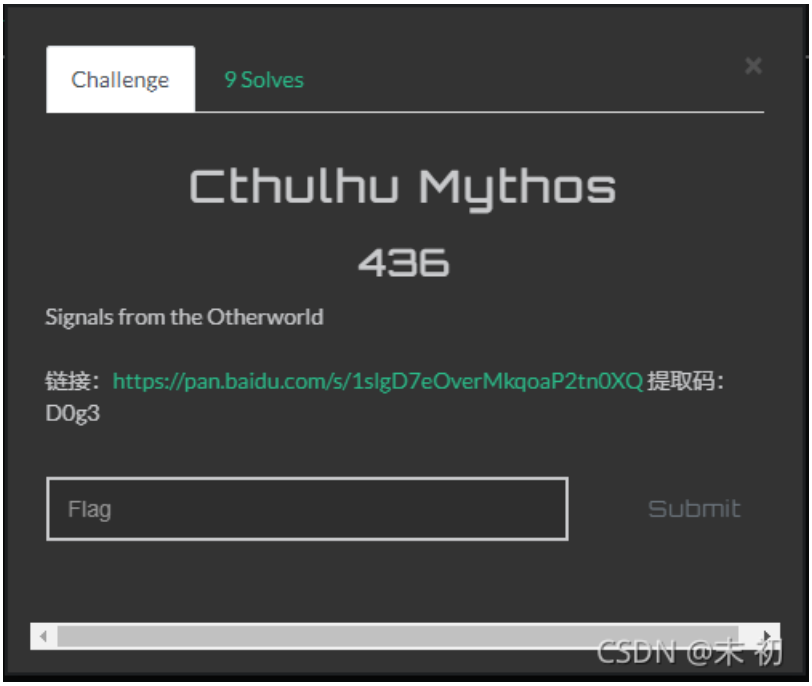


然后根据题目名称提示将密码中的空格换成下划线、以及`jinx's_flag_in_silent.jpg`的名称，直接尝试`SilentEye`解密

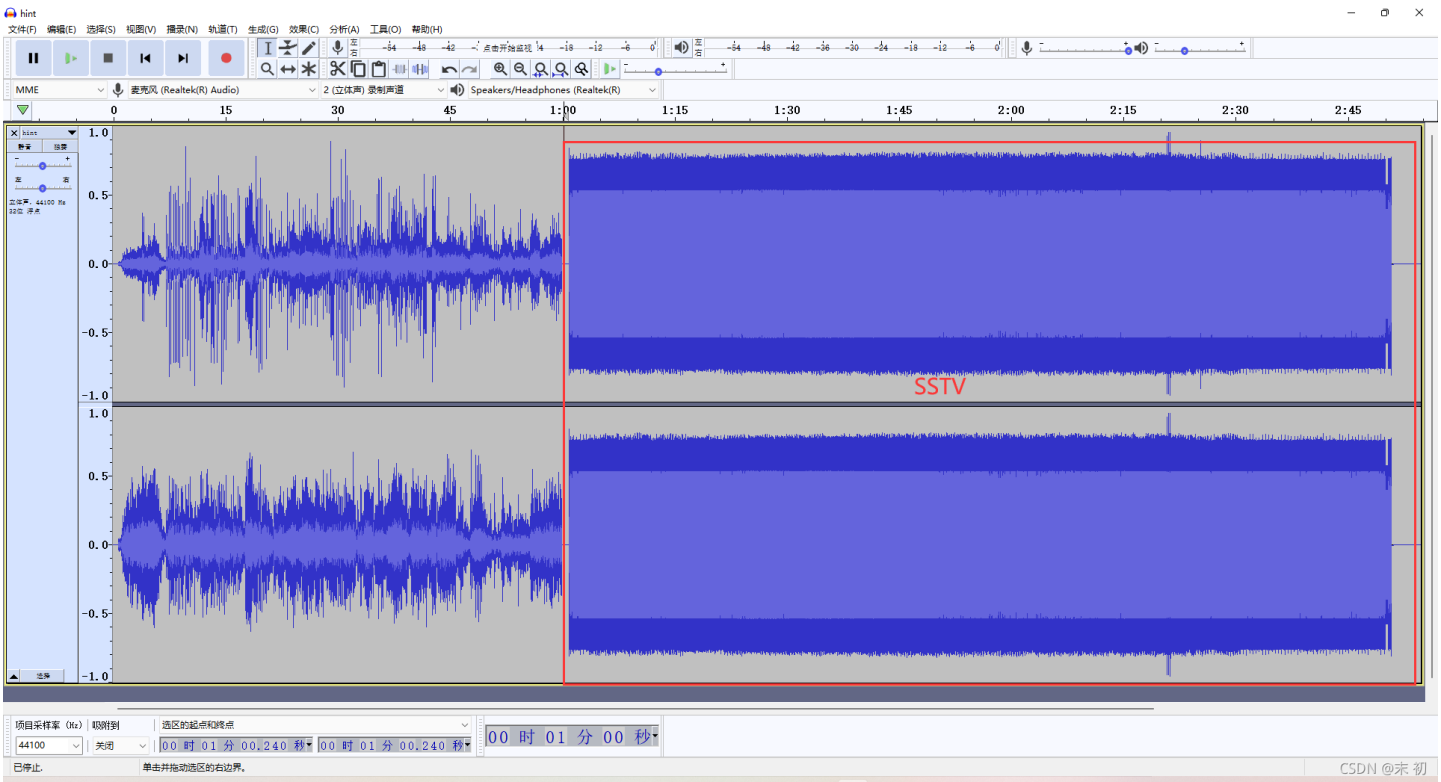



```
D0g3{544f3225-bbaf-47dc-ba8d-5bda54cbaecb}
```

Cthulhu Mythos



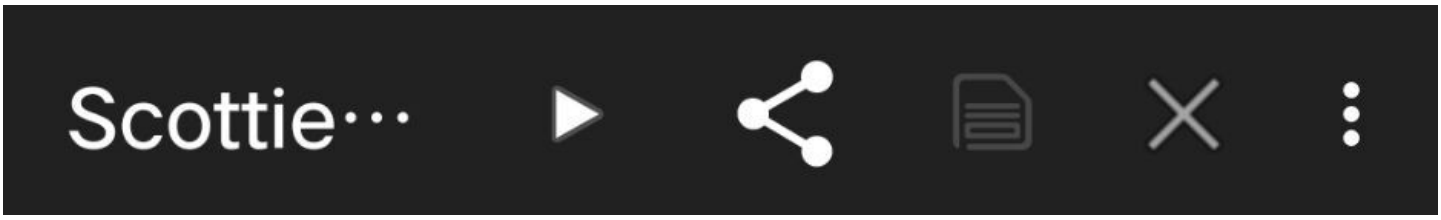
`hint.mp3`听一下，发现前面是泰拉瑞亚的主题曲，后面部分很明显是SSTV



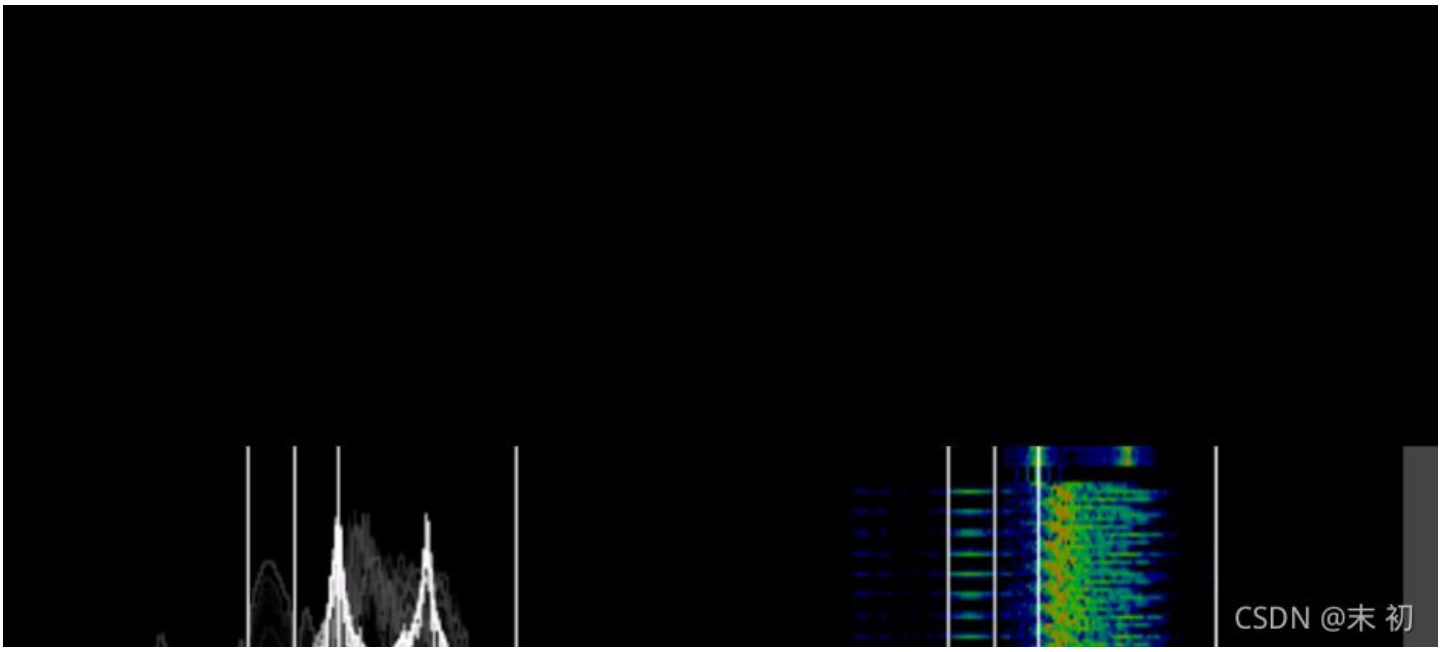
因为格式问题没法直接用`QSSTV`，`RX-SSTV`的话又比较麻烦要调整电脑录音设备，就直接用`Robot36`听吧

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

下载好传到手机上，安装，然后听就完事了，多听几遍确认信息




MRPVI4TZL5KGK4TSGRZGSYJB
PU======
Lunatic Legend




```
>>> from base64 import *
>>> b32decode('MRPVI4TZL5KGK4TSGRZGSYJBPU======')
b'd_Try_Terr4ria!}'
```

`The Evil Watcher.wld`

.WLD 文件扩展名

▶ **3个文件类型** 使用 **.wld** 文件扩展名。

档案类型1　Terraria世界档案

开发人员　重新逻辑

声望　4.3 (47 投票)

类别　游戏档案

格式　二进制

什么是WLD文件?

Terraria使用的游戏文件，这是2D动作冒险和沙盒构建游戏； 保存一个包含地形，水，怪物，物品和其他物体的世界； 用于单人或多人游戏，可以由Terraria专用服务器加载，该服务器允许玩家通过网络托管Terraria世界。

更多信息

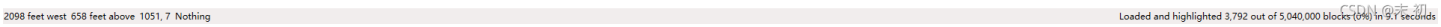
世界文件可以是预先安装的游戏地图，也可以是用户创建的自定义地图。 WLD文件保存到Windows中的以下目录： `[用户] \ Documents \ My Games \ Terraria \ Worlds \`.

可以使用Terraria地图编辑器（TEdit）编辑WLD文件。 可以使用MoreTerra（Terraria World Viewer）在没有Terraria游戏的情况下查看它们。

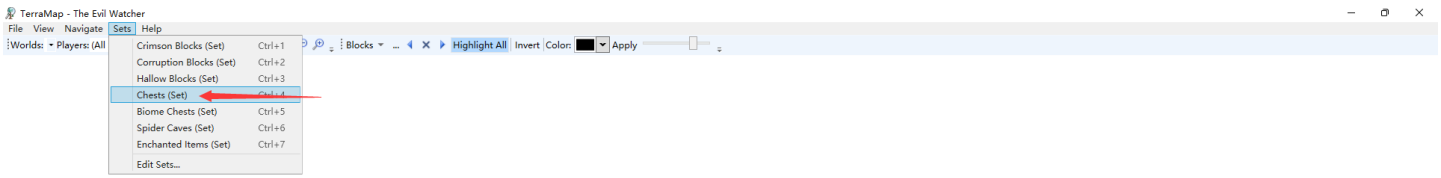
关于泰拉瑞亚地图编辑器

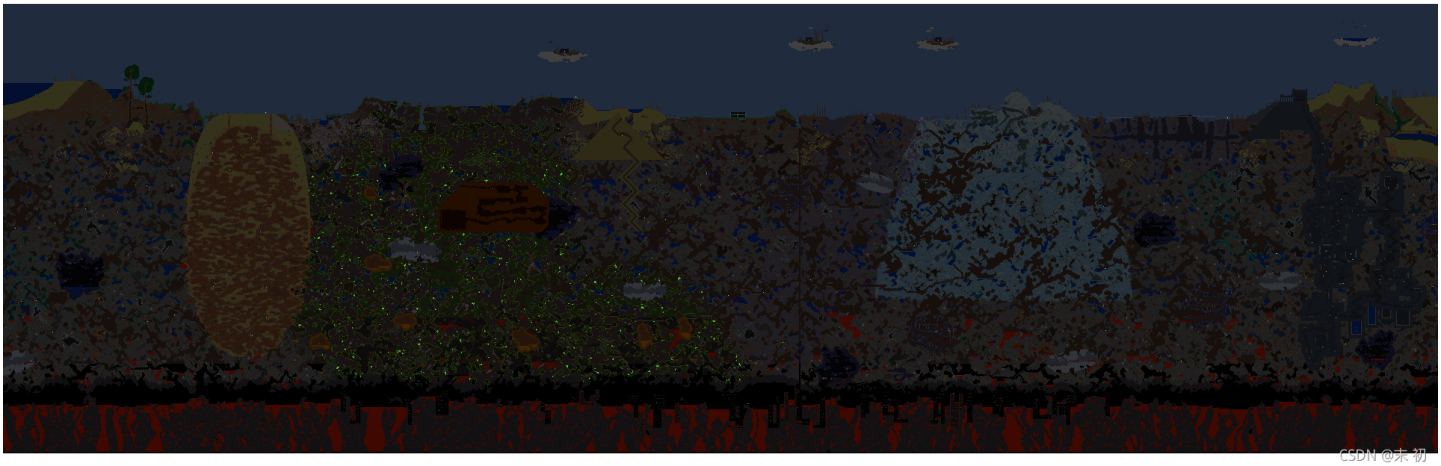
- https://www.bilibili.com/read/cv275739?from=search
- https://www.binaryconstruct.com/downloads/
- https://www.bilibili.com/video/BV1Za4y1a7uN
- https://m33.wiki/extension/wld.html

使用`TerraMap`打开`The Evil Watcher.wld`； `Players->All Spoilers`

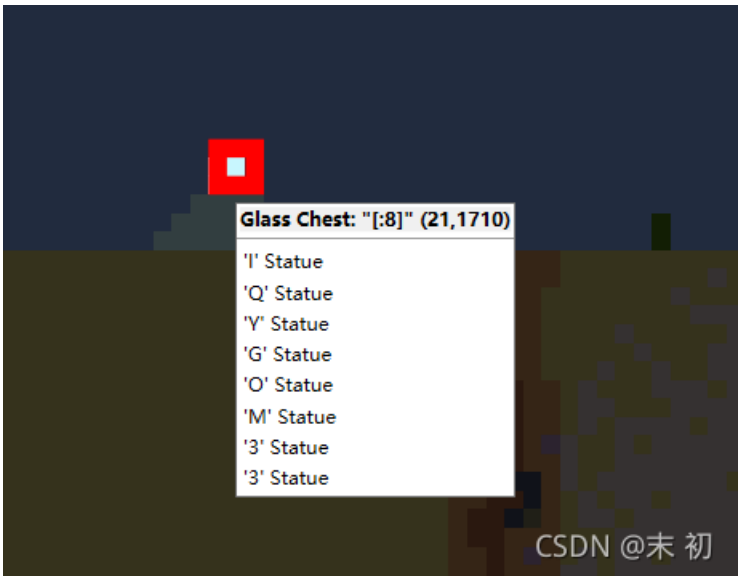


`Sets->Chests` 找宝箱

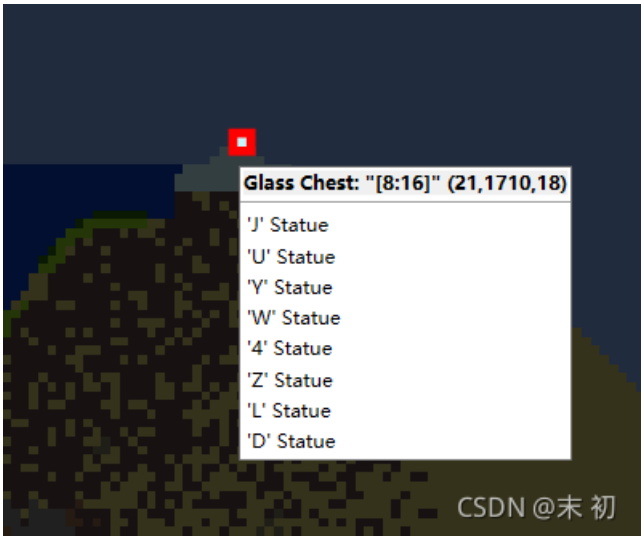




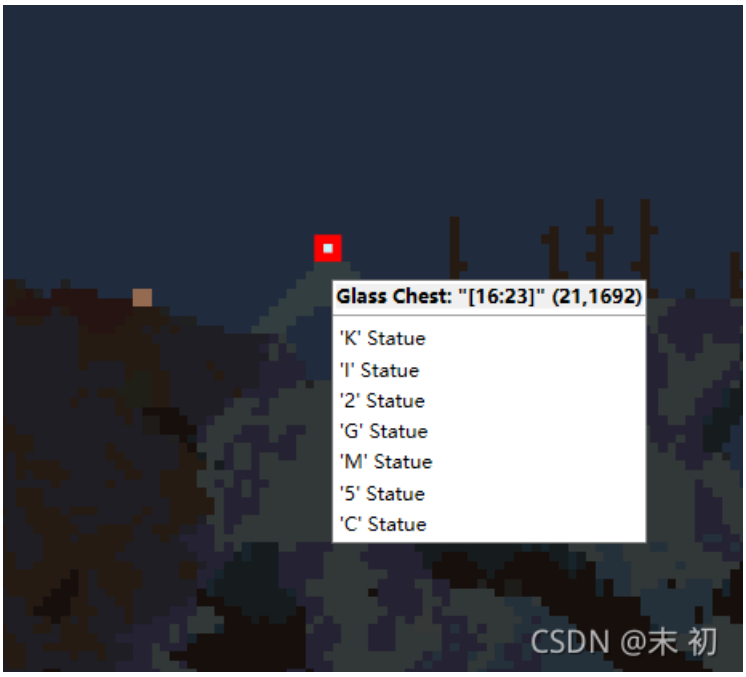
在地面上找到四个`Class Chest`




```
IQYGOM33
```




```
JUYW4ZLD
```




```
KI2GM5C
```

总共四部分，前三部分为：`IQYGOM33JUYW4ZLDKI2GM5C`

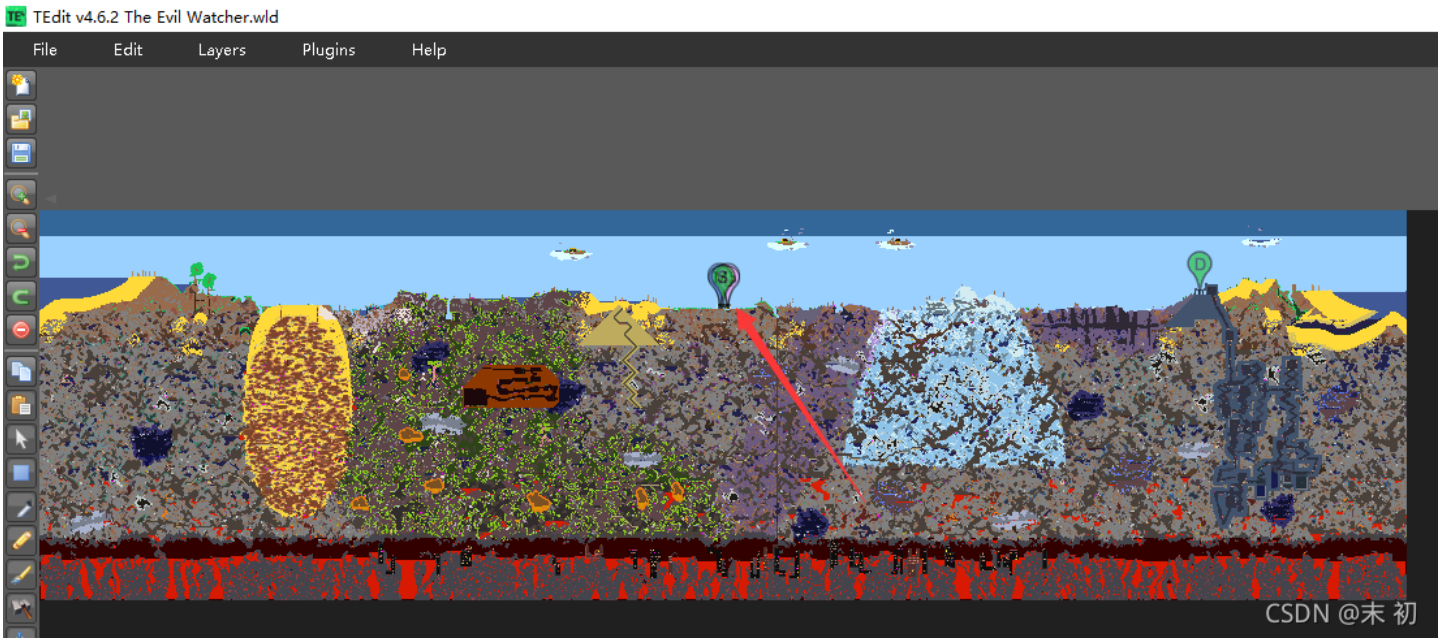
Base32

IQYGOM33JUYW4ZLDKI2GM5C

编码 解码 清空

D0g3{M1necR4ft@

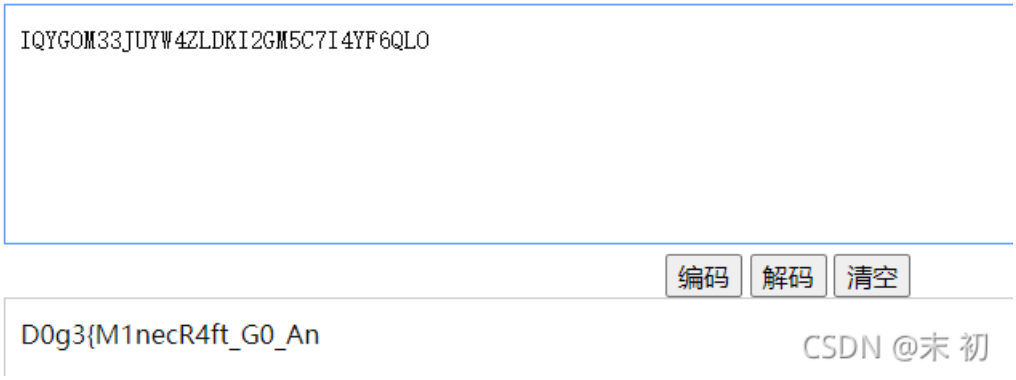
第四部分可以使用`TEdit`打开`The Evil Watcher.wld`



在这个位置找到第四部分

```
7I4YF6QLO
```

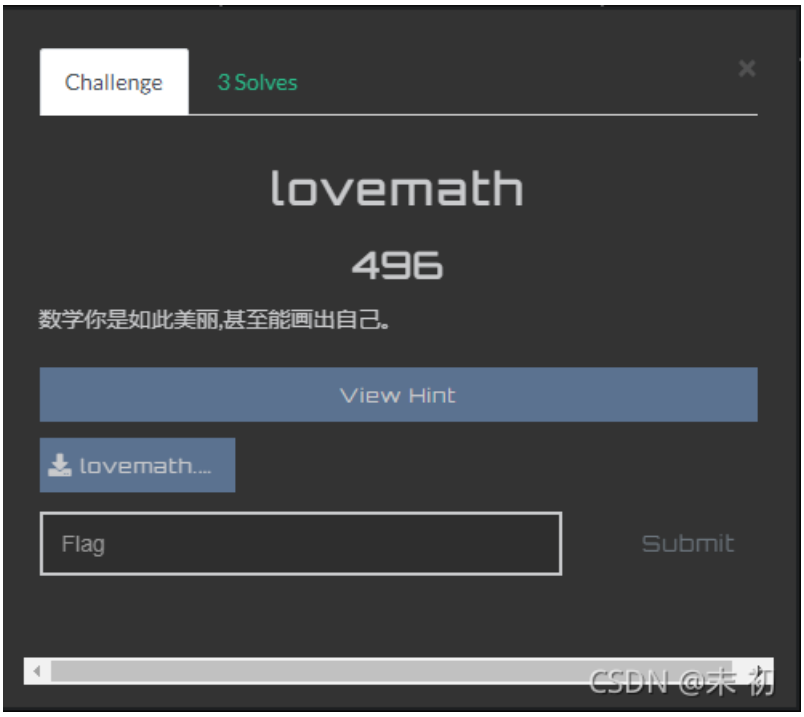
最终得到的base32为：`IQYGOM33JUYW4ZLDKI2GM5C7I4YF6QLO`



最终flag

```
D0g3{M1necR4ft_G0_And_Try_Terr4ria!}
```

lovemath




```
hint: not blindwater but you can search it
```

crc32爆破


```
-------------Filename CRC Info-------------
[+] flag.zip: 0xc38199da
[+] flag_01.txt: 0xa430239a
[+] flag_02.txt: 0xf81abecd
[+] flag_03.txt: 0x2a75b14e
[+] flag_04.txt: 0x2d2c423c
[+] flag_05.txt: 0xd9e12803
-------------------------------------------
```

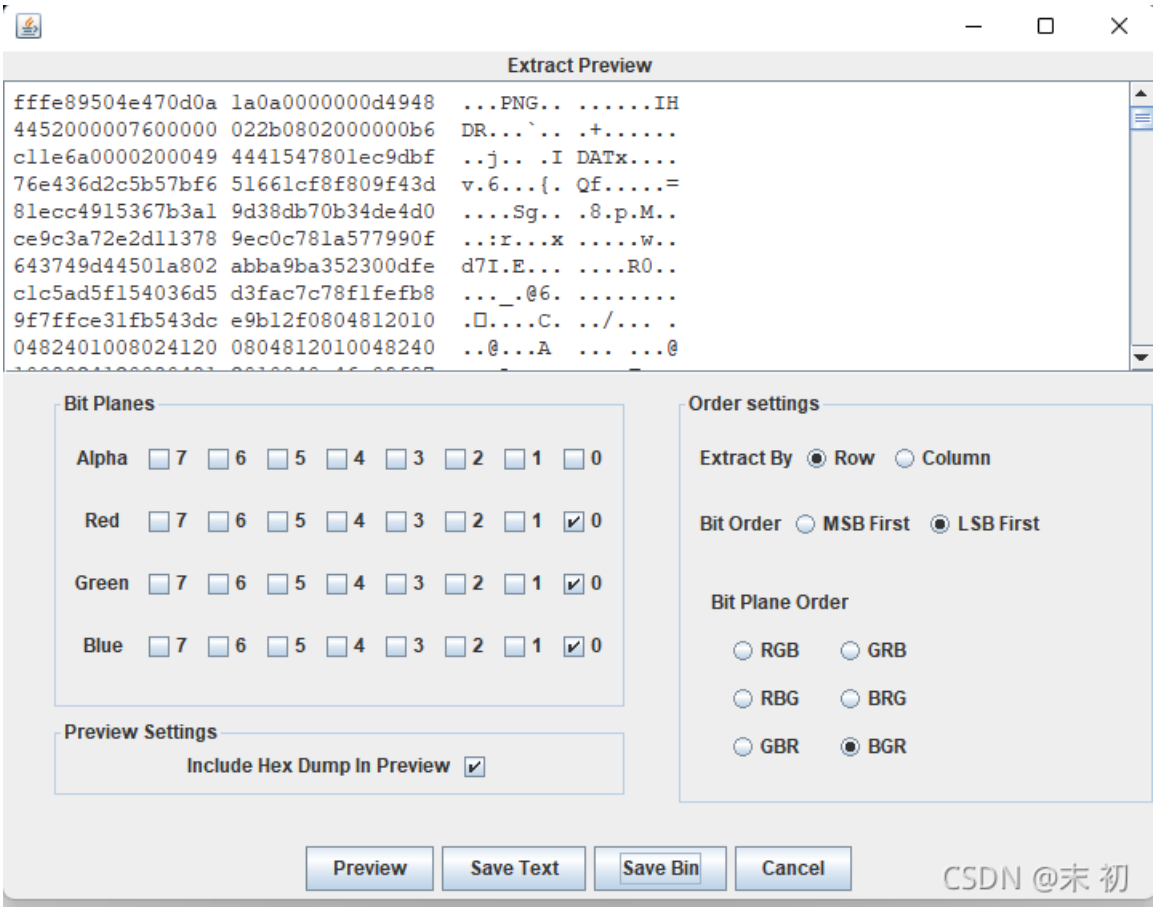
```
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0xa430239a
4 bytes: {0x56, 0x34, 0xbc, 0x00}
verification checksum: 0xa430239a (OK)
alternative: 3RAsk0 (OK)
alternative: 5jYl3N (OK)
alternative: ANz4c9 (OK)
alternative: DViXxW (OK)
alternative: EJg5bZ (OK)
alternative: JE94EM (OK)
alternative: JYvhDY (OK)
alternative: O1YuZg (OK)
alternative: R3ix8v (OK)
alternative: _lAJB8 (OK)
alternative: dYZaCR (OK)
alternative: mOBoUw (OK)
alternative: pMrb7f (OK)
alternative: qlmCE3 (OK)
alternative: sq6Mub (OK)
alternative: th1s_I (OK)
alternative: uhpBDP (OK)
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0xf81abecd
4 bytes: {0xf6, 0x44, 0x6a, 0xcc}
verification checksum: 0xf81abecd (OK)
alternative: 5XyM2J (OK)
alternative: 9kccWY (OK)
alternative: DdIyyS (OK)
alternative: MrQwov (OK)
alternative: ONTi6k (OK)
alternative: RLddTz (OK)
alternative: s_Y0ur (OK)
alternative: uZPcET (OK)
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0x2a75b14e
4 bytes: {0x39, 0x11, 0xcc, 0x5b}
verification checksum: 0x2a75b14e (OK)
```

```
alternative: 0njyYo (OK)
alternative: 4Wf40T (OK)
alternative: 7wmGsD (OK)
alternative: 8x3FTS (OK)
alternative: 9xrwOJ (OK)
alternative: Cn_SKk (OK)
alternative: KdI0GC (OK)
alternative: R_upLi (OK)
alternative: S3GlS4 (OK)
alternative: S_4AWp (OK)
alternative: UFrNfB (OK)
alternative: W7ZmRW (OK)
alternative: _pa33w (OK)
alternative: caljpn (OK)
alternative: d5Fi7M (OK)
alternative: dxkTZE (OK)
alternative: jwtdfK (OK)
alternative: ln2kWy (OK)
alternative: rPFIwl (OK)
alternative: w8iTiR (OK)
alternative: x77UNE (OK)
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0x2d2c423c
4 bytes: {0x78, 0x6b, 0xc3, 0x45}
verification checksum: 0x2d2c423c (OK)
alternative: 0rd_We (OK)
alternative: 1nj2Mh (OK)
alternative: BSNT74 (OK)
alternative: CrQuEa (OK)
alternative: DkVKoJ (OK)
alternative: FWSU6W (OK)
alternative: P2Suvv (OK)
alternative: Sbdw06 (OK)
alternative: Wfyv1U (OK)
alternative: ZIm5NK (OK)
alternative: dderTO (OK)
alternative: jkzBhA (OK)
alternative: rLHoyf (OK)
alternative: uUOQSM (OK)
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0xd9e12803
4 bytes: {0x9f, 0x48, 0x0c, 0x36}
verification checksum: 0xd9e12803 (OK)
alternative: 1c0m3e (OK)
alternative: 2_tBqa (OK)
alternative: 3_5sjx (OK)
alternative: 98DoSQ (OK)
alternative: A_Ahce (OK)
alternative: FFFVIN (OK)
alternative: J8qEAU (OK)
alternative: K80tZL (OK)
alternative: NLP5Ef (OK)
alternative: PnkKdg (OK)
alternative: Rs0ET6 (OK)
alternative: WwluNL (OK)
alternative: YxsErB (OK)
alternative: ZD7j0F (OK)
alternative: ZXx61R (OK)
alternative: _a5JCp (OK)
alternative: fIuorK (OK)
alternative: hFj_NE (OK)
alternative: jZd2IH (OK)
```

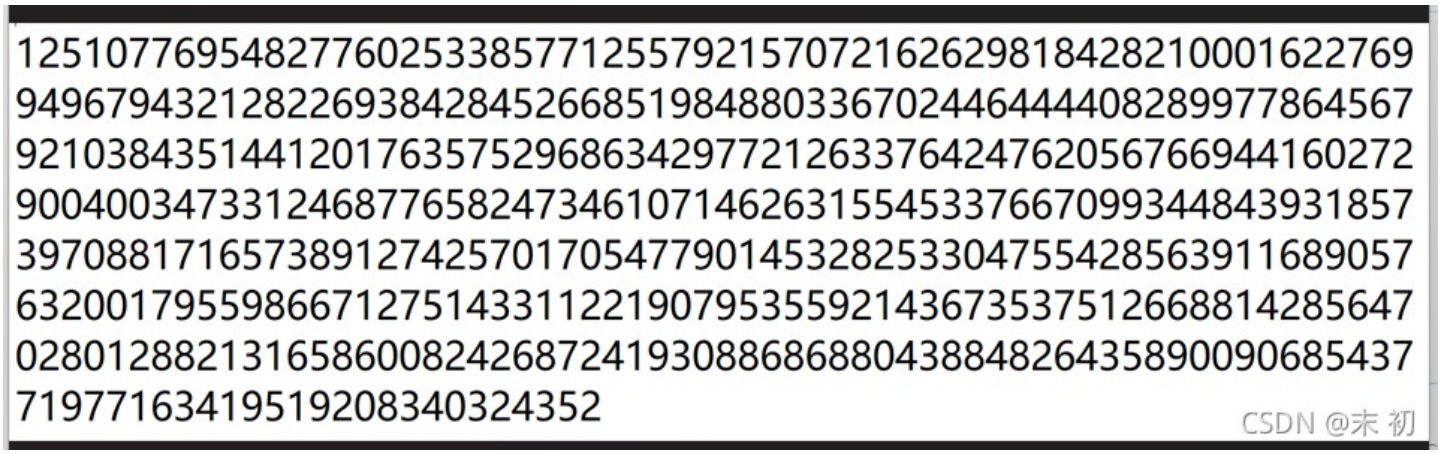
```
alternative: o_madn (OK)
alternative: paXr_b (OK)
alternative: wx_LuI (OK)
PS D:\Tools\Misc\crc32>
```

得到密码：`th1s_Is_Y0ur_pa33w0rd_We1c0m3e`

`blind.png`存在LSB隐写PNG内容



保存下来使用010Editor将前面的几个干扰字节去掉，得到图片



图片OCR：https://www.onlineocr.net/zh_hant/

```
12510776954827760253385771255792157072162629818428210001622769949679432128226938428452668519848803367024464444082899778645679210384351441201763575296863429772126337642476205676694416027290040034733124687765824734610714626315545337667099344843931857397088171657389127425701705477901453282533047554285639116890576320017955986671275143311221907953559214367353751266881428564702801288213165860082426872419308868688043884826435890090685437719771634195192083403243 52
```

根据题目给出的提示`画出自己`；需要用到一种叫`塔珀自指公式(Tupper's self-referential formula)`的公式

- https://zh.wikipedia.org/wiki/%E5%A1%94%E7%8F%80%E8%87%AA%E6%8C%87%E5%85%AC%E5%BC%8F

脚本参考：https://www.cnblogs.com/1024th/p/14418846.html

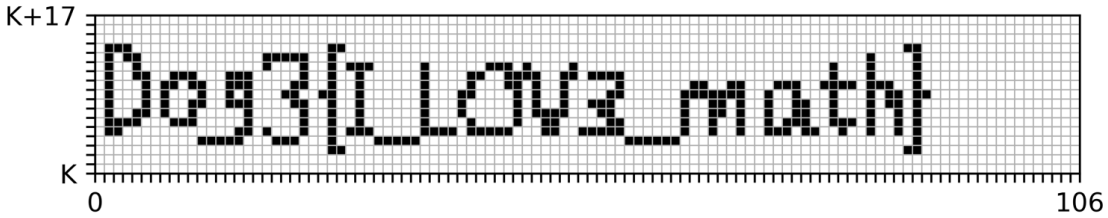
把`K`的值换成上面的数字即可

```
"""
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")
```

照片 - Tupper-plot.png

全屏

$$\frac{1}{2} < \left\lfloor \mathrm{mod}\left(\left\lfloor \frac{y}{17} \right\rfloor 2^{-17\lfloor x \rfloor - \mathrm{mod}(\lfloor y \rfloor, 17)}, 2 \right) \right\rfloor$$



K=12510776954827760253385771255792157072162629818428210001622769949679
432128226938428452668519848803367024464444082899778645679210384351441
201763575296863429772126337642476205676694416027290040034733312468776
582473461071462631554533766709934484393185739708817165738912742570170
547790145328253304755428563911689057632001795598667127514331122190795
355921436735375126688142856470280128821316586008242687241930886868804
388482643589009068543771977163419519208340324352

CSDN @未 初

```
D0g3{I_Lov3_math}
```