# 2020 第三届江西省高校网络安全技能大赛 线上赛Writeup

原创

F10NAF11pp3d 于 2020-08-31 20:39:44 发布 1136 收藏 3

分类专栏： CTF线上赛 文章标签： 网络安全 php



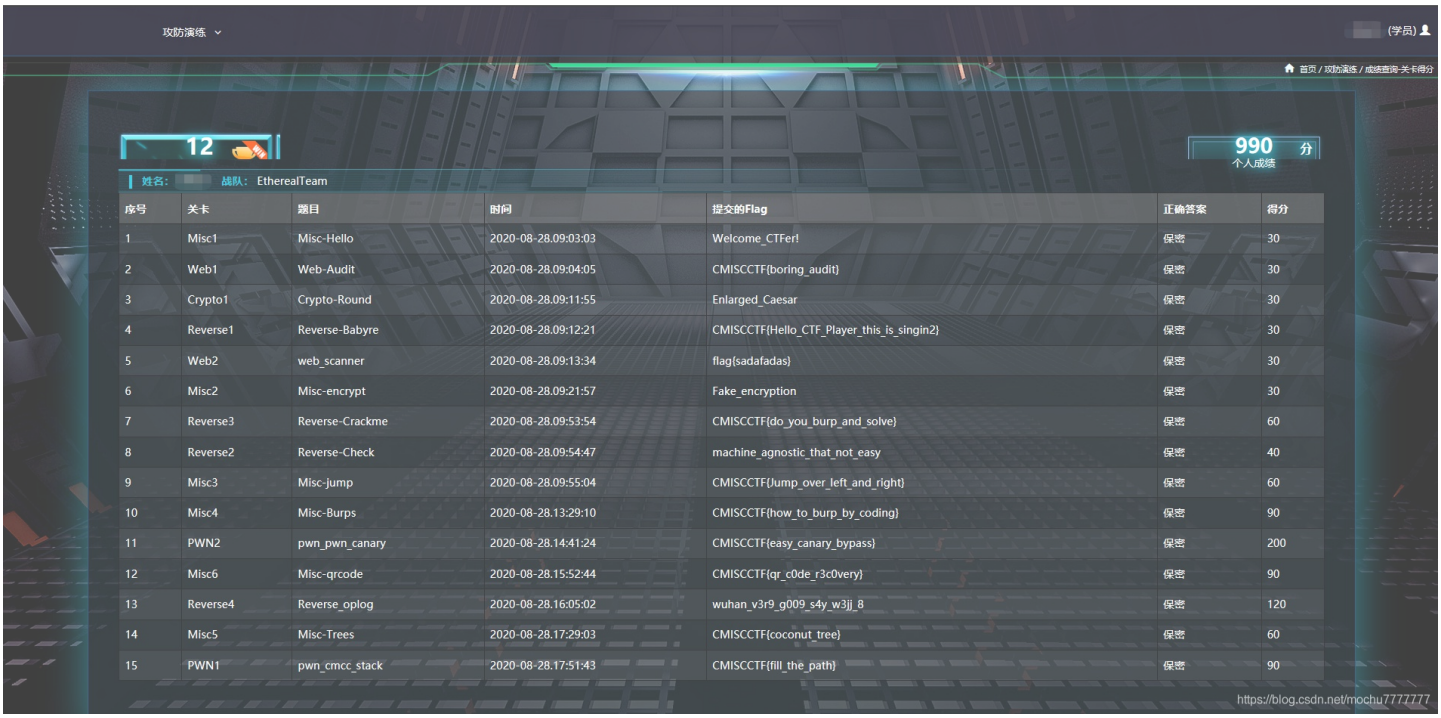
本文链接：https://blog.csdn.net/m0_46481239/article/details/108329008



CTF线上赛 专栏收录该内容

4 篇文章 1 订阅

订阅专栏




| 序号 | 关卡 | 题目 | 时间 | 提交的Flag | 正确答案 | 得分 |
|---|---|---|---|---|---|---|
| 1 | Misc1 | Misc-Hello | 2020-08-28.09:03:03 | Welcome_CTFer! | 保密 | 30 |
| 2 | Web1 | Web-Audit | 2020-08-28.09:04:05 | CMISCCTF{boring_audit} | 保密 | 30 |
| 3 | Crypto1 | Crypto-Round | 2020-08-28.09:11:55 | Enlarged_Caesar | 保密 | 30 |
| 4 | Reverse1 | Reverse-Babyre | 2020-08-28.09:12:21 | CMISCCTF{Hello_CTF_Player_this_is_singin2} | 保密 | 30 |
| 5 | Web2 | web_scanner | 2020-08-28.09:13:34 | flag{sadafadas} | 保密 | 30 |
| 6 | Misc2 | Misc-encrypt | 2020-08-28.09:21:57 | Fake_encryption | 保密 | 30 |
| 7 | Reverse3 | Reverse-Crackme | 2020-08-28.09:53:54 | CMISCCTF{do_you_burp_and_solve} | 保密 | 60 |
| 8 | Reverse2 | Reverse-Check | 2020-08-28.09:54:47 | machine_agnostic_that_not_easy | 保密 | 40 |
| 9 | Misc3 | Misc-jump | 2020-08-28.09:55:04 | CMISCCTF{Jump_over_left_and_right} | 保密 | 60 |
| 10 | Misc4 | Misc-Burps | 2020-08-28.13:29:10 | CMISCCTF{how_to_burp_by_coding} | 保密 | 90 |
| 11 | PWN2 | pwn_pwn_canary | 2020-08-28.14:41:24 | CMISCCTF{easy_canary_bypass} | 保密 | 200 |
| 12 | Misc6 | Misc-qrcode | 2020-08-28.15:52:44 | CMISCCTF{qr_c0de_r3c0very} | 保密 | 90 |
| 13 | Reverse4 | Reverse_oplog | 2020-08-28.16:05:02 | wuhan_v3r9_g009_s4y_w3jj_8 | 保密 | 120 |
| 14 | Misc5 | Misc-Trees | 2020-08-28.17:29:03 | CMISCCTF{coconut_tree} | 保密 | 60 |
| 15 | PWN1 | pwn_cmcc_stack | 2020-08-28.17:51:43 | CMISCCTF{fill_the_path} | 保密 | 90 |

## 赛题类型

- Misc
  - Misc1-Hello
  - Misc2-encrypt
  - Misc3-jump
  - Misc4-Burps
  - Misc5-Trees
  - Misc6-qrcode
  - Misc7-blind
  - Misc8-music
- Web
  - Web1-Audit
  - Web2-scanner
  - Web3-greatctf
  - Web4-Admin
- Reverse
  - Reverse1-Babyre
  - Reverse2-Check
  - Reverse3-Crackme
  - Reverse4-oplog
- Pwn
  - Pwn1_cmcc_stack
  - Pwn2_pwn_canary
- Crypto
  - Crypto1-Round


---

# Misc

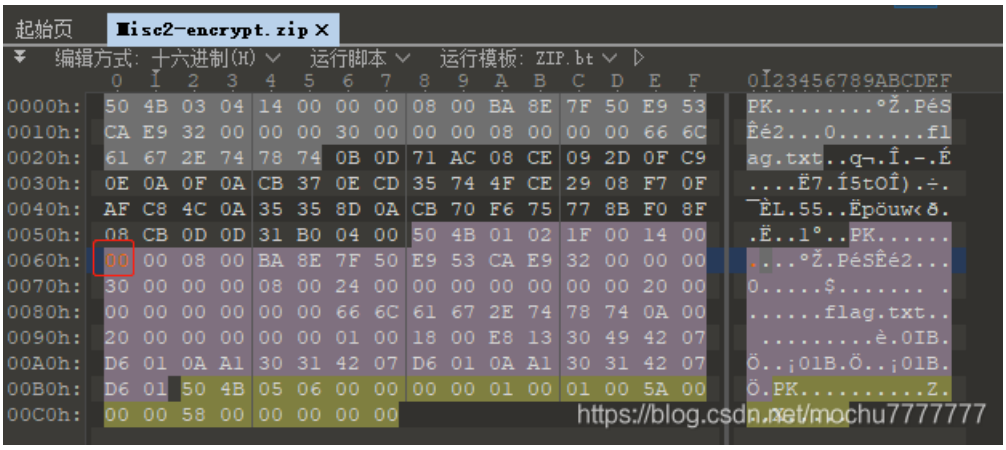
## Misc1-Hello

```
PS C:\Users\Administrator\Downloads> php -r "var_dump(base64_decode('Q01JU0NDVEZ7V2VsY29tZV9DVEZlciF9'));"
string(24) "CMISCCTF{Welcome_CTFer!}"
```

```
CMISCCTF{Welcome_CTFer!}
```

## Misc2-encrypt

zip伪加密



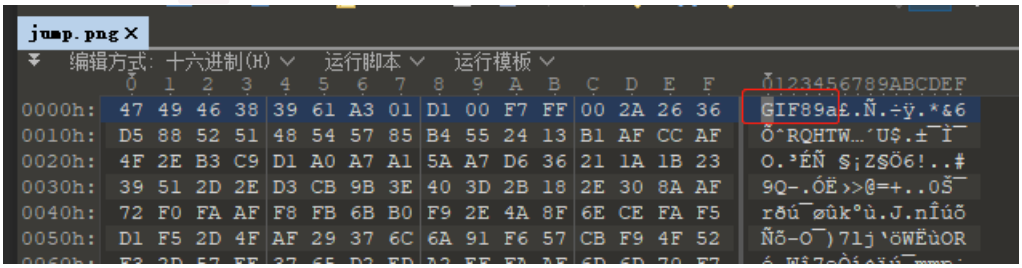

修改这个位置的`09`为`00`或者其他`偶数`即可，解压得到`base64`密文，两次`base64 decode`即可

```
PS C:\Users\Administrator\Downloads> php -r "var_dump(base64_decode('UTAxSlUwTkRWRVo3Um1GclpWOWxibU55ZVhCMGFXOXVmUT09'));"
string(36) "Q01JU0NDVEZ7RmFrZV9lbmNyeXB0aW9ufQ=="
PS C:\Users\Administrator\Downloads> php -r "var_dump(base64_decode('Q01JU0NDVEZ7RmFrZV9lbmNyeXB0aW9ufQ=='));"
string(25) "CMISCCTF{Fake_encryption}"
```

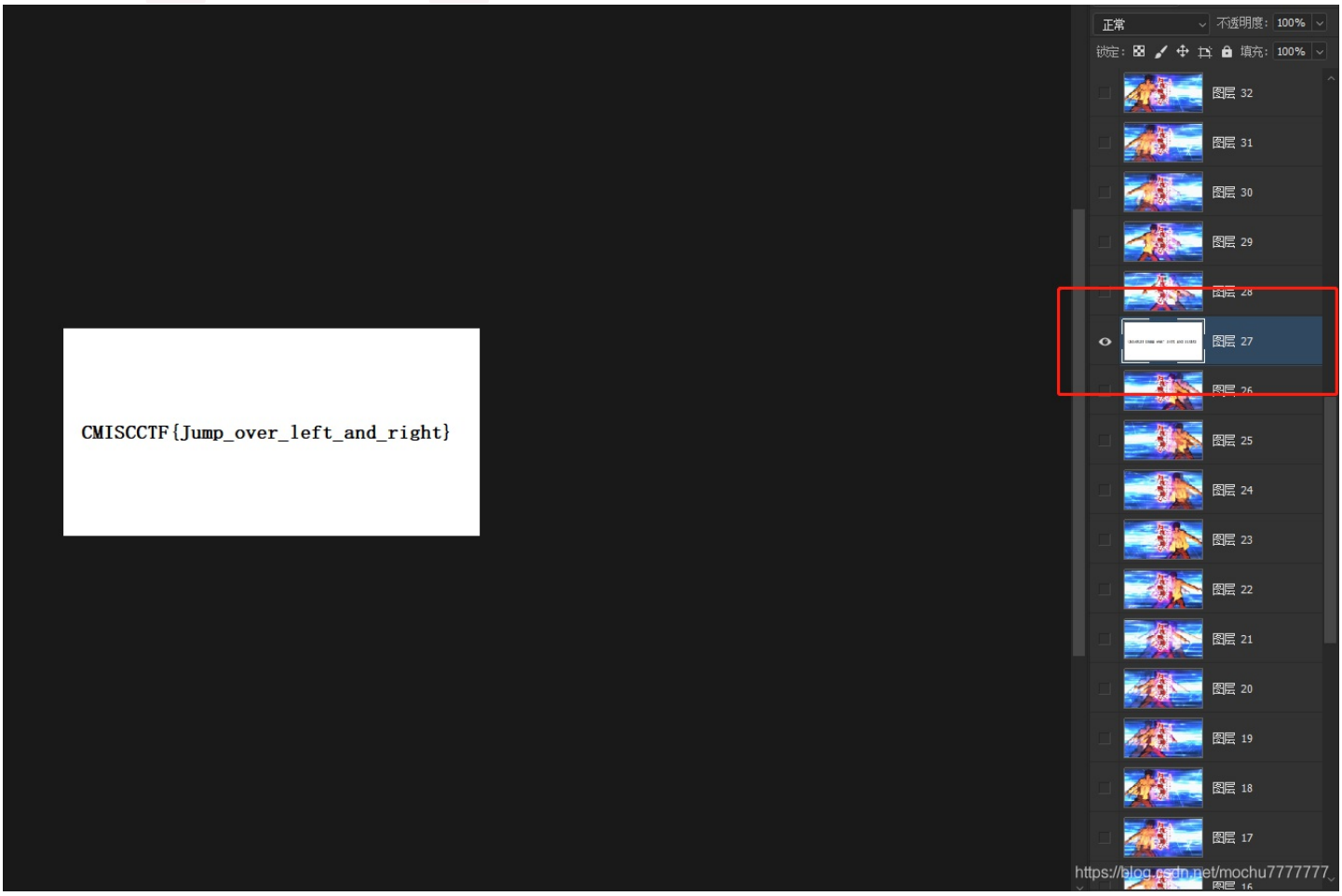
```
CMISCCTF{Fake_encryption}
```

## Misc3-jump

`png`的图，`GIF`的头，修改后缀为`gif`



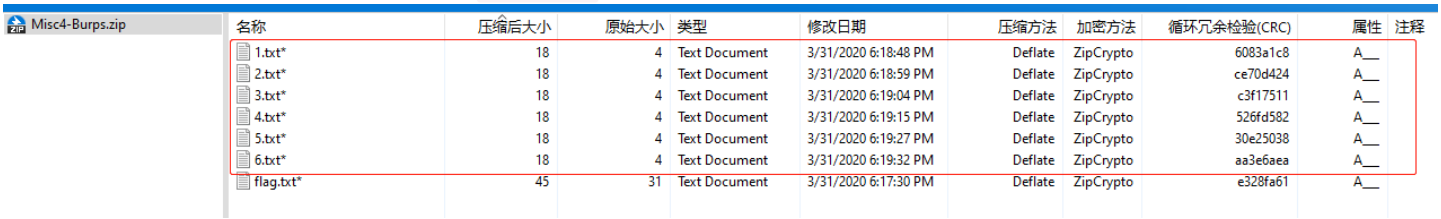

动图，会显示`flag`，使用PS打开，找到`flag`所在的图层即可




```
CMISCCTF{Jump_over_left_and_right}
```

## Misc4-Burps

原始文件很小，一般小于5Byte即可猜测为`CRC32碰撞`




CRC32碰撞脚本: https://github.com/theonlypwner/crc32

将`1.txt-6.txt`的冗余校验码提取出来

```
1.txt 0x6083a1c8
2.txt 0xce70d424
3.txt 0xc3f17511
4.txt 0x526fd582
5.txt 0x30e25038
6.txt 0xaa3e6aea
```

然后使用脚本碰撞，观察每一次得到的`4 bytes`，每一次得到4块十六进制数，将十六进制转换为字符

```
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0x6083a1c8
4 bytes: {0x74, 0x68, 0x65, 0x5f}
verification checksum: 0x6083a1c8 (OK)
alternative: Cb9fTf (OK)
alternative: Jhn4CW (OK)
alternative: K9Md4R (OK)
alternative: Lmggsq (OK)
alternative: Nls5B4 (OK)
alternative: PNHKc5 (OK)
alternative: TJUJbV (OK)
alternative: X4bYjM (OK)
alternative: apQQ_2 (OK)
alternative: bmKntz (OK)
alternative: e9am3Y (OK)
alternative: tEfsYS (OK)
alternative: v4NPmF (OK)
PS D:\Tools\Misc\crc32> php -r "var_dump(hex2bin('7468655f'));"
string(4) "the_"
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0xce70d424
4 bytes: {0x70, 0x61, 0x73, 0x73}
verification checksum: 0xce70d424 (OK)
alternative: 1bItKW (OK)
alternative: 4GKT8a (OK)
alternative: 5fTuJ4 (OK)
alternative: Ffa_X0 (OK)
alternative: KtdPOv (OK)
alternative: TJQCtz (OK)
alternative: WjZ07j (OK)
alternative: ZENsHt (OK)
alternative: bmOgbV (OK)
alternative: dhF4Rp (OK)
alternative: fiRfc5 (OK)
alternative: l3262D (OK)
alternative: wXxEd7 (OK)
alternative: yWguX9 (OK)
alternative: yjv90a (OK)
alternative: yv9e1u (OK)
PS D:\Tools\Misc\crc32> php -r "var_dump(hex2bin('70617373'));"
string(4) "pass"
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0xc3f17511
4 bytes: {0x77, 0x6f, 0x72, 0x64}
verification checksum: 0xc3f17511 (OK)
alternative: 4GLZ9v (OK)
alternative: FG8A0k (OK)
alternative: Jhm3Tl (OK)
alternative: SSQs_F (OK)
alternative: TJVMum (OK)
alternative: bmHicA (OK)
alternative: etOWIj (OK)
alternative: l3583S (OK)
alternative: lbWY_O (OK)
```

```
alternative: 1bWY_O (OK)
alternative: tEetNh (OK)
alternative: yjq71v (OK)
alternative: zJzDrf (OK)
PS D:\Tools\Misc\crc32> php -r "var_dump(hex2bin('776f7264'));"
string(4) "word"
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0x526fd582
4 bytes: {0x5f, 0x68, 0x65, 0x72}
verification checksum: 0x526fd582 (OK)
alternative: BbSWOR (OK)
alternative: FfNVN1 (OK)
alternative: JhE4Cz (OK)
alternative: KtKYYw (OK)
alternative: RS8ESI (OK)
alternative: SSytHP (OK)
alternative: WWduI3 (OK)
alternative: e9Jm3t (OK)
alternative: izl2S7 (OK)
alternative: u503G7 (OK)
alternative: v4ePmk (OK)
alternative: wXWLr6 (OK)
alternative: zJRCep (OK)
PS D:\Tools\Misc\crc32> php -r "var_dump(hex2bin('5f686572'));"
string(4) "_her"
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0x30e25038
4 bytes: {0x65, 0x5f, 0x63, 0x69}
verification checksum: 0x30e25038 (OK)
alternative: 8tDDMh (OK)
alternative: Gf5PS3 (OK)
alternative: Jt0_Du (OK)
alternative: Ktqn_l (OK)
alternative: LmvPuG (OK)
alternative: SSCCNK (OK)
alternative: Y42_wb (OK)
alternative: bmZYrL (OK)
alternative: cqT4hA (OK)
alternative: e9pZ5o (OK)
alternative: lbEiNB (OK)
alternative: m3f99G (OK)
alternative: t5K5Z5 (OK)
alternative: tEwD_e (OK)
alternative: v4_gkp (OK)
alternative: zJhtck (OK)
PS D:\Tools\Misc\crc32> php -r "var_dump(hex2bin('655f6369'));"
string(4) "e_ci"
PS D:\Tools\Misc\crc32> python .\crc32.py reverse 0xaa3e6aea
4 bytes: {0x70, 0x68, 0x65, 0x72}
verification checksum: 0xaa3e6aea (OK)
alternative: 8tQsKs (OK)
alternative: Eg45dm (OK)
alternative: FfaVN1 (OK)
alternative: Jhj4Cz (OK)
alternative: KtdYYw (OK)
alternative: SSVtHP (OK)
alternative: WWKuI3 (OK)
alternative: bmOntW (OK)
alternative: e9em3t (OK)
alternative: fiRou4 (OK)
alternative: izC2S7 (OK)
alternative: v4JPmk (OK)
```

```
alternative: wXxLr6 (OK)
PS D:\Tools\Misc\crc32> php -r "var_dump(hex2bin('70686572'));"
string(4) "pher"
PS D:\Tools\Misc\crc32>
```

得到解压密码

```
the_password_here_cipher
```

解压`flag.txt`即可得到flag

```
CMISCCTF{how_to_burp_by_coding}
```

## Misc5-Trees

解压得到这样一张图片


## 正解

```
from PIL import Image

img = Image.open('enc.png')
w = img.width
h = img.height
img_obj = Image.new("RGB",(w//16,h//16))

for x in range(w//16):
    for y in range(h//16):
        (r,g,b)=img.getpixel((x*16,y*16))
        img_obj.putpixel((x,y),(r,g,b))

img_obj.save('ok.png')
```


## 非正解

通过`stegsolve`适当调整颜色通道和偏移量或者使用`PS`调整颜色通道、拉曲线等一系列骚操作，能够模糊得到:`CMISCCTF{coconut_tree}`

结合背景图为椰树，椰树英文为：`coconut`。猜测flag

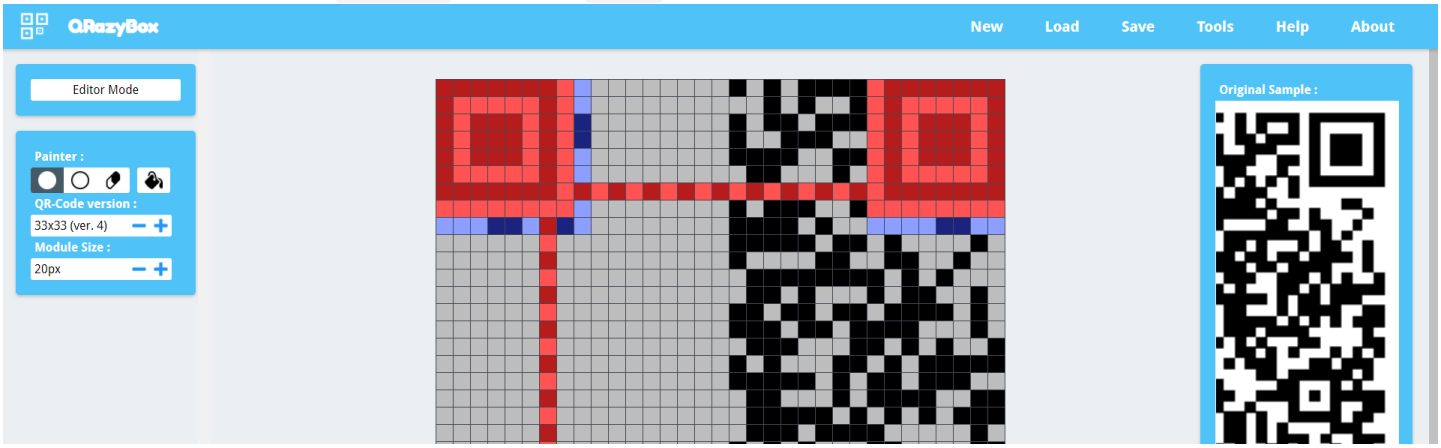
```
CMISCCTF{coconut_tree}
```

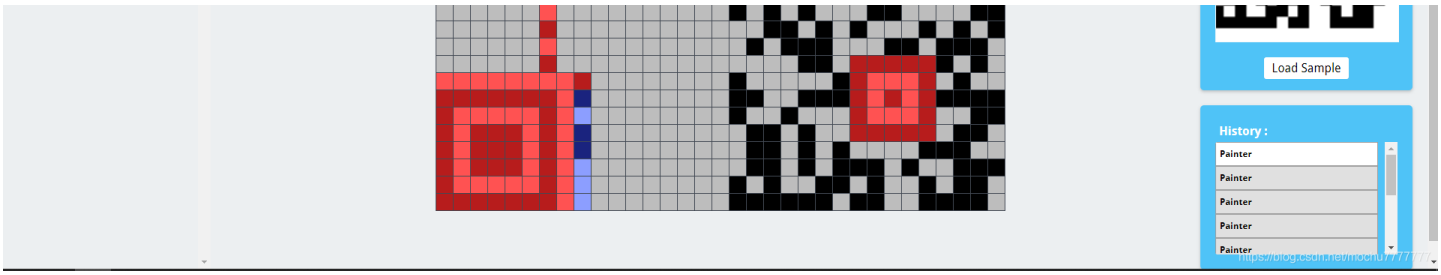
## Misc6-qrcode

解压得到，残缺的二维码

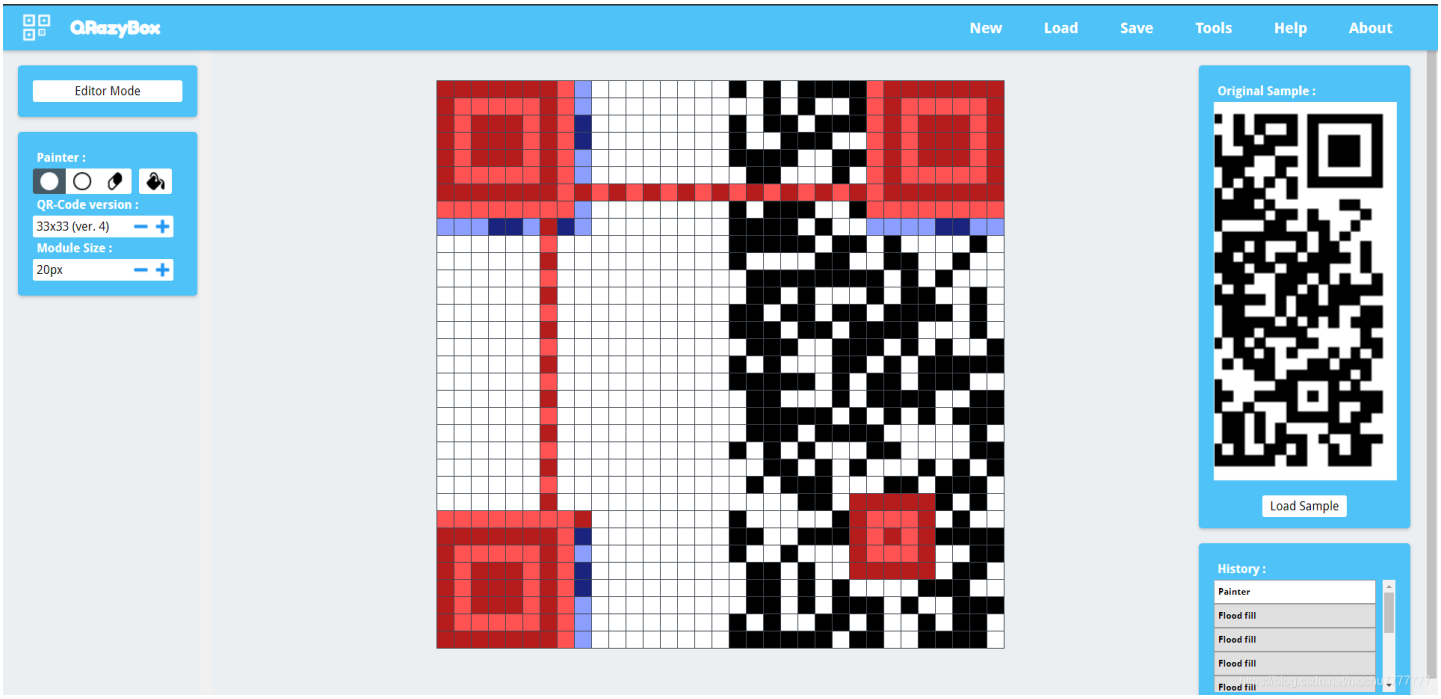

残缺二维码修复在线网站：https://merricx.github.io/qrazybox/

使用在线修复网站，二维码是`version4`版本，尺寸`33x33`，把这一半填好

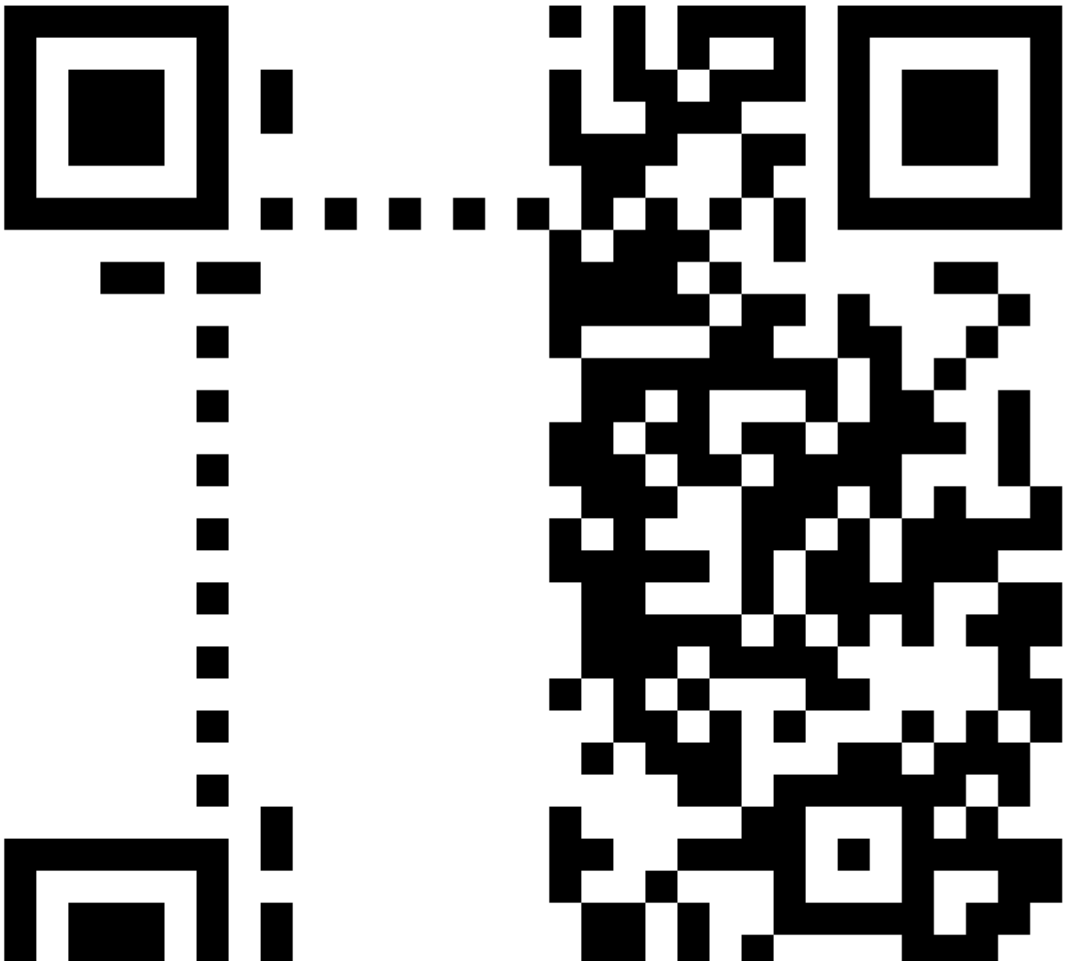




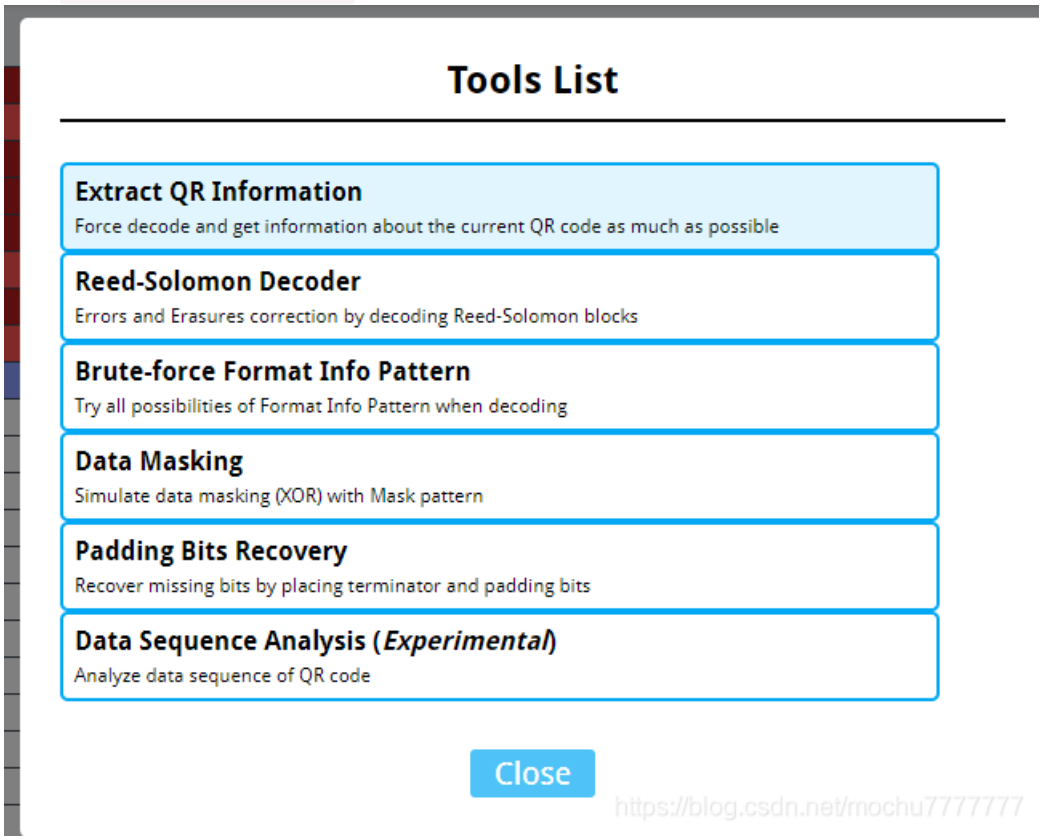

然后将灰色的地方使用油漆桶，填白

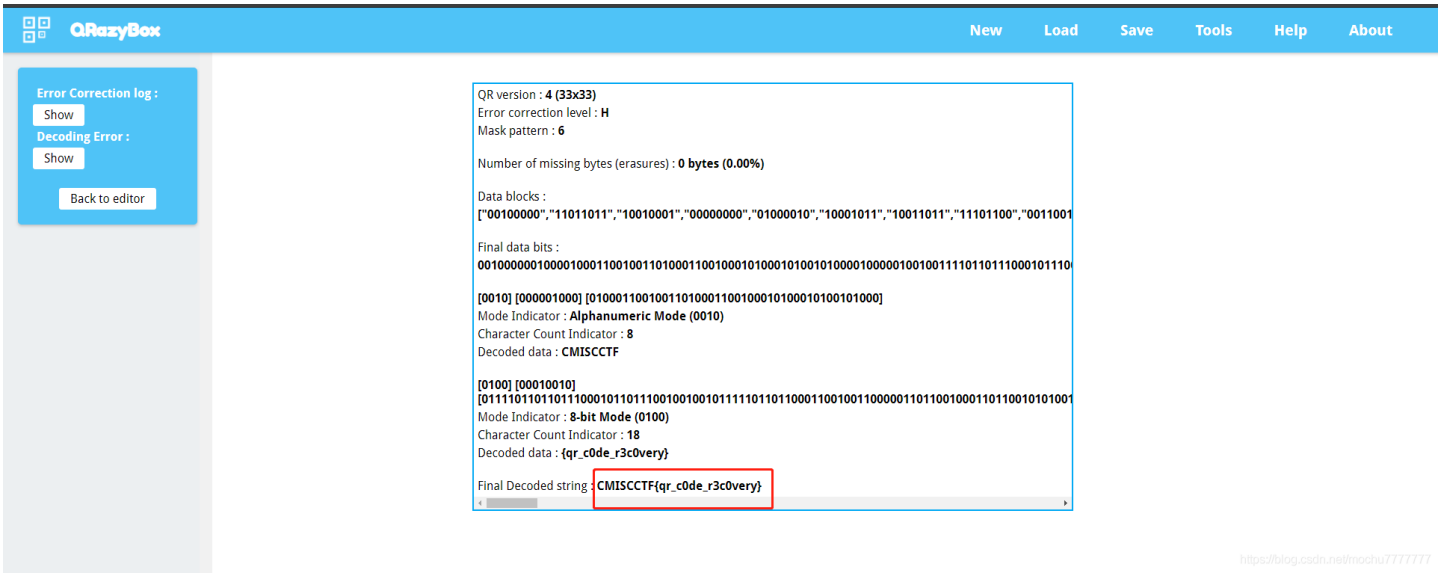



得到如下二维码



点击`Tools`识别，选择`Extract QR Information`

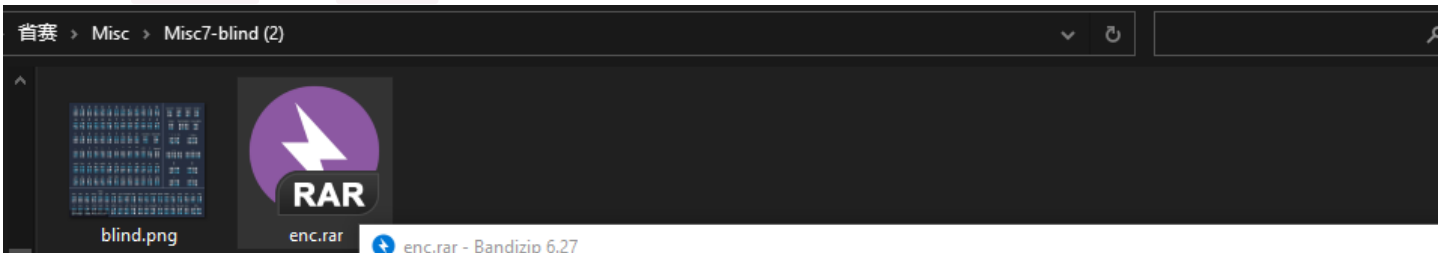


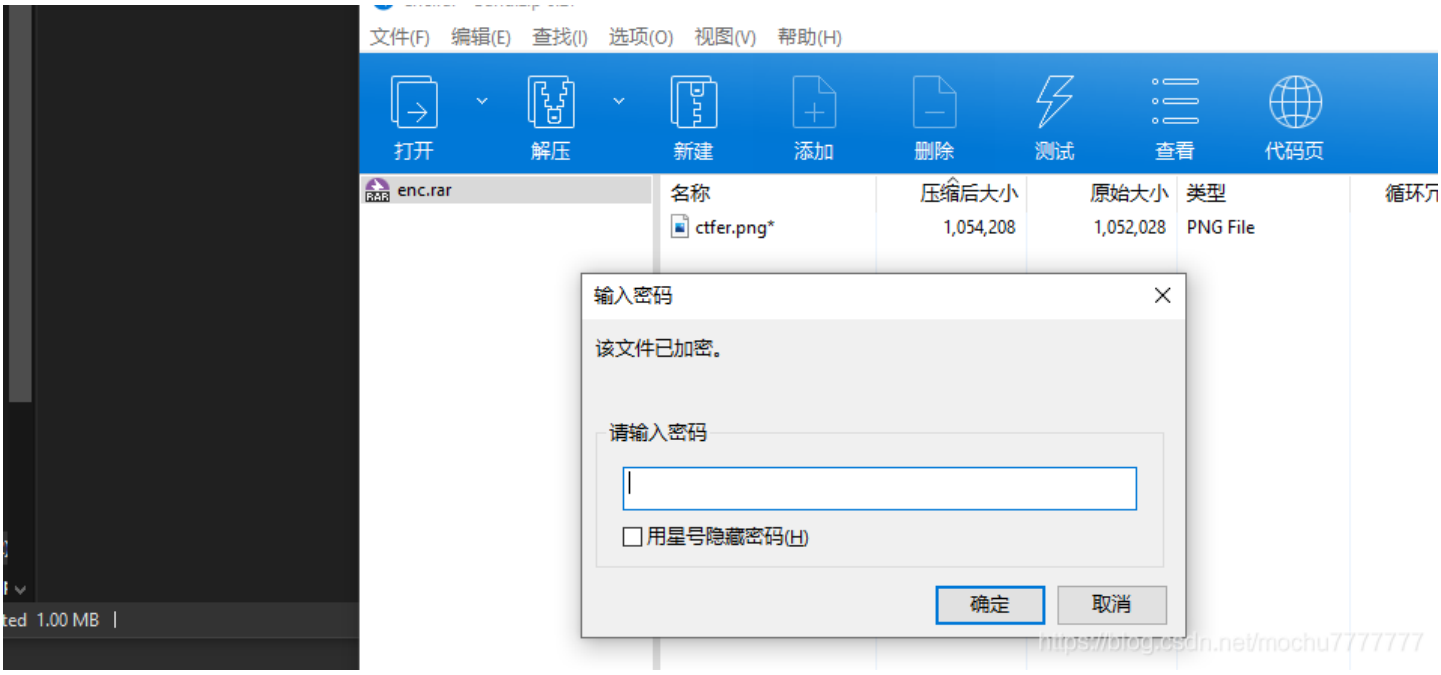




```
CMISCCTF{qr_c0de_r3c0very}
```

## Misc7-blind

题目一张图`blind.png`，一个`enc.rar`的压缩包，压缩包有密码，猜测解压密码信息要从图片上获得






`binwalk` 查看 `blind.png`

```
→  Desktop ls
blind.png  crc32  CTFer.png  output  tools
→  Desktop binwalk blind.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             PNG image, 1023 x 864, 8-bit/color RGBA, non-interlaced
168           0×A8            Zlib compressed data, compressed
1135463       0×115367        PNG image, 1023 x 864, 8-bit/color RGB, non-interlaced
2285037       0×22DDED        MySQL ISAM index file Version 5

→  Desktop
```



图片中还有一张图，使用 `foremost` 将这两张图单独分离出来

```
→  Desktop ls
blind.png  crc32  CTFer.png  tools
→  Desktop foremost blind.png
Processing: blind.png
|*|
→  Desktop ls
blind.png  crc32  CTFer.png  output  tools
→  Desktop cd output
→  output ls
audit.txt  png
→  output cd png
→  png ls
00000000.png  00002217.png
→  png binwalk 00000000.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             PNG image, 1023 x 864, 8-bit/color RGBA, non-interlaced
168           0×A8            Zlib compressed data, compressed

→  png binwalk 00002217.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             PNG image, 1023 x 864, 8-bit/color RGB, non-interlaced
1149574       0×118A86        MySQL ISAM index file Version 5
```

```
→ png
```

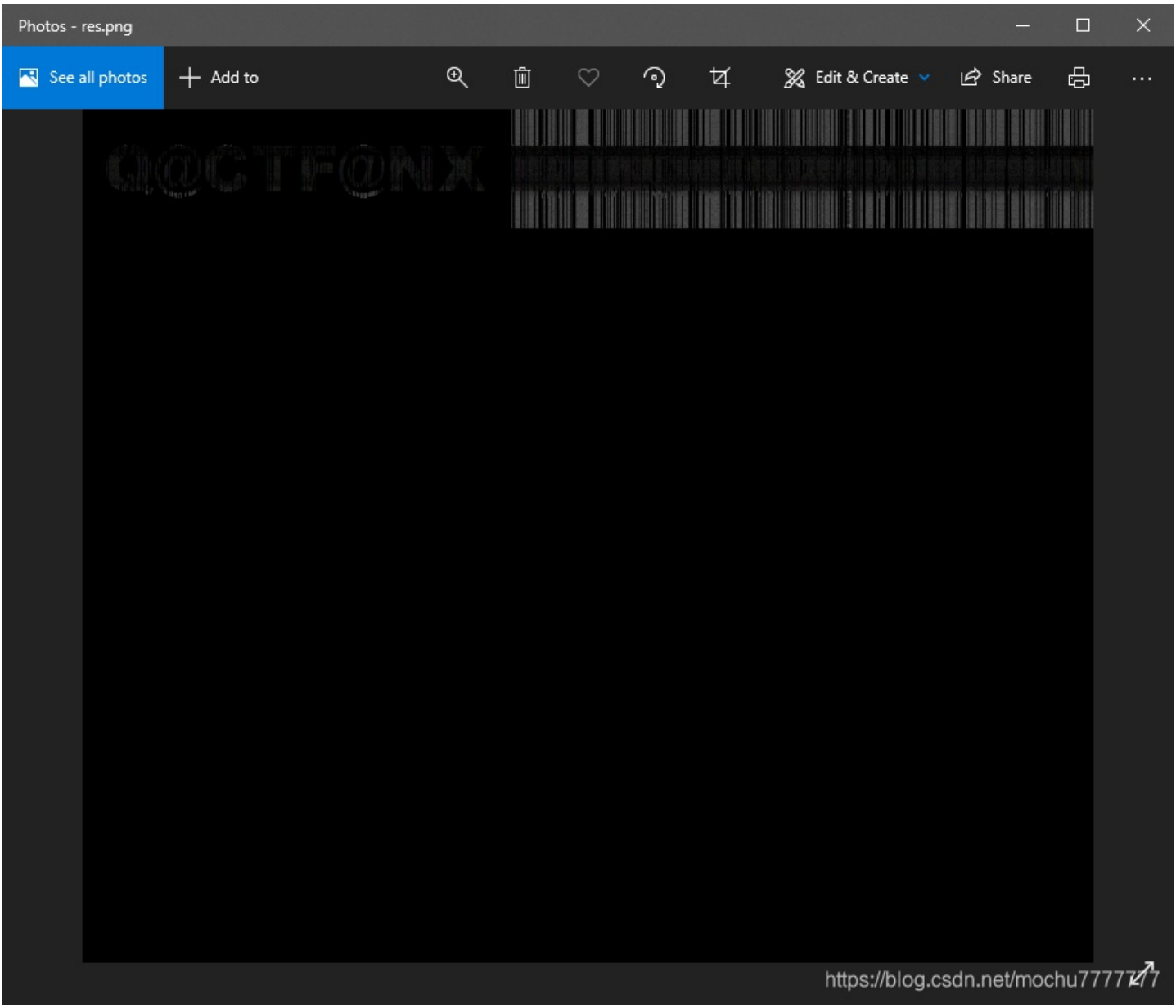


将得到的这两张图，使用以下脚本提取盲水印，根据题目名`blind`也可以猜测到这里是要使用`盲水印`

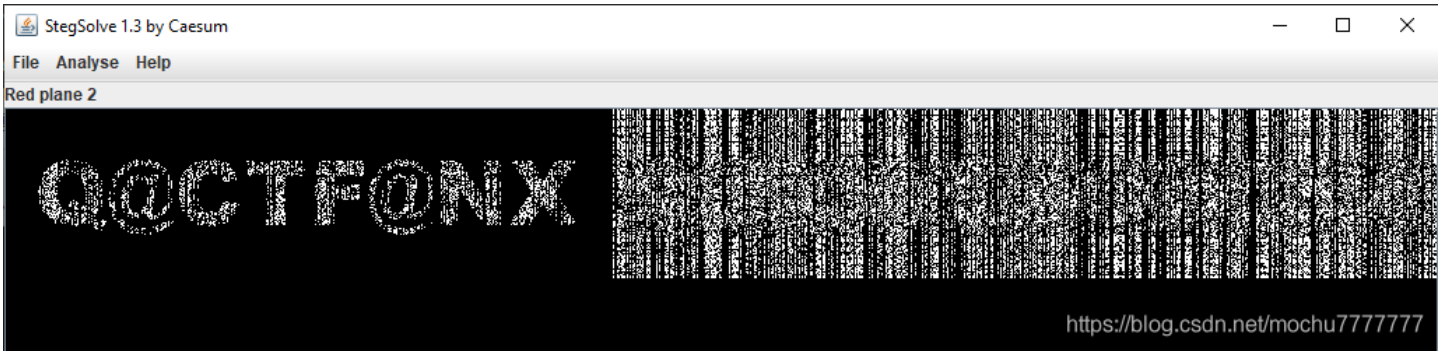
```python
# coding=utf-8
import cv2
import numpy as np
import random
import os
from argparse import ArgumentParser
ALPHA = 5


def build_parser():
    parser = ArgumentParser()
    parser.add_argument('--original', dest='ori', required=True)
    parser.add_argument('--image', dest='img', required=True)
    parser.add_argument('--result', dest='res', required=True)
    parser.add_argument('--alpha', dest='alpha', default=ALPHA)
    return parser


def main():
    parser = build_parser()
    options = parser.parse_args()
    ori = options.ori
    img = options.img
    res = options.res
    alpha = options.alpha
    if not os.path.isfile(ori):
        parser.error("original image %s does not exist." % ori)
    if not os.path.isfile(img):
        parser.error("image %s does not exist." % img)
    decode(ori, img, res, alpha)


def decode(ori_path, img_path, res_path, alpha):
    ori = cv2.imread(ori_path)
    img = cv2.imread(img_path)
    ori_f = np.fft.fft2(ori)
    img_f = np.fft.fft2(img)
    height, width = ori.shape[0], ori.shape[1]
    watermark = (ori_f - img_f) / alpha
    watermark = np.real(watermark)
    res = np.zeros(watermark.shape)
    random.seed(height + width)
    x = range(height / 2)
    y = range(width)
    random.shuffle(x)
    random.shuffle(y)
    for i in range(height / 2):
        for j in range(width):
            res[x[i]][y[j]] = watermark[i][j]
    cv2.imwrite(res_path, res, [int(cv2.IMWRITE_JPEG_QUALITY), 100])


if __name__ == '__main__':
    main()
```

```
python2 .\blind.py --original 1.png --image 2.png --result res.png
```

`res.png`




看不太清楚，用`stegslove`调一下通道即可看清楚




解压密码：Q@CTF@NX

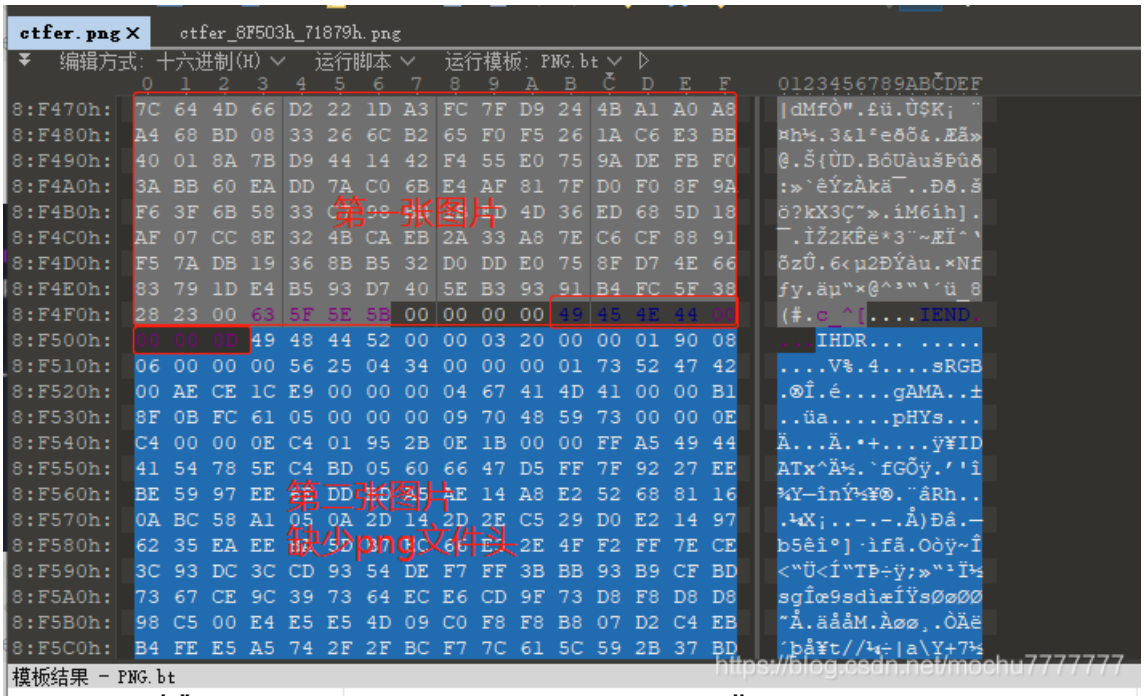
解压得到`CTFer.png`


老样子，`binwalk`先上

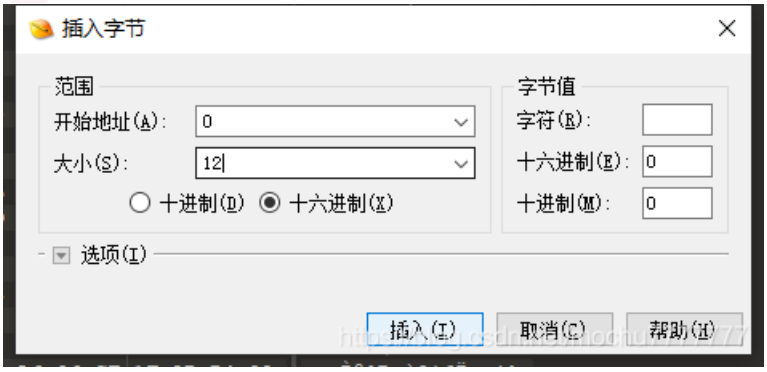
```
→  Desktop ls
blind.png  crc32  CTFer.png  output  tools
→  Desktop binwalk CTFer.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 800 x 400, 8-bit/color RGBA, non-interlaced
91            0x5B            Zlib compressed data, compressed
587090        0x8F552         Zlib compressed data, compressed

→  Desktop
```



两个`png`图片结尾信息，加上题目提示中有提到文件头，猜测这里应该是第二张png图片的文件头被去掉了，需要修补png头

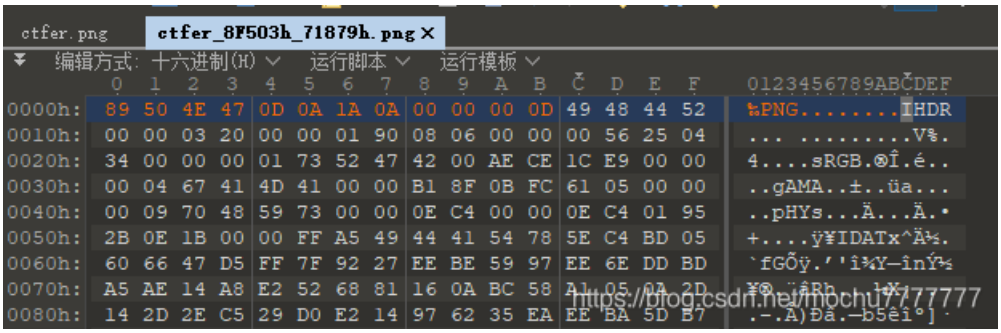



选中第二张图片的内容，`右键->选择->保存选择`存为`.png`后缀结尾即可，然后使用010 editor打开，在最开始的位置，`编辑->插入/覆盖->插入字节`，插入`12`个`00`




然后修改这个插入的`12个00`为png头即可

```
89 50 4E 47 0D 0A 1A 0A 00 00 00 0D
```

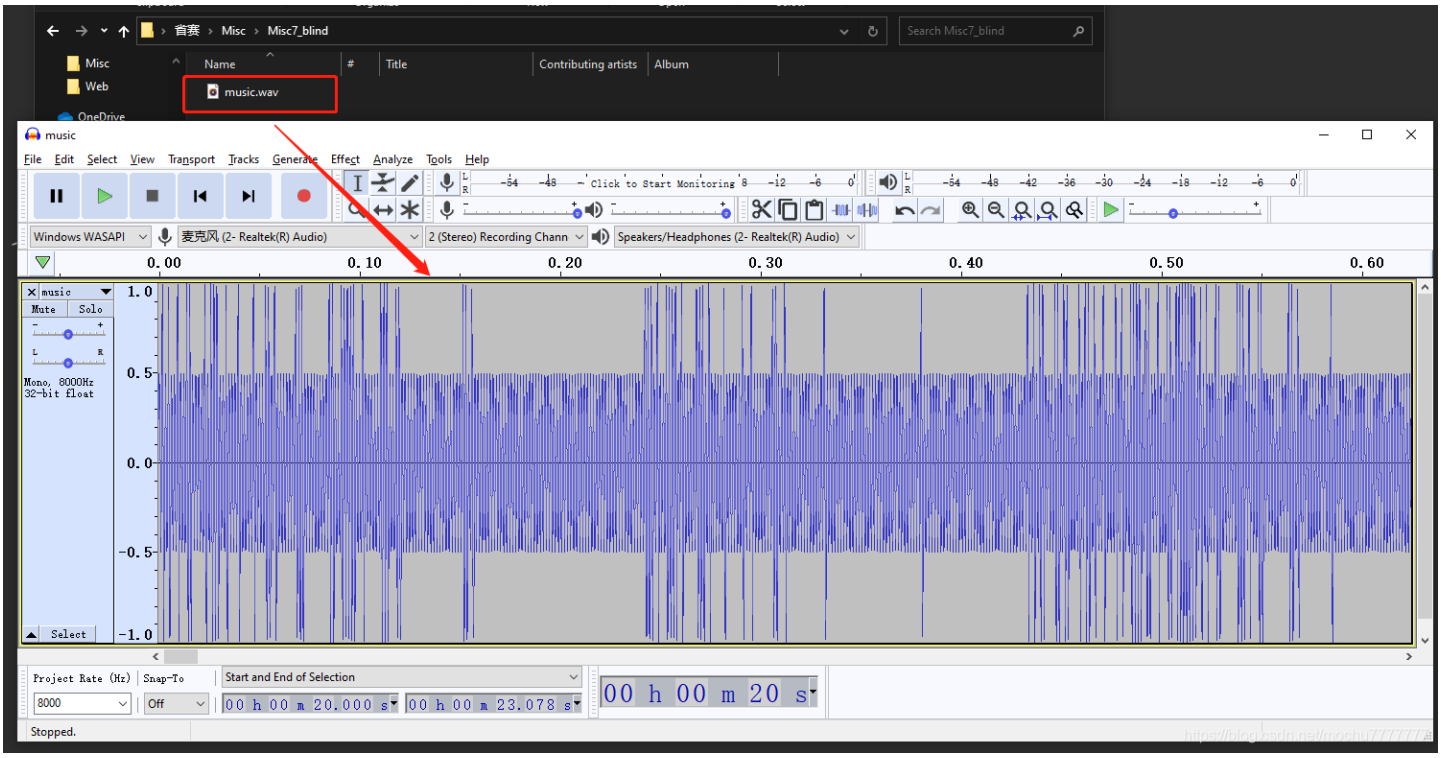



修改完后保存，即可得到flag


```
CMISCCTF{double_picture}
```

## Misc8-music

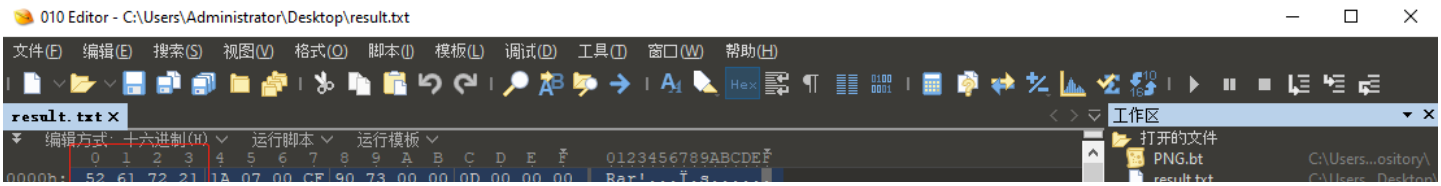
解压题目附件得到`music.wav`

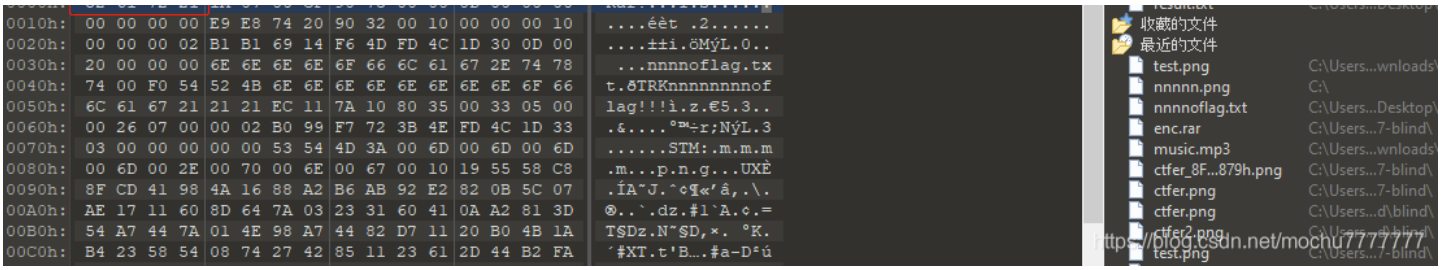



正放和反放没听出什么特征出来，看特征高低振幅明显，使用脚本对高低振幅转换为`01`，高振幅为`1`，低振幅为`0`，使用python脚本转换，并且将转换出来的`01`二进制文件流，八个一组转换为两位十六进制，然后将十六进制文件流写入文件内，脚本如下：

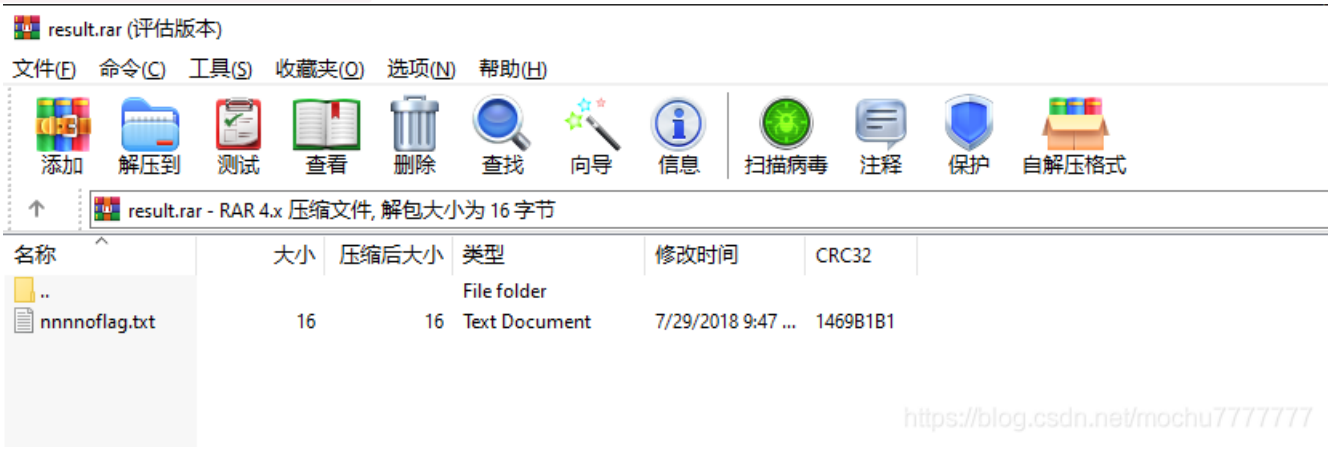
```python
import numpy as np
import struct
import wave
import re


def write_records(records, format, f):
    #Write a sequence of tuples to a binary file of structures.
    record_struct = Struct(format)
    for r in records:
        f.write(record_struct.pack(*r))

path = "./music.wav"
f = wave.open(path, "rb")
# 读取格式信息
# (nchannels, sampwidth, framerate, nframes, comptype, compname)
params = f.getparams()
nchannels, sampwidth, framerate, nframes = params[:4]
# 读取波形数据
str_data = f.readframes(nframes)
f.close()
#将波形数据转换为数组
wave_data = np.fromstring(str_data, dtype=np.short)
b = ''
# arr = [elem for elem in wave_data if elem >0]
max = 0
d = ''
for i in wave_data:
    if i <0:
        if max !=0:
            if max<25000:
                d +='0'
            else:
                d += '1'
                pass
        max = 0
    if max < i:
        max = i

print(d)
print("\n\n\n\n")
a = re.findall(r'.{8}',d)
hex_list=[]
for i in a:
    res = hex(int(i,2))
    hex_list.append(res)
print(hex_list)

with open("result.txt","wb") as f:
    for x in hex_list:
        s = struct.pack('B',int(x,16))
        f.write(s)
```

使用`010 editor`打开发现是`RAR`的头文件，修改文件后缀为`.rar`

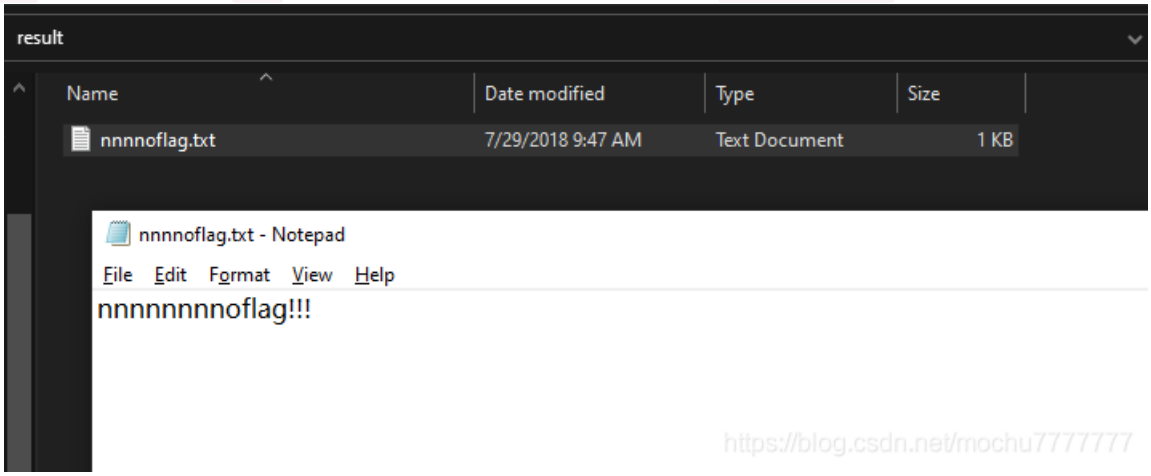





注：这里打开rar压缩包，请使用WinRAR



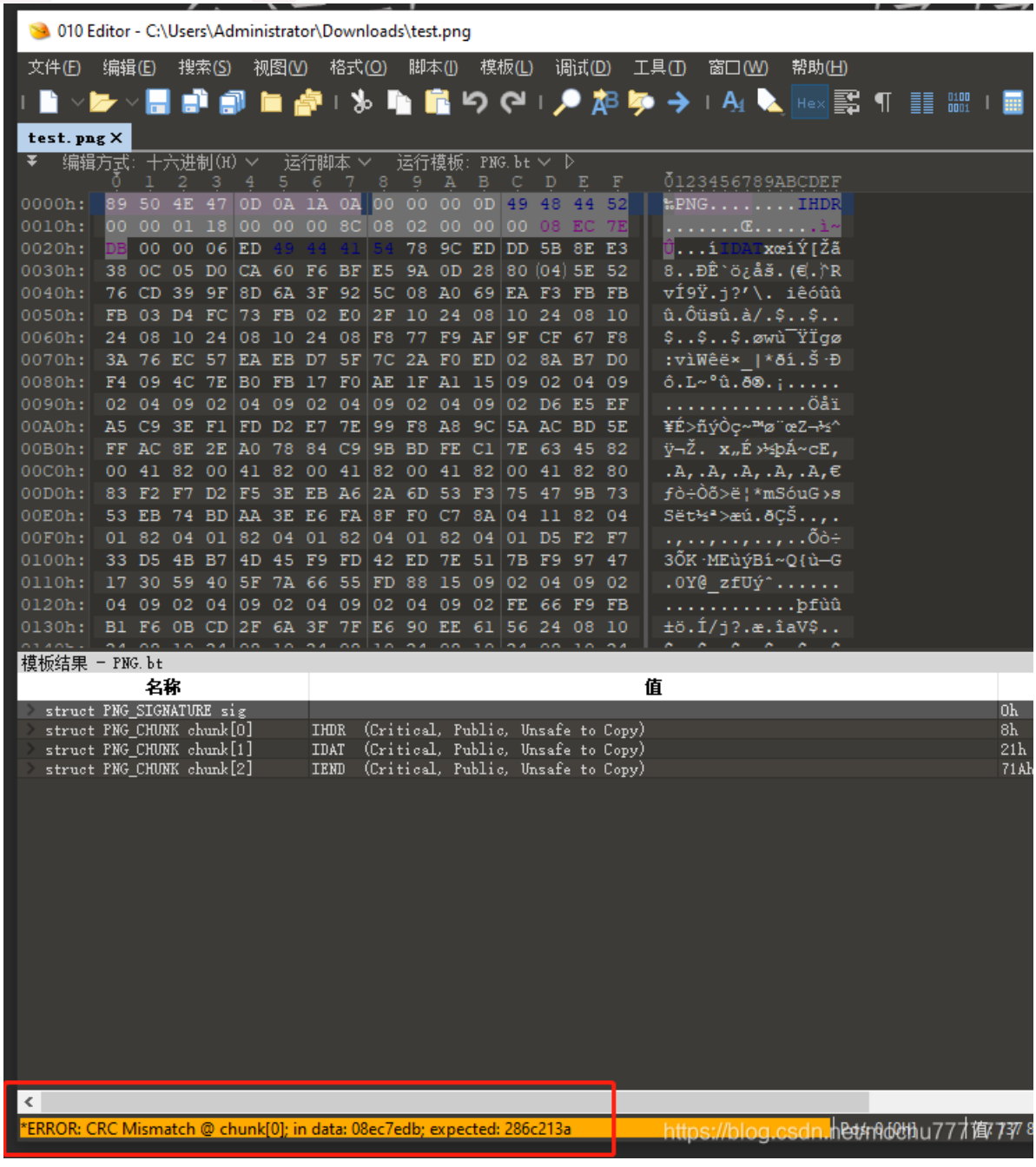

`nnnnoflag.txt` 并没有flag内容， `rar` 的压缩包中也无其他内容，猜测这里有 `NTFS文件流隐写`



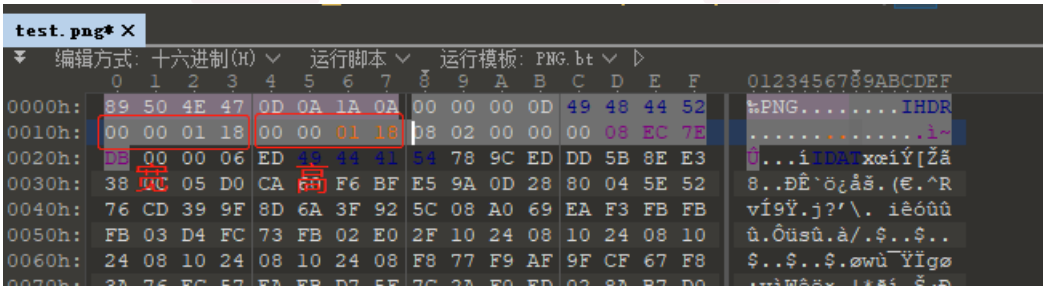

使用 `ntfsstreamseditor` 扫描 `nnnnoflag.txt` 的目录

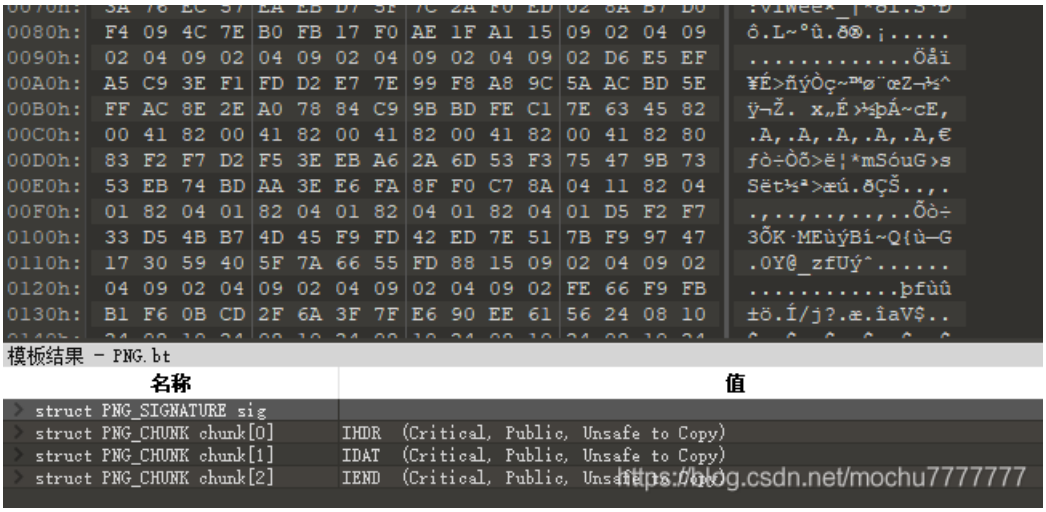

发现隐写了张图片，导出图片得到如下半张二维码

使用`01 editor`打开，左下角出现了`CRC`不匹配的错误




猜测图片有可能修改了宽高，使用`01 editor`修改十六进制高度和宽度相同为：`01 18`






保存，得到完整二维码

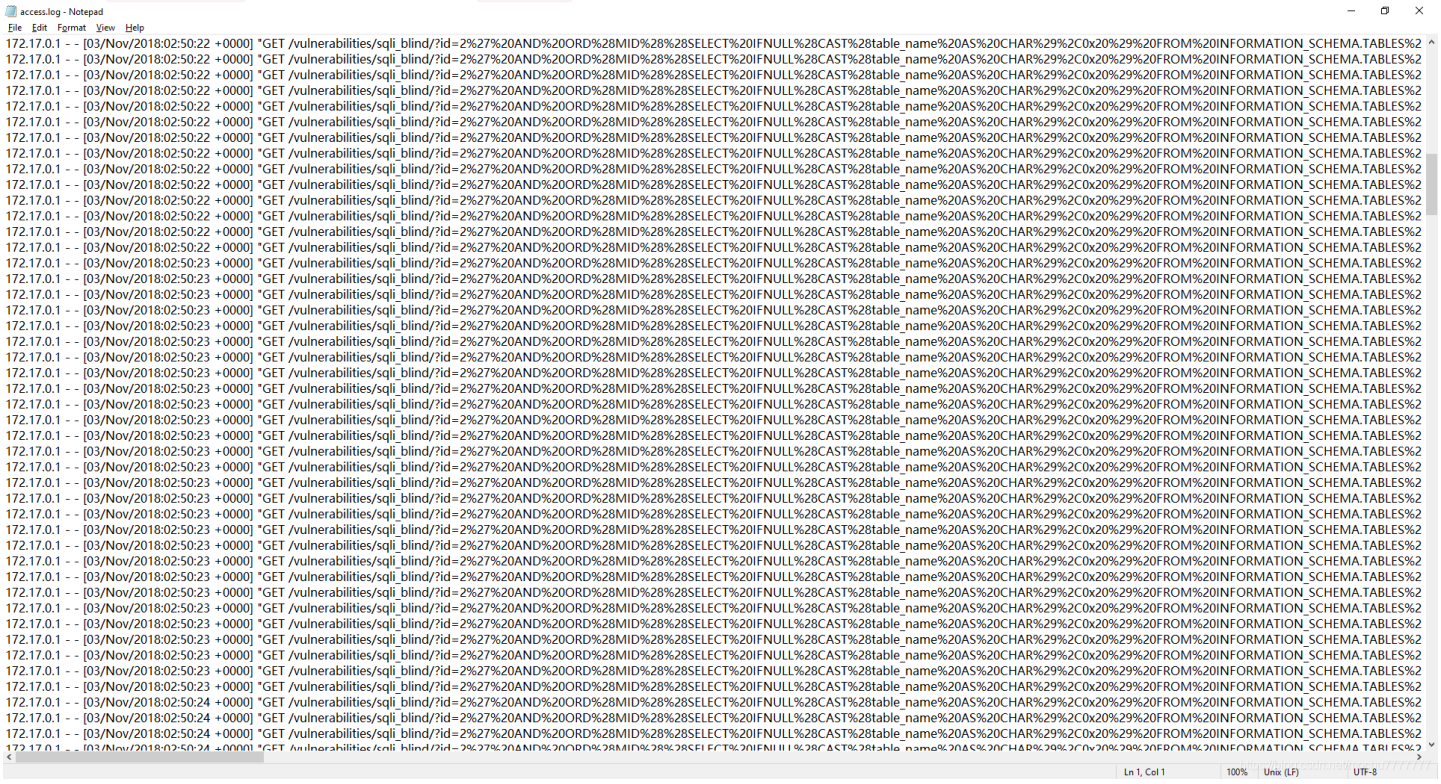

扫描即可得到flag

```
flag{4dcfda814ec9fd4761c1139fee3f65eb}
```

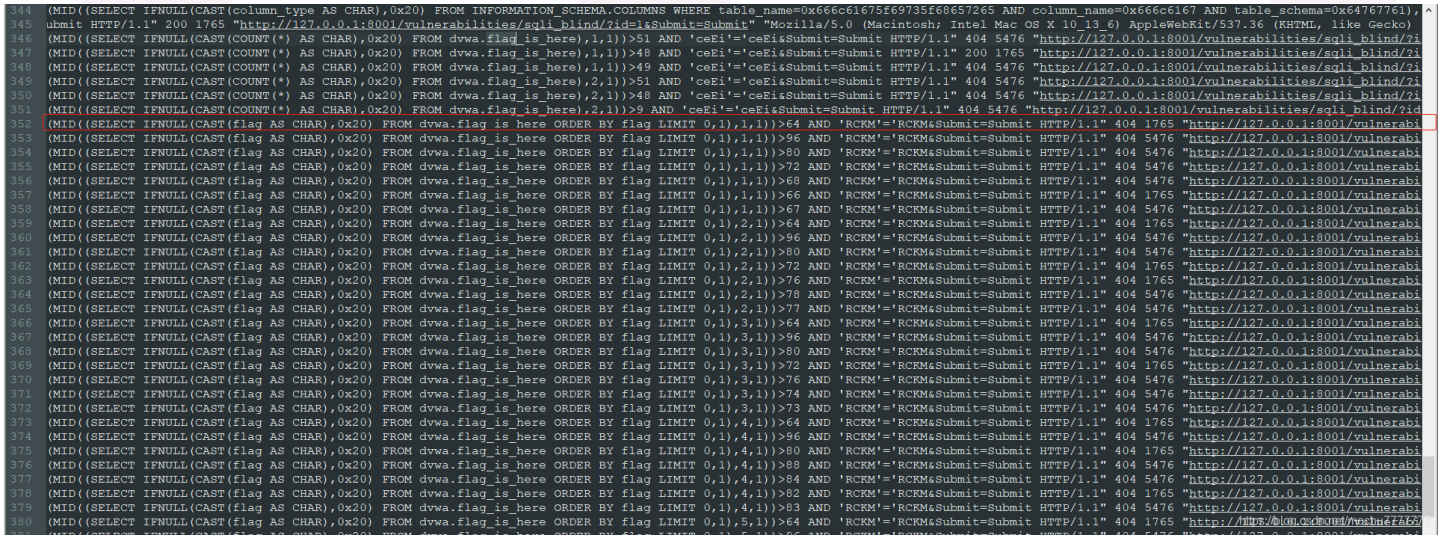
# Web

## Web1-Audit

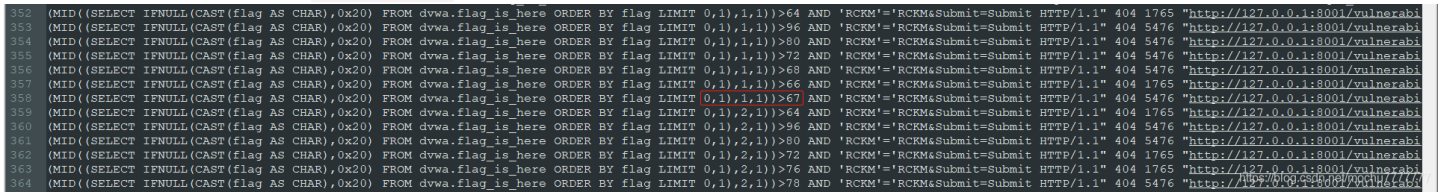
解压得到`access.log`日志文件，看内容是`sql盲注`的日志文件




找到查`flag字段内容`的payload，从这里开始注入出`flag字段内容`的每一个字符，为了直观的观看可以将整个内容`urldecode`



从这一行开始尝试通过比对`字符ascii码`来判断每一位字符



第1位ascii码：`67`

以此类推

以此类推

第2位ascii码：`77`
第3位ascii码：`73`
第4位ascii码：`83`
第5位ascii码：`67`
第6位ascii码：`67`
第7位ascii码：`84`
第8位ascii码：`70`
第9位ascii码：`123`
第10位ascii码：`98`
第11位ascii码：`111`
第12位ascii码：`114`
第13位ascii码：`105`
第14位ascii码：`110`
第15位ascii码：`103`
第16位ascii码：`95`
第17位ascii码：`97`
第18位ascii码：`117`
第19位ascii码：`100`
第20位ascii码：`105`
第21位ascii码：`116`
第22位ascii码：`125`

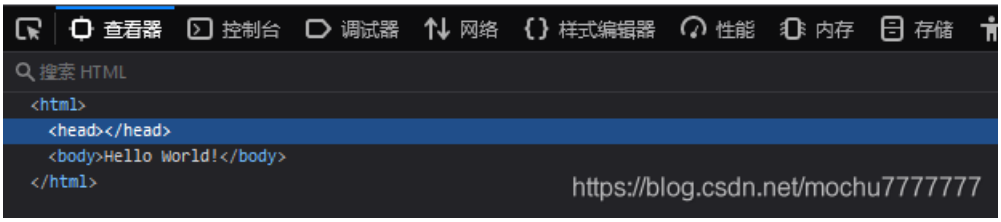
将每一位逐个转为字符得到如下：

```
CMISCCTF{boring_audit}
```

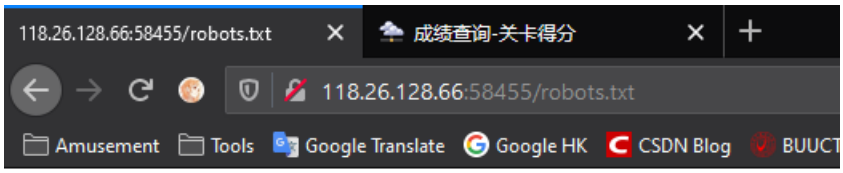
## Web2-scanner

除了Hello World!啥也看到

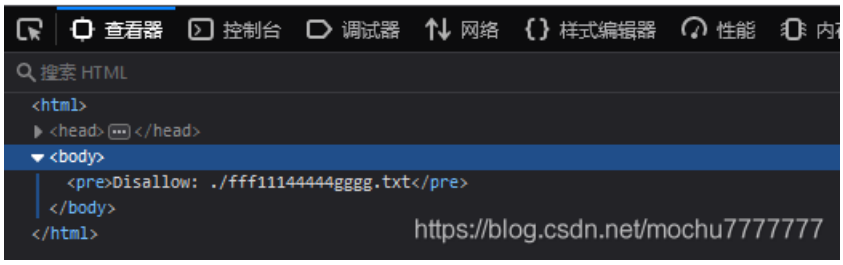

Hello World!



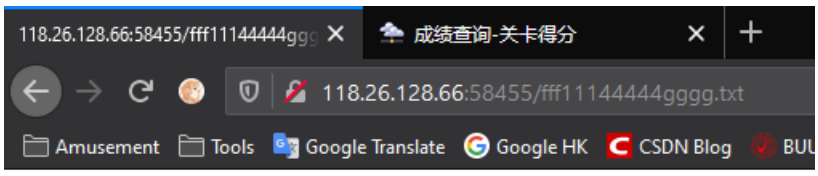

查看`robots.txt`



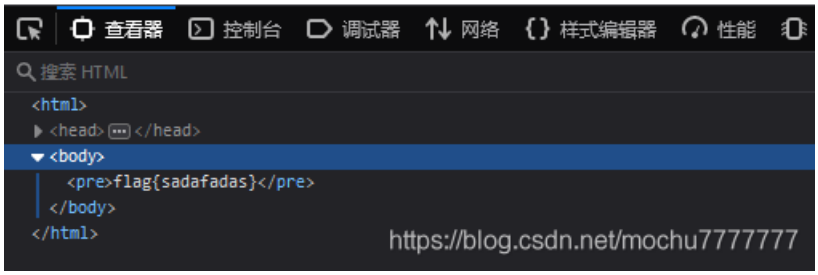

```
Disallow: ./fff11144444gggg.txt
```




访问`fff11144444gggg.txt`




```
flag{sadafadas}
```




`flag{sadafadas}`

## Web3-greatctf

## Web4-Admin

2019CISCN 华北赛区半决赛 Day1 Web3
http://blog.sgdream.cn/2019/06/09/ctf-2019-%E5%8D%8E%E5%8C%97%E8%B5%9B%E5%8C%BA%E5%8D%8A%E5%86%B3%E8%B5%9B%E8%AE%B0%E5%BD%95-web/

https://www.dazhuanlan.com/2019/10/18/5da8af7e5119a/

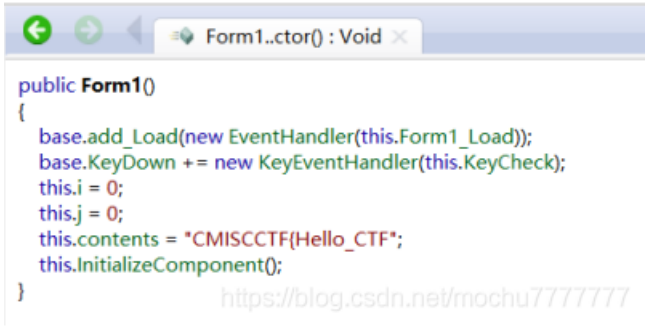
https://xz.aliyun.com/t/5383#toc-3

# Reverse

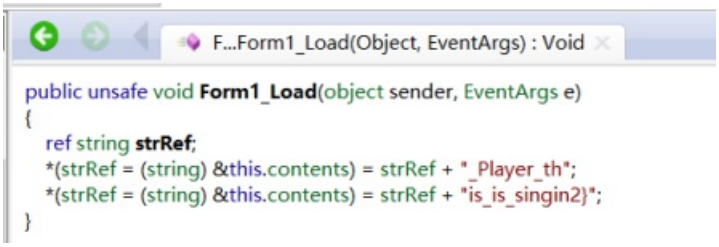
## Reverse1-Babyre

检查`exe`，发现是`.NET`，直接反编译

第一部分flag在构造函数中




其余的flag在加载的函数中




```
CMISCCTF{Hello_CTF_Player_this_is_singin2}
```

## Reverse2-Check

`Angr +符号执行`

```c
#define MAX_SIZE 0x40
#include <stdio.h>
#include <stdlib.h>

unsigned char check[] = {
  0x03, 0x12, 0x1a, 0x17, 0x0a, 0xec, 0xf2, 0x14, 0x0e, 0x05, 0x03, 0x1d,
  0x19, 0x0e, 0x02, 0x0a, 0x1f, 0x07, 0x0c, 0x01, 0x17, 0x06, 0x0c, 0x0a,
  0x19, 0x13, 0x0a, 0x16, 0x1c, 0x18, 0x08, 0x07, 0x1a, 0x03, 0x1d, 0x1c,
  0x11, 0x0b, 0xf3, 0x87, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x05
};


int reverse(unsigned char *a1) {
  int k; // [rsp+0h] [rbp-18h]
  int j; // [rsp+4h] [rbp-14h]
  int i; // [rsp+8h] [rbp-10h]

  for ( i = 0; i < MAX_SIZE; ++i )
    a1[i] += 5;
  for ( j = 0; j < MAX_SIZE - 1; ++j )
    a1[j] ^= a1[j + 1];
  for ( k = 0; ; ++k )
  {
    if ( k >= MAX_SIZE )
      return 1;
    if ( check[k] != a1[k] )
      break;
  }
  return 0LL;
}


int main(int argc, char *argv[]) {
        char flaggie[0x50];

        read(0, flaggie, 0x40);
    return reverse(flaggie);
}
```

使用以下命令编译为二进制文件

```
gcc source.c -no-pie -o flag
```

```python
import angr

p = angr.Project('./flag')
pg = p.factory.simulation_manager()
# find is the addr for the "good case" we want to reach
# avoid is the "bad case"
e = pg.explore(find=0x0040120d, avoid=0x0040123f)
if len(e.found) > 0:
print(e.found[0].posix.dumps(0))
```

```
>>> bytes(data)
b'CMISCCTF{machine_agnostic_that_not_easy}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
>>>
```

```
CMISCCTF{machine_agnostic_that_not_easy}
```

## Reverse3-Crackme

反编译后发现用户名和密码进行计算产生一个值 a1, a1进行接下来的计算
找到关键计算部分：

```
v3 = 31145713934493368781LL;
v4 = 3609637387099048214LL;
v5 = 6493761806479585431LL;
v6 = 16916018;
v7 = 2075;
v8 = 16;
for ( i = 0; i <= 30; ++i )
  putchar(a1 ^ *((char *)&v3 + i));
```

其中这几个数据类型识别有问题，正常应该是char数组
转换后依次尝试a1的值，找到一个这样的字符串：`FTCCSIMC_uoy_od{dna_prublos_ev}`
将该字符串倒序，分组、拼接即得flag：

```
CMISCCTF{do_you_burp_and_solve}
```

## Reverse4-oplog

反编译，发现`smc`

取`code[0x20:]`重新反编译看到正常`solidity`代码

```
from z3 import *

s = Solver()

a = Int('r1')
b = Int('r2')
c = Int('r3')

S1 = 0x88c218df8c5c25674af5808d963bfee9
S2 = 0xfa8cca1bced017e0ab064d4844c3020b
S3 = 0xe0ac283049469716cebd61a5b97b8bef
s.add(r1 > 0)
s.add(r2 > 0)
s.add(r3 > 0)
s.add(r1 < m1)
s.add(r2 < m2)
s.add(r3 < m3)

x1 = a * 0xd062 + b * 0x37b9 + c * 0xcc13

x2 = a * 0xa4fb + b * 0xa0a5 + c * 0x2fca

x3 = a * 0x8f9b + b * 0x9805 + c * 0xa6a0

mod = 0x80000000000000000000000000000000

v1 = 23579977885348111403331663368091779157240 20
v2 = 94024083436562980853861433269689272115769
v3 = 76867657257233810311465466602503314032464 17

key = (14678491206170330851881690558556870568208252 % mod) ^ v1
```

```
s.add((v1 ^ key) == (x1 % mod))
s.add((v2 ^ key) == (x2 % mod))
s.add((v3 ^ key) == (x3 % mod))

assert s.check() == sat

a = s.model()[r1].as_long()
b = s.model()[r2].as_long()
c = s.model()[r3].as_long()

from functools import reduce

def egcd(a, b):
    if 0 == b:
        return 1, 0, a
    x, y, q = egcd(b, a % b)
    x, y = y, (x - a // b * y)
    return x, y, q


def chinese_remainder(pairs):
    mod_list, remainder_list = [p[0] for p in pairs], [p[1] for p in pairs]
    mod_product = reduce(lambda x, y: x * y, mod_list)
    mi_list = [mod_product//x for x in mod_list]
    mi_inverse = [egcd(mi_list[i], mod_list[i])[0] for i in range(len(mi_list))]
    x = 0
    for i in range(len(remainder_list)):
        x += mi_list[i] * mi_inverse[i] * remainder_list[i]
        x %= mod_product
    return x

val = chinese_remainder([(m1, r1), (m2, r2), (m3, r3)])
print(bytes.fromhex(hex(val)[2:]))
```

```
flag{wuhan_v3r9_g009_s4y_w3jj_8}
```

# Pwn

## Pwn1_cmcc_stack

无法复现了，就帖个原题

https://blog.csdn.net/arttnba3/article/details/108068170

## Pwn2_pwn_canary

同上

https://blog.csdn.net/arttnba3/article/details/108068170

# Crypto

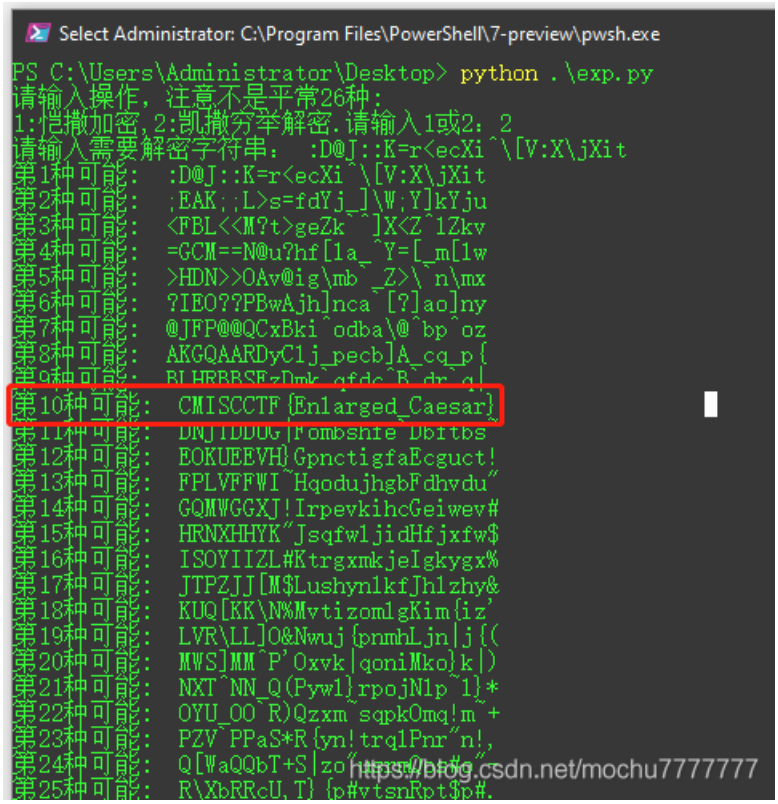
## Crypto1-Round

题目是如下字符串

```
:D@J::K=r<ecXi^\[V:X\jXit
```

凯撒密码

```python
def change(c,i):
    num=ord(c)
    if(num>=33 and num<=126):
        num=33+(num+i-33)%(94)#126-33=93
    return chr(num)


def kaisa_jiAmi(string,i):
    string_new=''
    for s in string:
        string_new+=change(s,i)
    print(string_new)
    return string_new

#本题有种暴力解密感觉
def kaisa_jiEmi(string):
    for i in range(0,94):
        print('第'+str(i+1)+'种可能:',end='  ')
        #区别在于 string 是该对象原本就是字符串类型, 而 str()则是将该对象转换成字符串类型。
        kaisa_jiAmi(string,i)

#你要知道input输入的数据类型都是string
def main():
    print('请输入操作，注意不是平常26种:')
    choice=input('1:恺撒加密,2:凯撒穷举解密.请输入1或2：')
    if choice=='1':
        string=input('请输入需要加密字符串：  ')
        num=int(input('请输入需要加密的KEY：  '))
        kaisa_jiAmi(string,num)
    elif choice=='2':
        string=input('请输入需要解密字符串：  ')
        kaisa_jiEmi(string)
    else:
        print('输入错误，请重试')
        main()

if __name__=='__main__':
    main()
```




CMISCCTF{Enlarged_Caesar}