




2021第四届浙江省大学生网络与信息安全竞赛预赛部分 Writeup

原创

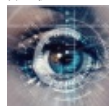
塞纳河畔的春水  于 2021-10-27 14:57:50 发布  43786  收藏 9

分类专栏: [CTF_Writeup](#) 文章标签: [python](#) [信息安全](#) [pycharm](#) [图像处理](#) [网络安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_42815161/article/details/120992830

版权



[CTF_Writeup](#) 专栏收录该内容

8 篇文章 2 订阅

订阅专栏

文章目录

Web

Checkin

RE

crackPYC

Crypto

Easy Railfence

MISC

你能找到flag吗?

qrimg

前言: 这次预赛感觉比去年难多了, 难题都没解出来, 还是太菜了orz。

Web

Checkin

纯签到题, 题目给了一个网址, 直接burpsuite抓包, 在响应头上拿到flag

Request
 GET / HTTP/1.1
 Host: 2d08121c-a342-49c8-97f8-87e9027f351f.zj-ctf.dasctf.com
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
 Connection: close
 Upgrade-Insecure-Requests: 1
 Cache-Control: max-age=0

Response
 ... Value
 ... 200 OK
 ... openresty/1.15.8.1
 ... Sat, 23 Oct 2021 04:59:58 GMT
 ... text/html; charset=UTF-8
 ... 23
 ... close
 ... PHP/7.2.30
 ... DASCTF{eefe178367a4906ec4fd10d326da0e8b}

So, Where is the flag?

Done CSDN @塞纳河畔的毒水 363 bytes | 35 millis

RE

crackPYC

题目提供python字节码，直接找到关键部分进行人工反编译，脚本如下：

```

"""
19      174 SETUP_LOOP          48 (to 224)
        176 LOAD_NAME            11 (range)
        178 LOAD_CONST          36 (32)
        180 CALL_FUNCTION        1
        182 GET_ITER
    >> 184 FOR_ITER            36 (to 222)
        186 STORE_NAME        12 (i)

20      188 LOAD_NAME            13 (ord)
        190 LOAD_NAME            4 (str)
        192 LOAD_NAME            12 (i)

    st[i] = ord(str[i])

194 BINARY_SUBSCR
        196 CALL_FUNCTION        1
        198 LOAD_NAME            10 (key)
        200 LOAD_NAME            12 (i)

202 LOAD_NAME            6 (len)
        204 LOAD_NAME            10 (key)
        206 CALL_FUNCTION        1

208 BINARY_MODULO
        210 BINARY_SUBSCR

```

```

212 BINARY_XOR
214 LOAD_NAME          9 (st)
216 LOAD_NAME         12 (i)
218 STORE_SUBSCR

st[i] ^ key[i % len(key)]

2          0 LOAD_CONST      1 (0)
          2 STORE_FAST       1 (num)

num = 0

3          4 SETUP_LOOP      42 (to 48)
          6 LOAD_GLOBAL        0 (range)
          8 LOAD_CONST        2 (8)
         10 CALL_FUNCTION      1
         12 GET_ITER
    >>   14 FOR_ITER        30 (to 46)
         16 STORE_FAST       2 (i)

for i in range(0,8)

4          18 LOAD_FAST        1 (num)
         20 LOAD_CONST        3 (7508399208111569251)
         22 BINARY_SUBTRACT
         24 LOAD_CONST        4 (4294967295)
         26 BINARY_MODULO
         28 STORE_FAST        1 (num)

num =(num - 7508399208111569251) % 4294967295

5          30 LOAD_FAST        0 (key)
         32 LOAD_METHOD        1 (append)
         34 LOAD_FAST        1 (num)
         36 LOAD_CONST        5 (24)
         38 BINARY_RSHIFT

        key.append(num >> 24)
"""
num = 0
key = []
for i in range(0, 8):
    num = (num - 7508399208111569251) % 4294967295
    key.append(num >> 24)
print(key)
# key = [40, 80, 121, 161, 202, 242, 27, 67]
text = [108, 17, 42, 226, 158, 180, 96, 115, 64, 24, 38, 236, 179, 173, 34, 22, 81, 113, 38, 215, 165, 135,
        97, 45, 254, 250, 172, 43, 62]
for i in range(len(text)):
    tmp = text[i] ^ key[i % 8]
    print(chr(tmp), end="")
# DASCTF{0hH_My_9Uy!_vou_D_1T_0^0}

```

Crypto

Easy Railfence

题目:

```
reetdrvhs0eutbftafmeon}linnd=a1c0h!gcedos{neuwkYav0ir0ceytounw
```

题目名提示栅栏，直接尝试栅栏编码解密，在12栏发现flag字样

```
12栏: rtntflag{YOucanc1imbsder0fenceeveny0ud0notevehuandhowitworks!}=}
```

The screenshot shows a web-based tool for decoding Rail Fence Ciphers. The interface includes a 'Recipe' section with a title 'Rail Fence Cipher Decode', a 'Key' input field containing '12', and an 'Offset' input field containing '0'. The 'Input' section contains the ciphertext: 'reetdrvhs0eutbftafmeon}linnd=a1c0h!gcedos{neuwkYav0ir0ceytounw'. The 'Output' section displays the result: 'rtntflag{YOucanc1imbsder0fenceeveny0ud0notevehuandhowitworks!}=}'. Metadata for the input and output is shown in the top right and bottom right corners, respectively.

尝试遍历offset无果

改到13栏，遍历offset，最终在offset=5处拿到flag

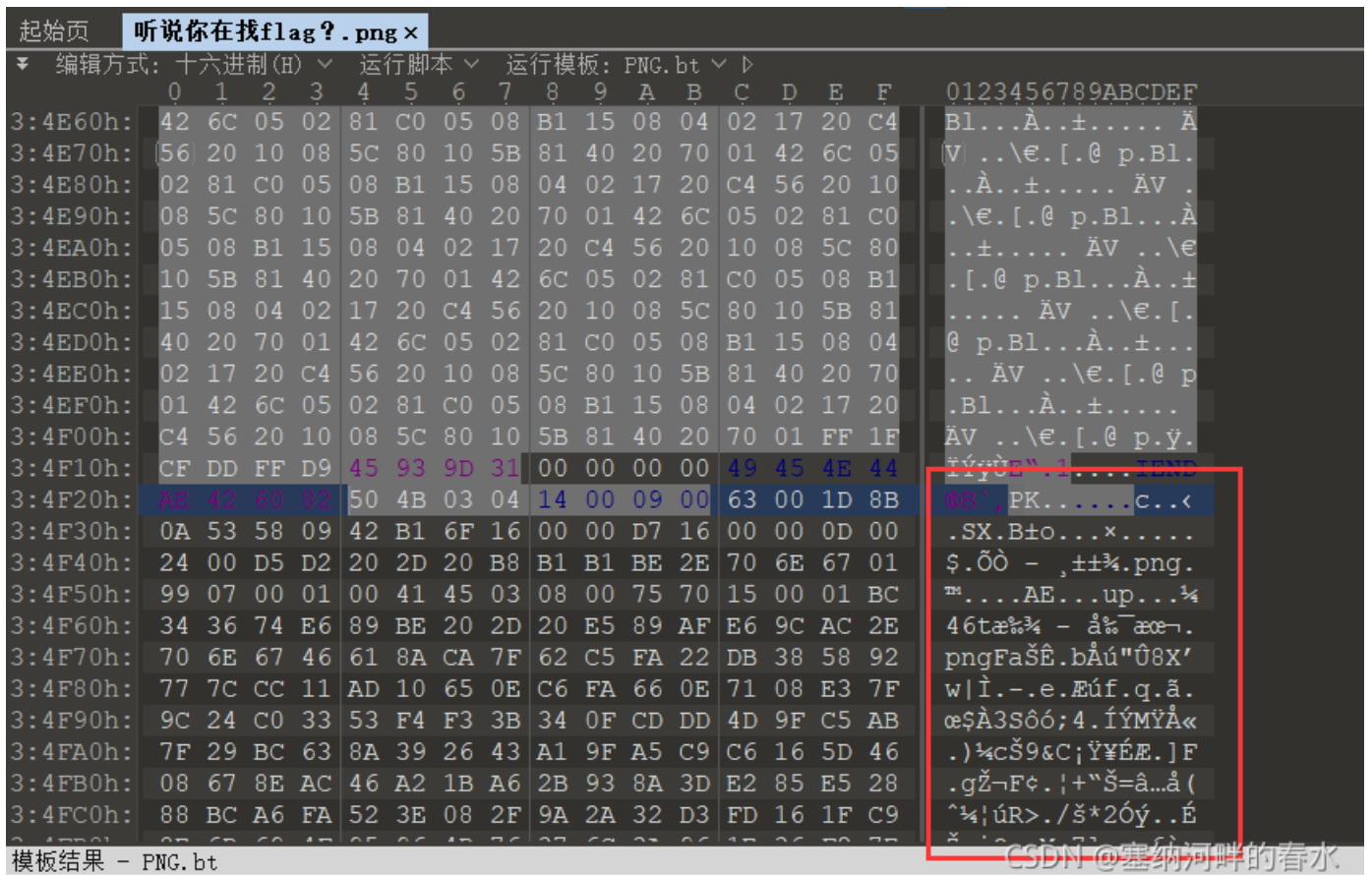
This screenshot shows the same tool with the 'Key' set to '13' and the 'Offset' set to '5'. The input remains the same ciphertext. The output now shows the flag: 'flag{YOucanc1imb0verthefenceeveny0ud0notunderstandhowitworks!}=}'. The metadata for the output shows a length of 63 and 1 line.

```
flag{YOucanc1imb0verthefenceeveny0ud0notunderstandhowitworks!}=}
```

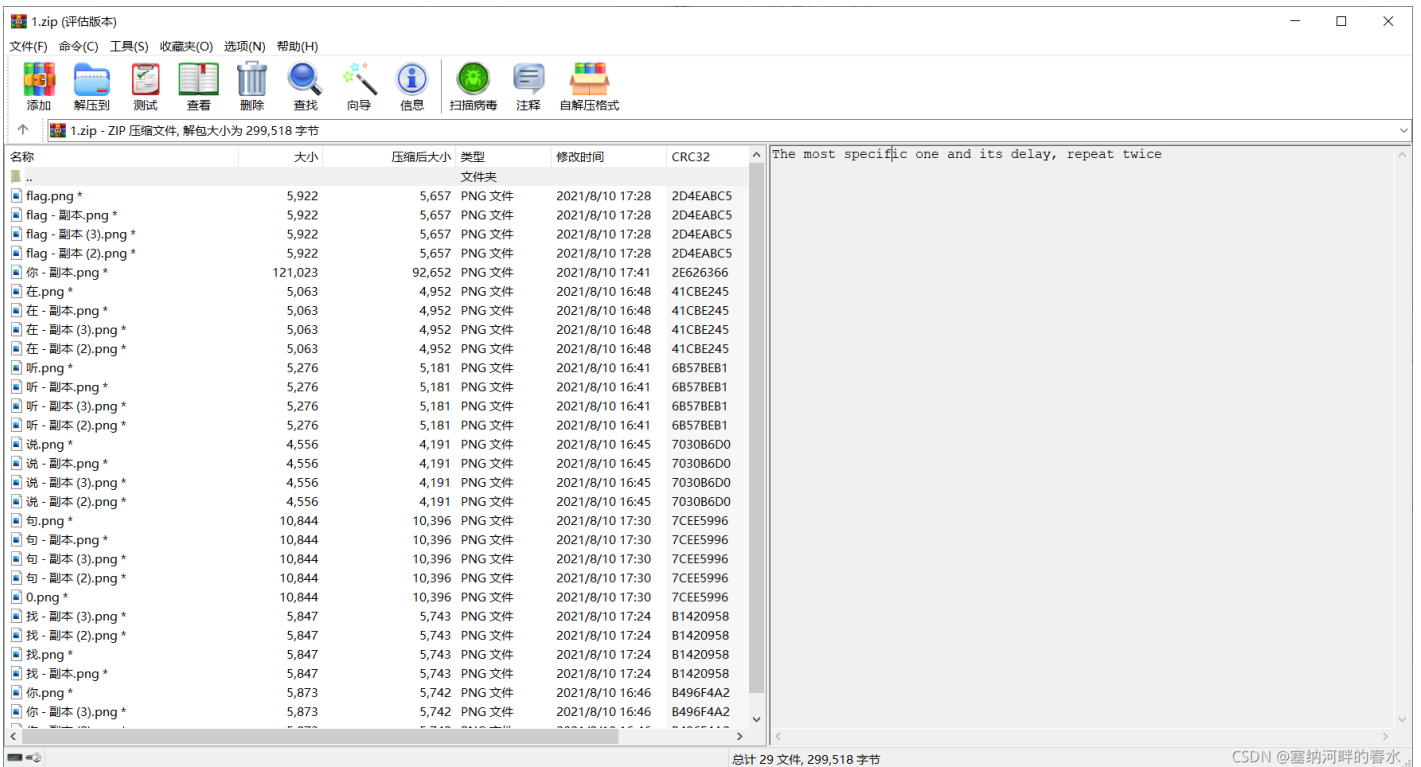
MISC

你能找到flag吗?

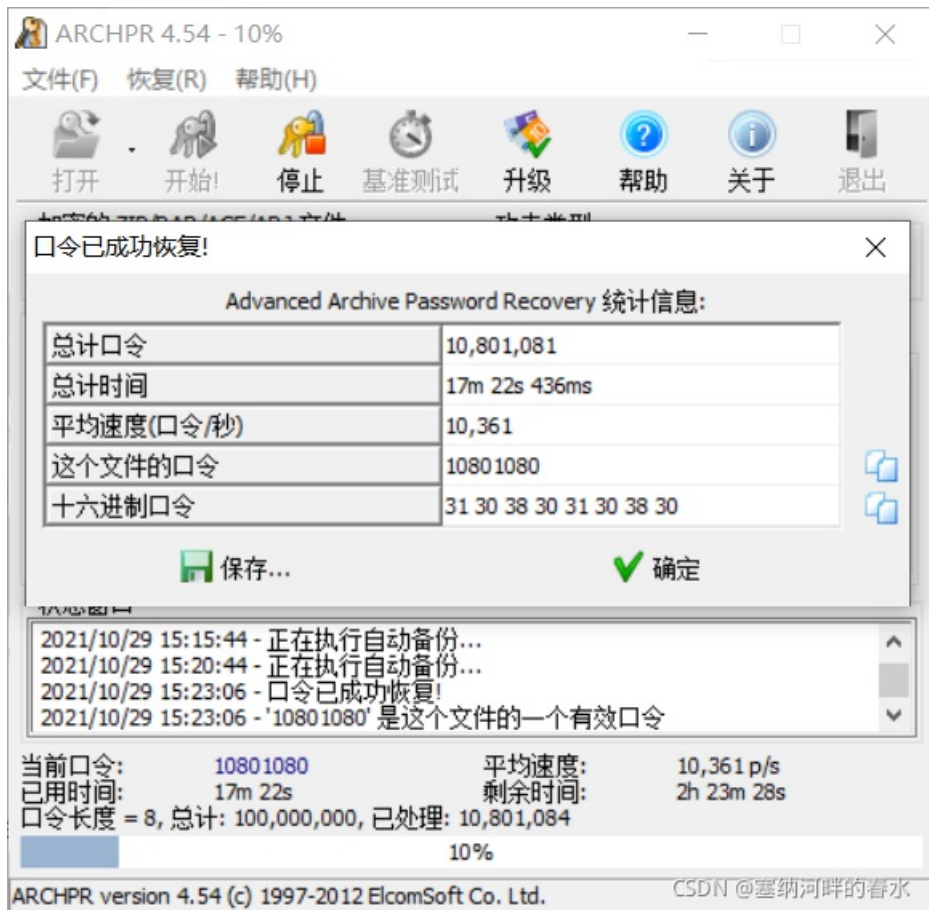
图片给了一张png图片，010Editor打开发现末尾存在压缩包，将压缩包分离。



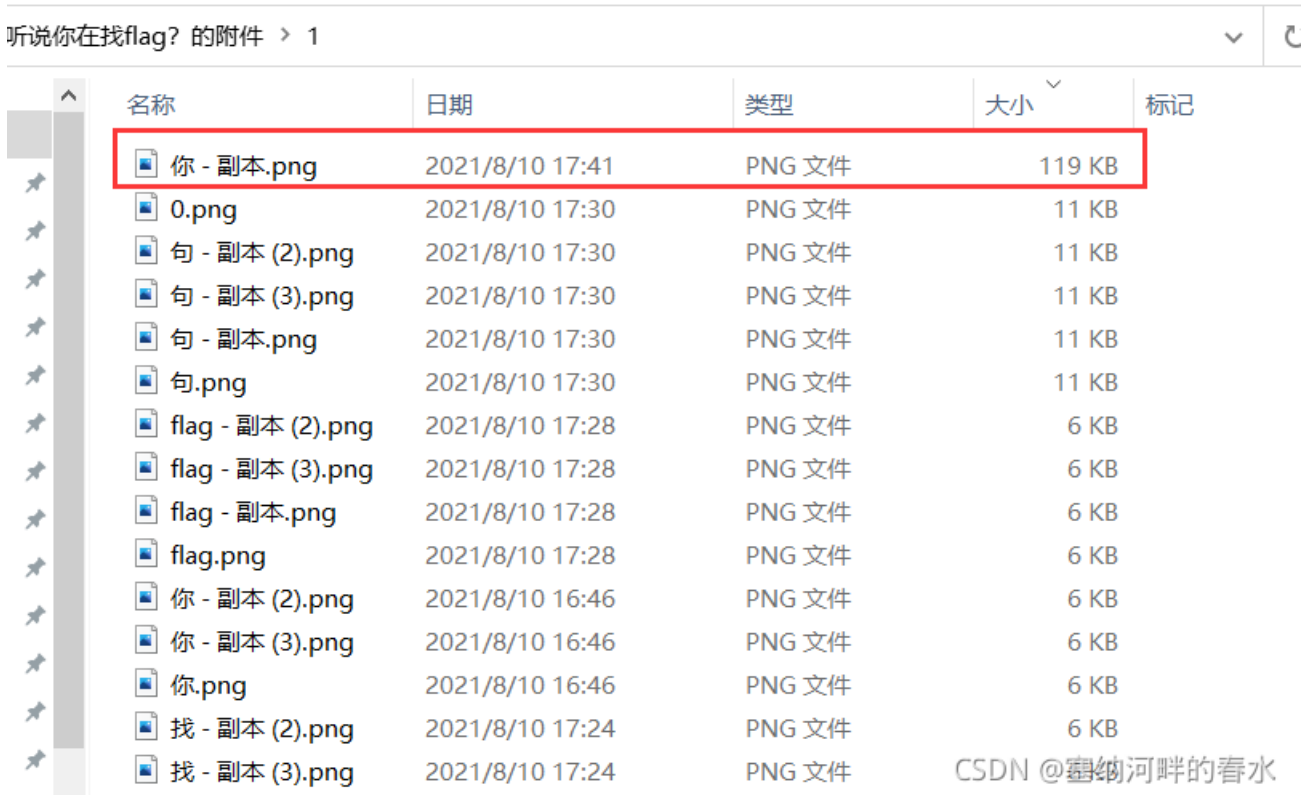
得到一个加密压缩包，里面含有大量图片。



找密码，fuzz一下没什么发现尝试纯数字爆破，十几分钟后，密码10801080



有密码后直接解压，发现有张图片特别大（没发现特别大，题目其实也给了hint，后面再说）。



010Editor打开，在最后发现一串base64图片编码。

```

起始页 你 - 副本.png x
编辑方式: 十六进制(H) 运行脚本 运行模板: PNG.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
1630h: 4E 6A C7 FE 60 54 23 08 21 44 95 51 64 2B 84 10 NjÇb`T#.!D•Qd+...
1640h: 11 90 6C 85 10 22 02 92 AD 10 42 44 40 B2 15 42 ..l....".'-.BD@².B
1650h: 88 08 48 B6 42 08 11 01 C9 56 08 21 22 20 D9 0A ^.HqB...ÉV.! " Ù.
1660h: 21 44 04 24 5B 21 84 88 80 64 2B 84 10 11 90 6C !D.[$[!,"^èd+,...l
1670h: 85 10 22 02 92 AD 10 42 44 40 B2 15 42 88 08 48 ...".'-.BD@².B^ .H
1680h: B6 42 08 11 01 C9 56 08 21 22 20 D9 0A 21 44 04 qB...ÉV.! " Ù.!D.
1690h: 24 5B 21 84 88 80 64 2B 84 10 11 90 6C 85 10 22 $[!,"^èd+,...l...."
16A0h: 02 92 AD 10 42 44 40 B2 15 42 88 08 48 B6 42 08 .'-.BD@².B^ .HqB.
16B0h: 11 01 C9 56 08 21 22 20 D9 0A 21 44 04 24 5B 21 ..ÉV.! " Ù.!D.[$[!
16C0h: 84 88 80 64 2B 84 10 11 90 6C 85 10 22 02 92 AD ..^èd+,...l....".'-
16D0h: 10 42 44 40 B2 15 42 88 AE E3 DC FF 01 75 1B 3D .BD@².B^@ãÛÿ.u.=
16E0h: 13 83 E9 F9 6F 00 00 00 00 49 45 4E 44 AE 42 60 .fèùo....IEND@B`
16F0h: 82 64 61 74 61 3A 69 6D 61 67 65 2F 70 6E 67 3B ,data:image/png;
1700h: 62 61 73 65 36 34 2C 69 56 42 4F 52 77 30 4B 47 base64,iVBORw0KG
1710h: 67 6F 41 41 41 41 4E 53 55 68 45 55 67 41 41 41 goAAAANSUHEUgAAA
1720h: 70 4D 41 41 41 47 51 43 41 59 41 41 41 42 2F 50 pMAAAGQCAYAAAB/P
1730h: 2B 6D 53 41 41 41 41 58 4E 53 52 30 49 41 72 +mSAAAAAXNSR0IAR
1740h: 73 34 63 36 51 41 41 41 41 52 6E 51 55 31 42 41 s4c6QAAAAArnQU1BA
1750h: 41 43 78 6A 77 76 38 59 51 55 41 41 41 41 4A 63 ACxjwv8YQUAAAAJc
1760h: 45 68 5A 63 77 41 41 45 6E 51 41 41 42 4A 30 41 EhZcwAAEnQAABJ0A
1770h: 64 35 6D 48 33 67 41 41 50 2B 6C 53 55 52 42 56 d5mH3gAAP+LSURBV
1780h: 48 68 65 37 4A 30 4A 76 48 62 56 2B 50 34 33 71 Hhe7J0JvHbV+P43q
1790h: 57 67 67 69 52 53 52 42 73 31 46 6B 71 4A 52 45 WggiRSR
CSDN@塞纳河畔的春水

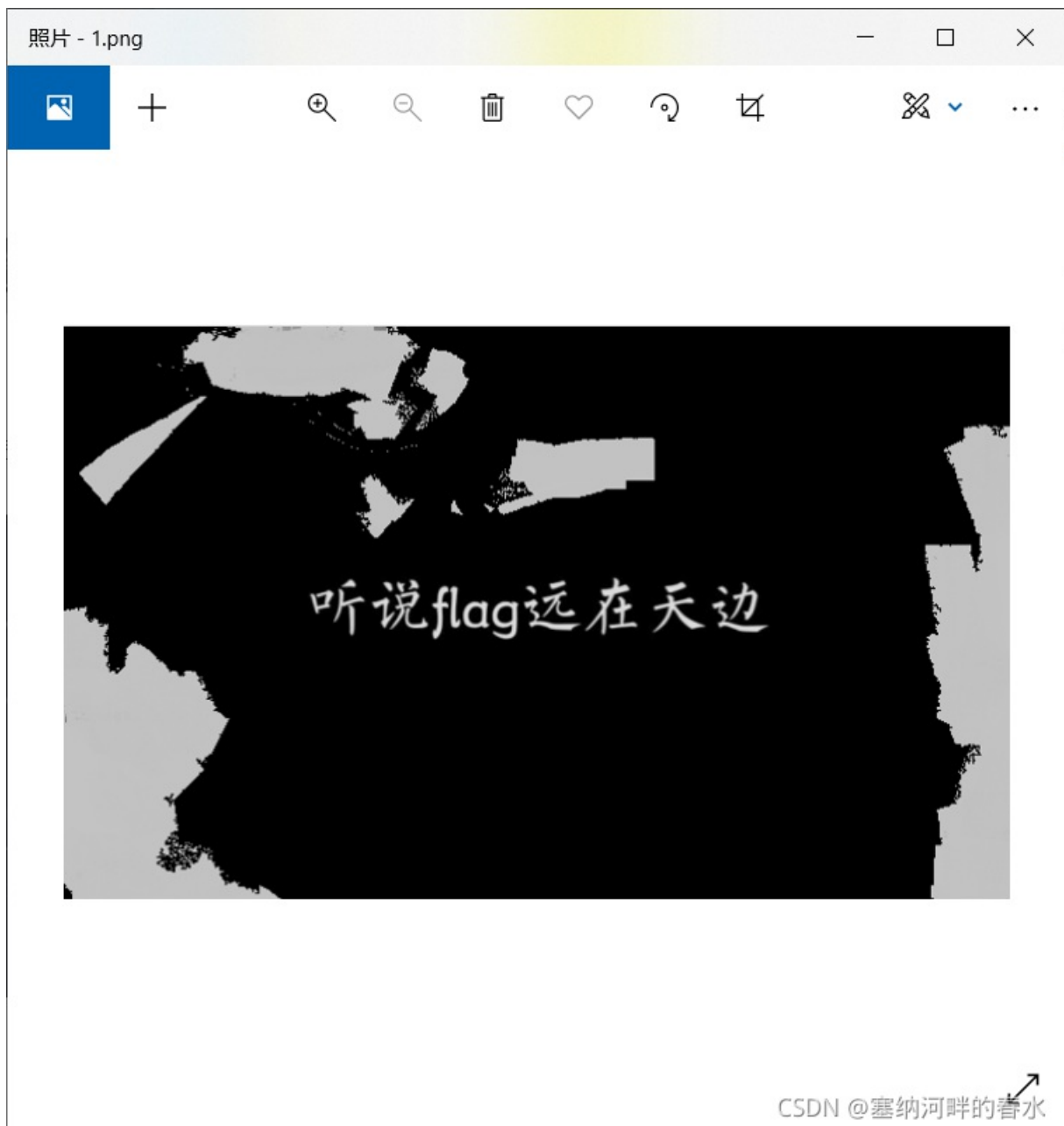
```

截取出base64后运行010Editor内置脚本编码。

```

你 - 副本.png x
编辑方式: 十六进制(H) 运行脚本 运行模板: PNG.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
n: 69 56 42 4F 52 41 41 41 41 4E iVBORw0KGgoAAAAN
n: 53 55 68 45 55 41 41 41 47 51 SUHEUgAAApMAAAGQ
n: 43 41 59 41 41 53 41 41 41 41 CAYAAAB/P+mSAAAA
n: 41 58 4E 53 52 63 36 51 41 41 AXNSR0IARs4c6QAA
n: 41 41 52 6E 51 78 6A 77 76 38 AArnQU1BAACxjwv8
n: 59 51 55 41 41 5A 63 77 41 41 YQUAAAAJcEhZcwAA
n: 45 6E 51 41 41 6D 48 33 67 41 EnQAABJ0Ad5mH3gA
n: 41 50 2B 6C 53 65 37 4A 30 4A AP+LSURBVHhe7J0J
n: 76 48 62 56 2B 67 69 52 53 52 vHbV+P43qWggiRSR
n: 42 73 31 46 6B 4B 56 47 68 36 Bs1FkqJREw3KVGH6
n: 47 35 55 78 6F 6B 4B 44 52 71 G5UxoTL3N4VkkDRq
n: 4A 6C 4F 6D 55 52 63 32 6E 51 J1OmUEIjkb1Rc2nQ
n: 54 46 45 30 79 75 34 33 37 76 TFE0yO/vu3qu437v
n: 64 2B 31 6E 4F 76 4F 64 66 33 d+1n0Oc55z3vOdf3
n: 38 31 6C 6E 54 66 63 61 39 6E 37 32 33 75 73 36 81lnTfca9n723us6
n: 61 2B 33 68 4D 56 64 65 64 66 58 2F 4E 57 61 4F a+3hMVdedfX/NWaO
n: 35 7A 47 50 65 55 7A 7A 66 2F 2F 33 66 79 4F 2B 5zGPeUzzf//3fyO+
n: 30 73 7A 6B 52 37 2B 58 69 4C 39 62 7A 6A 50 47 0szkR7+XiL9bzjPG
n: 47 47 4D 6D 47 34 78 61 48 71 30 6D 4F 52 49 58 GGMmG4xaHq0MORIX
n: 33 59 54 46 32 39 37 32 74 75 61 6D 6D 32 35 71 3YTF2972tuamm25q
n: 6E 76 33 73 5A 7A 63 33 33 48 42 44 38 39 33 76 nv3sZzc33HBD893v
n: 66 72 66 5A 64 39 39 39 6D 36 32 32 32 71 72 35 frfZd999m6222qr5
n: 78 7A 2F 2B 30 54 7A 32 73 59 39 74 46 6C 68 67 xz/+0Tz2sY9tFlhg
CSDN@塞纳河畔的春水

```



得到一张图片，fuzz一下宽高不对劲，尝试爆破。

```
import struct
import binascii
from Crypto.Util.number import bytes_to_long

png_input = open('1.png', 'rb').read()

for i in range(0xFFFF):
    stream = png_input[12:20] + struct.pack('>i', i) + png_input[24:29]
    crc = binascii.crc32(stream)
    if crc == bytes_to_long(png_input[29:33]):
        print(i)
        # 800
```

修改png图片高度为800，出现一串编码，形似镜像后的base64。



解码拿到字符串 Finding...

The screenshot shows a web-based Base64 decoding tool. On the left, the 'Recipe' panel is set to 'From Base64' with the alphabet 'A-Za-z0-9+/' and the option 'Remove non-alphabet chars' checked. The 'Input' field on the right contains the Base64 string 'RmluZGluZy4uIg=='. Below the input, the 'Output' section displays 'Finding...'. The interface also features icons for saving, folder management, and deletion.

CSDN @塞纳河畔的春水

用密码去解压flag.rar，拿到最终flag。

```
DASCTF{flag_1s_close_at_your_hand}
```

qrimg

下载附件，是个gif文件，将其一帧一帧分离，共计312帧，尝试将其中的图片拖进StegSolve工具查看，发现在Blue plane0时会出现二维码，如下图所示：



312张图片每一张的Blue plane0通道都有二维码，写脚本提取：

```
from PIL import Image
import pyzbar.pyzbar as pyzbar
import os, base64

def qrcode_parse_content(img):
    barcodes = pyzbar.decode(img)

    result = []
    for barcode in barcodes:
        barcode_content = barcode.data.decode('utf-8')
        result.append(barcode_content)

    return result

def getB0(filepath):
    list1 = []
    im1 = Image.open(filepath)
    for h in range(height):
        for w in range(width):
            b0 = bin(im1.getpixel((w, h))[2])[-1] # 取b通道的低0位
            list1.append(b0)
    return list1
```

```

def genpic(list1):
    im1 = Image.new("RGB", (width, height), 'white')
    i = 0
    for y in range(height):
        for x in range(width):
            if list1[i] == '0':
                im1.putpixel([x, y], (0, 0, 0))
            else:
                im1.putpixel([x, y], (255, 255, 255))
            i = i + 1
    return im1

if __name__ == "__main__":
    tmppath = './tmp/'
    if not os.path.exists(tmppath):
        os.mkdir(tmppath)
    print(tmppath)

    tmp = []
    for i in range(312):
        filepath = './qrimg gif/' + 'IMG' + str(i).zfill(5) + '.bmp'
        width, height = Image.open(filepath).size # 得到宽高
        img = genpic(getB0(filepath))
        img.save(tmppath + str(i).zfill(5) + '.png')
        print(qrcode_parse_content(img))
        tmp += qrcode_parse_content(img)
    str1 = ''.join(tmp)
    print(str1)
    while True:
        try:
            str1 = base64.b64decode(str1).decode("utf-8")
            print(150 * '*')
            print(str1)
        except:
            print(str1)
            break

```

最后连成字符串为base64编码，多次解码之后拿到flag

```
flag{32bb3b8cec39e43a06038a9f96921906}
```