BUUCTF-刷题记录-6

原创

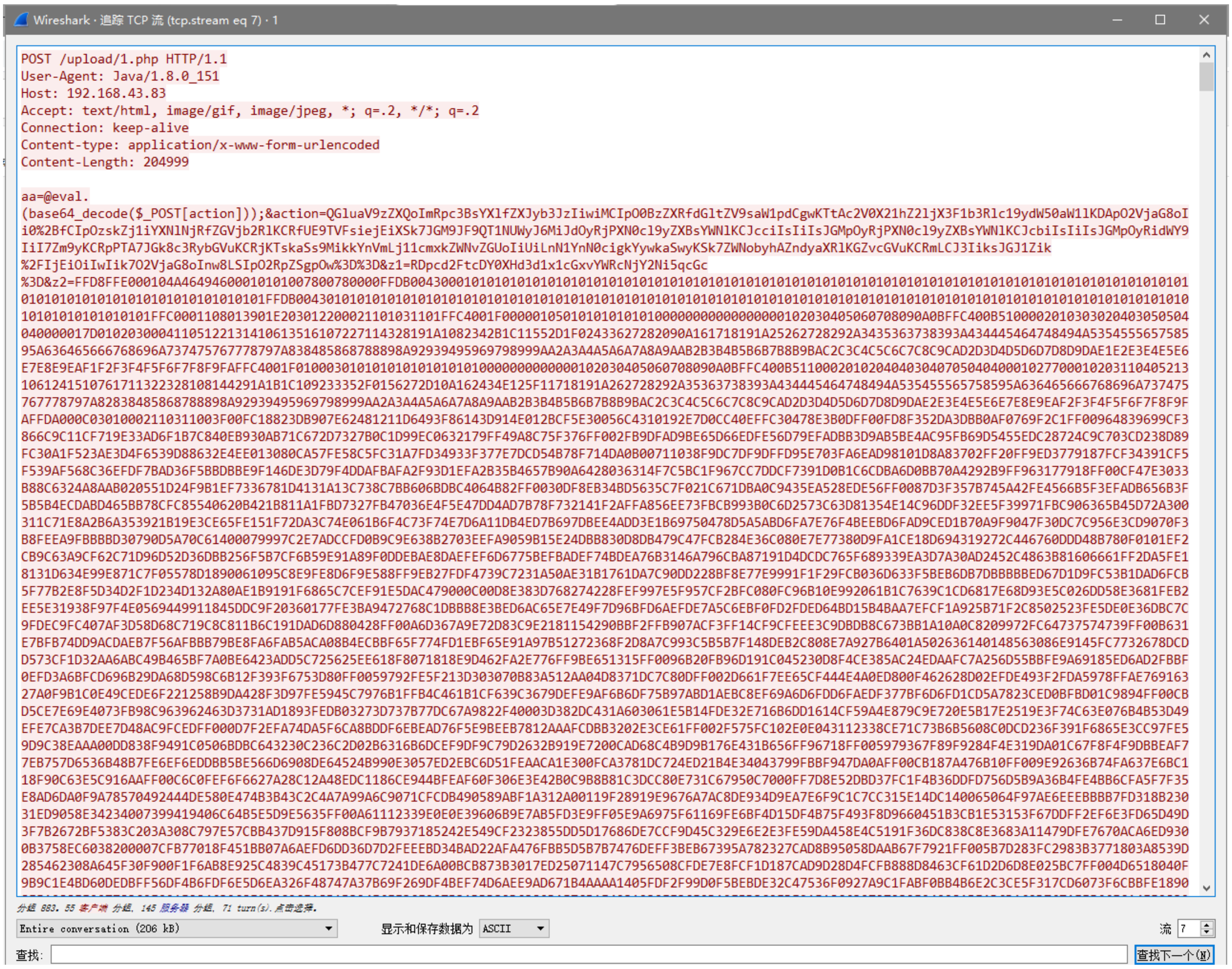
秋风瑟瑟... 于 2020-10-03 22:11:03 发布 311 收藏
分类专栏： BUUCTF刷题记录

BUUCTF刷题记录 专栏收录该内容
10 篇文章 0 订阅
订阅专栏

MISC

菜刀666

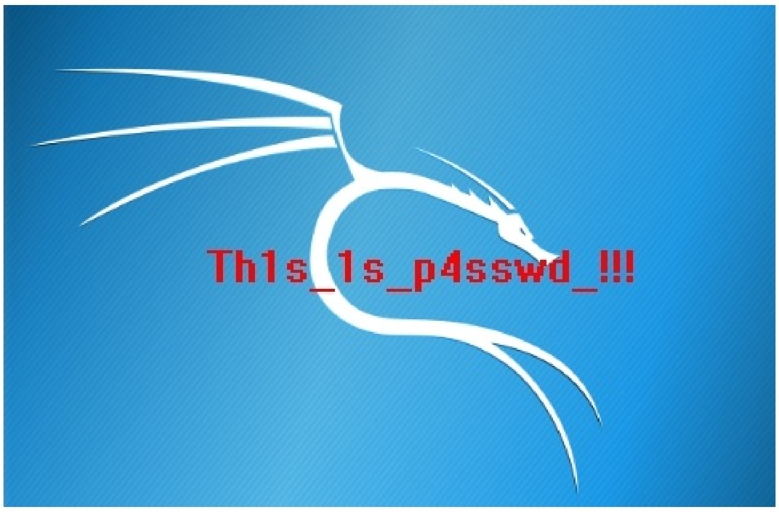
在tcp的第7个流里面发现传输了一张图片



```
POST /upload/1.php HTTP/1.1
User-Agent: Java/1.8.0_151
Host: 192.168.43.83
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 204999

aa=@eval.
(base64_decode($_POST[action]));&action=QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwiMCIpO0BzZXRfdGltZV9saW1pdCgwKTtAc2V0X21hZ2ljX3F1b3Rlc19ydW50aW1lKDApO2VjaG8oI
i0%2BfCIpOzskZj1iYXN1NjRfZGVjb2RlKCRfUE9TVFsiejEiXSk7JGM9JF9QT1NUWyJ6MiJdOyRjPXN0cl9yZXBsYWNlKCJcciIsIiIsJGMpOyRjPXN0cl9yZXBsYWNlKCJcbiIsIiIsJGMpOyRidWY9
IiI7Zm9yKCRpPTA7JGk8c3RybGVuKCRjKTskaSs9MikkYnVmLj11cmxkZWNvZGUoIiUiLnN1YnN0cigkYywkaSwyKSk7ZWNobyhAZndyaXRlKGZvcGVuKCRmLCJ3IiksJGJ1Zik
%2FIjEiOiIwIik7O2VjaG8oInw8LSIpO2RpZSgpOw%3D%3D&z1=RDpcd2FtcDY0XHd3d1x1cGxvYWRcNjY2Ni5qcGc
%3D&z2=FFD8FFE000104A46494600010101007800780000FFDB0043000101010101010101010101010101010101010101010101010101010101010101010101010101010101010101
0101010101010101010101010101FFDB00430101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010
10101010101010101FFC0001108013901E203012200021101031101FFC4001F0000010501010101010100000000000000000102030405060708090A0BFFC400B5100002010303020403050504
040000017D01020300041105122131410613516107227114328191A1082342B1C11552D1F02433627282090A161718191A25262728292A3435363738393A434445464748494A5354555657585
95A636465666768696A737475767778797A838485868788898A92939495969798999AA2A3A4A5A6A7A8A9AAB2B3B4B5B6B7B8B9BAC2C3C4C5C6C7C8C9CAD2D3D4D5D6D7D8D9DAE1E2E3E4E5E6
E7E8E9EAF1F2F3F4F5F6F7F8F9FAFFC4001F0100030101010101010101010000000000000102030405060708090A0BFFC400B51100020102040403040705040400010277000102031104052131
0612415107617113223281081442914A1B1C109233352F0156272D10A162434E125F11718191A262728292A35363738393A434445464748494A535455565758595A636465666768696A737475
767778797A82838485868788898A92939495969798999AA2A3A4A5A6A7A8A9AAB2B3B4B5B6B7B8B9BAC2C3C4C5C6C7C8C9CAD2D3D4D5D6D7D8D9DAE2E3E4E5E6E7E8E9EAF2F3F4F5F6F7F8F9F
AFFDA000C03010002110311003F00FC18823DB907E62481211D6493F86143D914E012BCF5E30056C4310192E7D0CC40EFFC30478E3B0DFF00FD8F352DA3DBB0AF0769F2C1FF00964839699CF3
866C9C11CF719E33AD6F1B7C840EB930AB71C672D7327B0C1D99EC0632179FF49A8C75F376FF002FB9DFAD9BE65D66EDFE56D79EFADBB3D9AB5BE4AC95FB69D5455EDC28724C9C703CD238D89
FC30A1F523AE3D4F6539D88632E4EE013080CA57FE58C5FC31A7FD34933F377E7DCD54B78F714DA0B00711038F9DC7DF9DFFD95E703FA6EAD98101D8A83702FF20FF9ED3779187FCF34391CF5
F539AF568C36EFDF7BAD36F5BBDBBE9F146DE3D79F4DDAFBAFA2F93D1EFA2B35B4657B90A6428036314F7C5BC1F967CC7DDCF7391D0B1C6CDBA6D0BB70A4292B9FF963177918FF00CF47E3033
B88C6324A8AAB020551D24F9B1EF7336781D4131A13C738C7BB606BDBC4064B82FF0030DF8EB34BD5635C7F021C671DBA0C9435EA528EDE56FF0087D3F357B745A42FE4566B5F3EFADB656B3F
5B5B4ECDABD465BB78CFC85540620B421B811A1FBD7327FB47036E4F5E47DD4AD7B78F732141F2AFFA856EE73FBCB993B0C6D2573C63D81354E14C96DDF32EE5F39971FBC906365B45D72A300
311C71E8A2B6A353921B19E3CE65FE151F72DA3C74E061B6F4C73F74E7D6A11DB4ED7B697DBEE4ADD3E1B69750478D5A5ABD6FA7E76F4BEEBD6FAD9CED1B70A9F9047F30DC7C956E3CD9070F3
B8FEEA9FBBBBD30790D5A70C61400079997C2E7ADCCFD0B9C9E638B2703EEFA9059B15E24DBB830D8DB479C47FCB284E36C080E7E77380D9FA1CE18D694319272C446760DDD48B780F0101EF2
CB9C63A9CF62C71D96D52D36DBB256F5B7CF6B59E91A89F0DDEBAE8DAEFEF6D6775BEFBADEF74BDEA76B3146A796CBA87191D4DCDC765F689339EA3D7A30AD2452C4863B81606661FF2DA5FE1
8131D634E99E871C7F05578D1890061095C8E9FE8D6F9E588FF9EB27FDF4739C7231A50AE31B1761DA7C90DD228BF8E77E9991F1F29FCB036D633F5BEB6DB7DBBBBBED67D1D9FC53B1DAD6FCB
5F77B2E8F5D34D2F1D234D132A80AE1B9191F6865C7CEF91E5DAC479000C00D8E383D768274228FEF997E5F957CF2BFC080FC96B10E992061B1C7639C1CD6817E68D93E5C026DD58E3681FEB2
EE5E31938F97F4E0569449911845DDC9F20360177FE3BA9472768C1DBBB8E3BED6AC65E7E49F7D96BFD6AEFDE7A5C6EBF0FD2FDED64BD15B4BAA7EFCF1A925B71F2C8502523FE5DE0E36DBC7C
9FDEC9FC407AF3D58D68C719C8C811B6C191DAD6D880428FF00A6D367A9E72D83C9E2181154290BBF2FFB907ACF3FF14CF9CFEEE3C9DBDB8C673BB1A10A0C8209972FC64737574739FF00B631
E7BFB74DD9ACDAEB7F56AFBBB79BE8FA6FAB5ACA08B4ECBBF65F774FD1EBF65E91A97B51272368F2D8A7C993C5B5B7F148DEB2C808E7A927B6401A502636140148563086E9145FC7732678DCD
D573CF1D32AA6ABC49B465BF7A0BE6423ADD5C725625EE618F8071818E9D462FA2E776FF9BE651315FF0096B20FB96D191C045230D8F4CE385AC24EDAAFC7A256D55BBFE9A69185ED6AD2FBBF
0EFD3A6BFCD696B29DA68D598C6B12F393F6753D80FF0059792FE5F213D303070B83A512AA04D8371DC7C80DFF002D661F7EE65CF444E4A0ED800F462628D02EFDE493F2FDA5978FFAE769163
27A0F9B1C0E49CEDE6F221258B9DA428F3D97FE5945C7976B1FFB4C461B1CF639C3679DEFE9AF6B6DF75B97ABD1AEBC8EF69A6D6FDD6FAEDF377BF6D6FD1CD5A7823CED0BFBD01C9894FF00CB
D5CE7E69E4073FB98C963962463D3731AD1893FEDB03273D737B77DC67A9822F40003D382DC431A603061E5B14FDE32E716B6DD1614CF59A4E879C9E720E5B17E2519E3F74C63E076B4B53D49
EFE7CA3B7DEE7D48AC9FCEDFF000D7F2EFA74DA5F6CA8BDDF6EBEAD76F5E9BEEB7812AAAFCDBB3202E3CE61FF002F575FC102E0E043112338CE71C73B6B5608C0DCD236F391F6865E3CC97FE5
9D9C38EAAA00DD838F9491C0506BDBC643230C236C2D02B6316B6DCEF9DF9C79D2632B919E7200CAD68C4B9D9B176E431B656FF96718FF005979367F89F9284F4E319DA01C67F8F4F9DBBEAF7
7EB757D6536B48B7FE6EF6EDDBB5BE566D6908DE64524B990E3057ED2EBC6D51FEAACA1E300FCA3781DC724ED21B4E34043799FBBF947DA0AFF00CB187A476B10FF009E92636B74FA637E6BC1
18F90C63E5C916AAFF00C6C0FEF6F6627A28C12A48EDC1186CE944BFEAF60F306E3E42B0C9B8B81C3DCC80E731C67950C7000FF7D8E52DBD37FC1F4B36DDFD756D5B9A36B4FE4BB6CFA5F7F35
E8AD6DA0F9A78570492444DE580E474B3B43C2C4A7A99A6C9071CFCDB490589ABF1A312A00119F28919E9676A7AC8DE934D9EA7E6F9C1C7CC315E14DC140065064F97AE6EEEBBBB7FD318B230
31ED9058E342340073999419406C64B5E5D9E5635FF00A61112339E0E0E39606B9E7AB5FD3E9FF05E9A6975F61169FE6BF4D15DF4B75F493F8D9660451B3CB1E53153F67DDFF2EF6E3FD65D49D
3F7B2672BF5383C203A308C797E57CBB437D915F808BCF9B7937185242E549CF2323855DD5D17686DE7CCF9D45C329E6E2E3FE59DA458E4C5191F36DC838C8E3683A11479DFE7670ACA6ED930
0B3758EC6038200007CFB77018F451BB07A6AEFD6DD36D7D2FEEEBD34BAD22AFA476FBB5D5B7B7476DEFF3BEB67395A782327CAD8B95058DAAB67F7921FF005B7D283FC2983B3771803A8539D
285462308A645F30F900F1F6AB8E925C4839C45173B477C7241DE6A00BCB873B3017ED25071147C7956508CFDE7E8FCF1D187CAD9D28D4FCFB888D8463CF61D2D6D8E025BC7FF004D6518040F
9B9C1E4BD60DEDBFF56DF4B6FDF6E5D6EA326F48747A37B69F269DF4BEF74D6AEE9AD671B4AAAA1405FDF2F99D0F5BEBDE32C47536F0927A9C1FABF0BB4B6E2C3CE5F317CD6073F6CBBFE1890
```

分组 883. 55 客户端 分组, 145 服务器 分组, 71 turn(s). 点击选择.
Entire conversation (206 kB)
显示和保存数据为
ASCII
流
7
查找:
查找下一个(N)

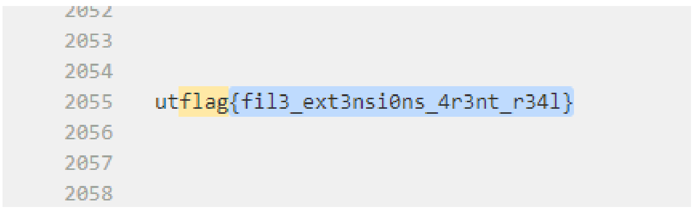
foremost分离出来一个带有密码的压缩包，把图片给提取出来，得到密码



解压得到flag

[UTCTF2020]basic-forensics

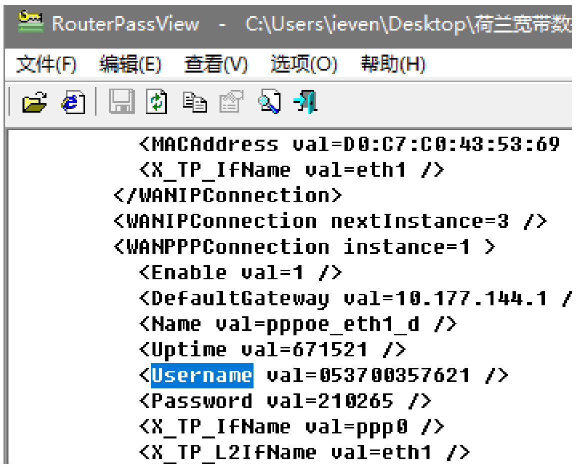
给了一个jpeg，但其实不是，修改为txt，直接全局搜索`flag`就看到了flag



```
utflag{fil3_ext3nsi0ns_4r3nt_r34l}
```

荷兰宽带数据泄露

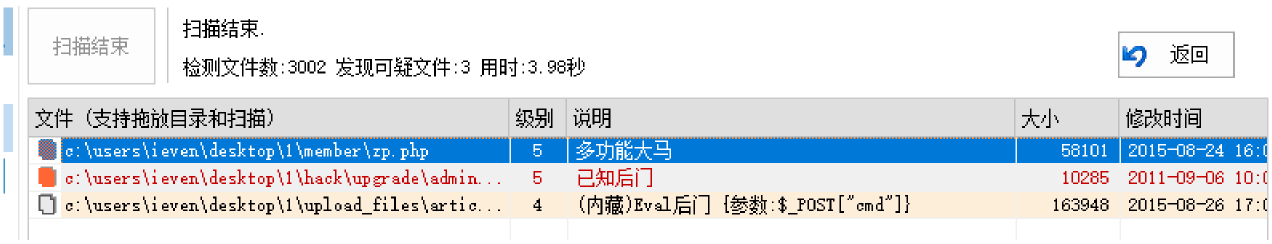
使用`routerpassview`工具查看路由器配置文件里面存的的用户名即可，flag即为用户名


RouterPassView - C:\Users\ieven\Desktop\荷兰宽带数
文件(F) 编辑(E) 查看(V) 选项(O) 帮助(H)

```
        <MACAddress val=D0:C7:C0:43:53:69
        <X_TP_IfName val=eth1 />
      </WANIPConnection>
      <WANIPConnection nextInstance=3 />
      <WANPPPConnection instance=1 >
        <Enable val=1 />
        <DefaultGateway val=10.177.144.1 /
        <Name val=pppoe_eth1_d />
        <Uptime val=671521 />
        <Username val=053700357621 />
        <Password val=210265 />
        <X_TP_IfName val=ppp0 />
        <X_TP_L2IfName val=eth1 />
```

webshell后门

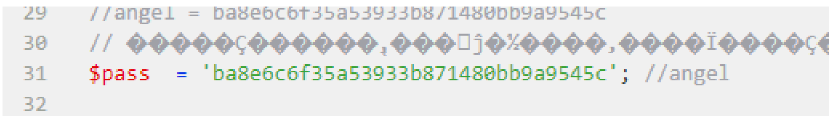
D盾扫一下，发现后门文件



扫描结束.
检测文件数:3002 发现可疑文件:3 用时:3.98秒

文件（支持拖放目录和扫描）	级别	说明	大小	修改时间
c:\users\ieven\desktop\1\member\zp.php	5	多功能大马	58101	2015-08-24 16:(
c:\users\ieven\desktop\1\hack\upgrade\admin...	5	已知后门	10285	2011-09-06 10:(
c:\users\ieven\desktop\1\upload_files\artic...	4	(内藏)Eval后门 {参数:$_POST["cmd"]}	163948	2015-08-26 17:(

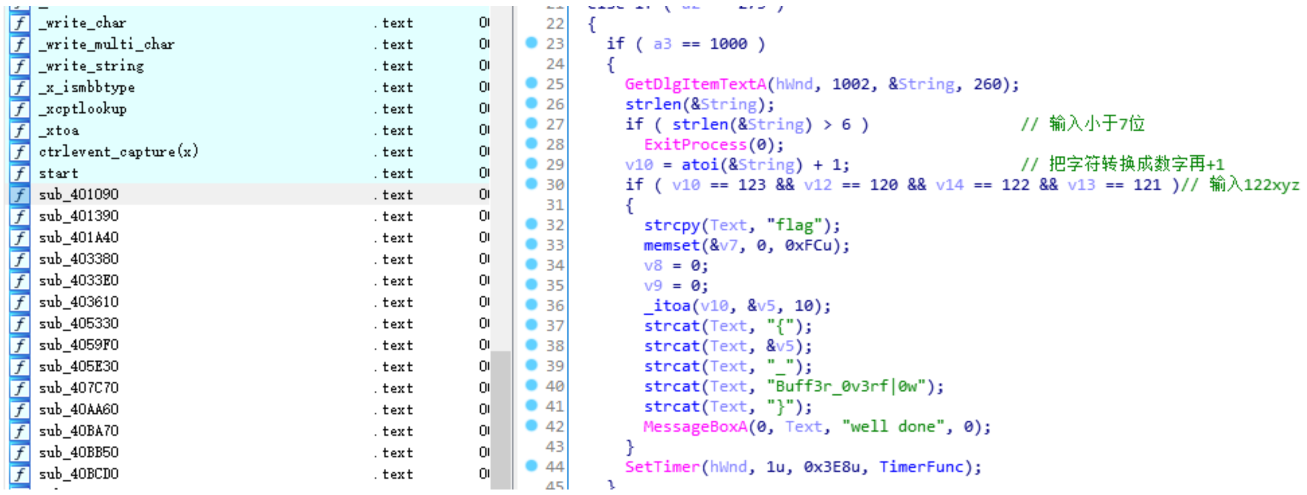
然后搜索pass就可以看得到密码，也就是flag了



```
//angel = ba8e6c6f35a53933b871480bb9a9545c
// ������ç�������,����□j�½�����,�����Ï�����ç�
$pass  = 'ba8e6c6f35a53933b871480bb9a9545c'; //angel
```

Mysterious

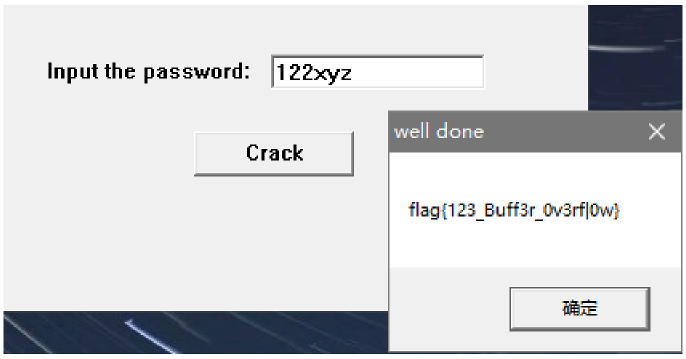
一个简单的逆向题，先定位到flag的地方，进行一个分析



```
_write_char             .text
_write_multi_char       .text
_write_string           .text
_x_ismbbtype            .text
_xcptlookup             .text
_xtoa                   .text
ctrlevent_capture(x)    .text
start                   .text
sub_401090              .text
sub_401390              .text
sub_401A40              .text
sub_403380              .text
sub_4033E0              .text
sub_403610              .text
sub_405330              .text
sub_4059F0              .text
sub_405E30              .text
sub_407C70              .text
sub_40AA60              .text
sub_40BA70              .text
sub_40BB50              .text
sub_40BCD0              .text
```

```c
{
  if ( a3 == 1000 )
  {
    GetDlgItemTextA(hWnd, 1002, &String, 260);
    strlen(&String);
    if ( strlen(&String) > 6 )                  // 输入小于7位
      ExitProcess(0);
    v10 = atoi(&String) + 1;                    // 把字符转换成数字再+1
    if ( v10 == 123 && v12 == 120 && v14 == 122 && v13 == 121 )// 输入122xyz
    {
      strcpy(Text, "flag");
      memset(&v7, 0, 0xFCu);
      v8 = 0;
      v9 = 0;
      _itoa(v10, &v5, 10);
      strcat(Text, "{");
      strcat(Text, &v5);
      strcat(Text, "_");
      strcat(Text, "Buff3r_0v3rf|0w");
      strcat(Text, "}");
      MessageBoxA(0, Text, "well done", 0);
    }
    SetTimer(hWnd, 1u, 0x3E8u, TimerFunc);
```

然后咱们输入`122xyz`就可以得到flag了



flag{123_Buff3r_0v3rf|0w}

蜘蛛侠呀

一顿分析无果，先列出流量包的隐藏文件

```
tshark -r out.pcap -T fields -e data > 1.txt
```

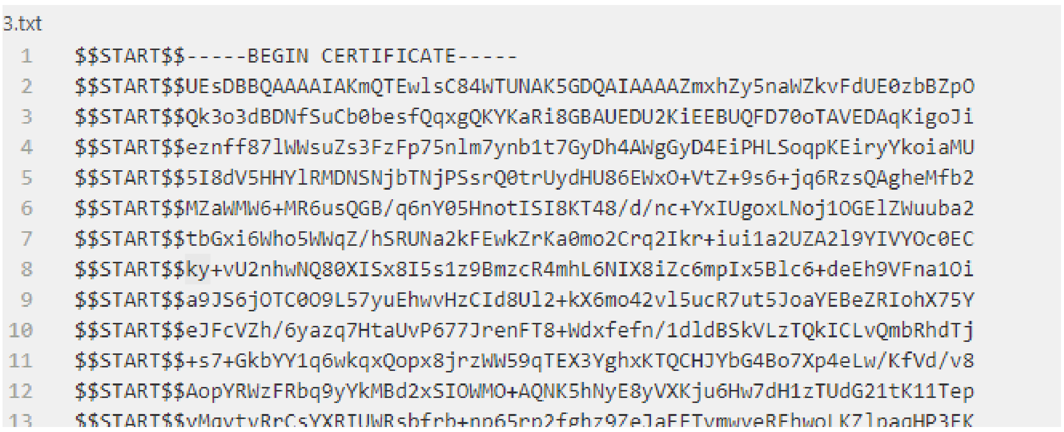
发现导出的文件里面，每四行都是重复的，于是去掉重复行，脚本如下

```python
f1 = open('1.txt','r').readlines()
f2 = open('2.txt','a')

for i in range(len(f1)):
 if f1[i].strip() == f1[i-1].strip():
  continue
 f2.write(f1[i].strip() + '\n')
```

然后再把每行的16进制转字符

```python
f1 = open("2.txt",'r').readlines()
f2 = open("3.txt","w")

for line in f1:
    newline = ''
    for i in range(len(line)//2):
        byte = line[i*2:i*2+2]
        num = int(byte,16)
        character = chr(num)
        newline += character
    f2.writelines(newline)
f2.close()
```

但是前面出现了一堆奇怪的字符



```
3.txt
$$START$$-----BEGIN CERTIFICATE-----
$$START$$UEsDBBQAAAAIAKmQTEwlsC84WTUNAK5GDQAIAAAAZmxhZy5naWZkvFdUE0zbBZpO
$$START$$Qk3o3dBDNfSuCb0besfQqxgQKYKaRi8GBAUEDU2KiEEBUQFD70oTAVEDAqKigoJi
$$START$$eznff87lWWsuZs3FzFp75nlm7ynb1t7GyDh4AWgGyD4EiPHLSoqpKEiryYkoiaMU
$$START$$5I8dV5HHYlRMDNSNjbTNjPSsrQ0trUydHU86EWxO+VtZ+9s6+jq6RzsQAgheMfb2
$$START$$MZaWMW6+MR6usQGB/q6nY05HnotISI8KT48/d/nc+YxIUgoxLNoj1OGElZWuuba2
$$START$$tbGxi6Who5WWqZ/hSRUNa2kFEwkZrKa0mo2Crq2Ikr+iui1a2UZA2l9YIVYOc0EC
$$START$$ky+vU2nhwNQ80XISx8I5s1z9BmzcR4mhL6NIX8iZc6mpIx5Blc6+deEh9VFna1Oi
$$START$$a9JS6jOTC0O9L57yuEhwvHzCId8Ul2+kX6mo42vl5ucR7ut5JoaYEBeZRIohX75Y
$$START$$eJFcVZh/6yazq7HtaUvP677JrenFT8+Wdxfefn/1dldBSkVLzTQkICLvQmbRhdTj
$$START$$+s7+GkbYY1q6wkqxQopx8jrzWW59qTEX3YghxKTQCHJYbG4Bo7Xp4eLw/KfVd/v8
$$START$$AopYRWzFRbq9yYkMBd2xSIOWMO+AQNK5hNyE8yVXKju6Hw7dH1zTUdG21tK11Tep
$$START$$yMgvtyRrCsYXRTUWRsbfrb+pp65rp2fghz97eJaFETvmwyeRFhwoLKZ1pagHP3FK
```

直接给他全部替换成空，然后使用notepad++插件转进行base64解码，保存为zip，存在一个flag.gif，然后就是时间隐写了，查看gif每一帧停留的秒数，分别替换成01，然后在8个一组转字符

```
identify -format "%T" flag.gif
```

得到

```
2050502050502050205020202050202020205050205020502050205050505050202050502020202050205050502050202066666
```

最后面的四个6不用管

```
01101101 01000100 00110101 01011111 00110001 01110100
```

最后转换成字符，得到

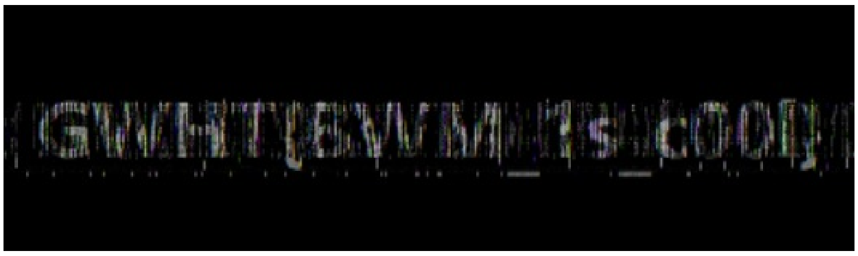
```
mD5_1t
```

把其进行md5加密得到的就是flag了

[GWCTF2019]huyao

一看就知道是盲水印了，不过这个用的是老版本的盲水印，`flag{BWM_1s_c00l}`

```
python2 decode.py --original 2.png --image 1.png --result res1.png
```



[湖南省赛2019]Findme

一共给了5张图片，第一张的宽高有问题，用脚本修复一下

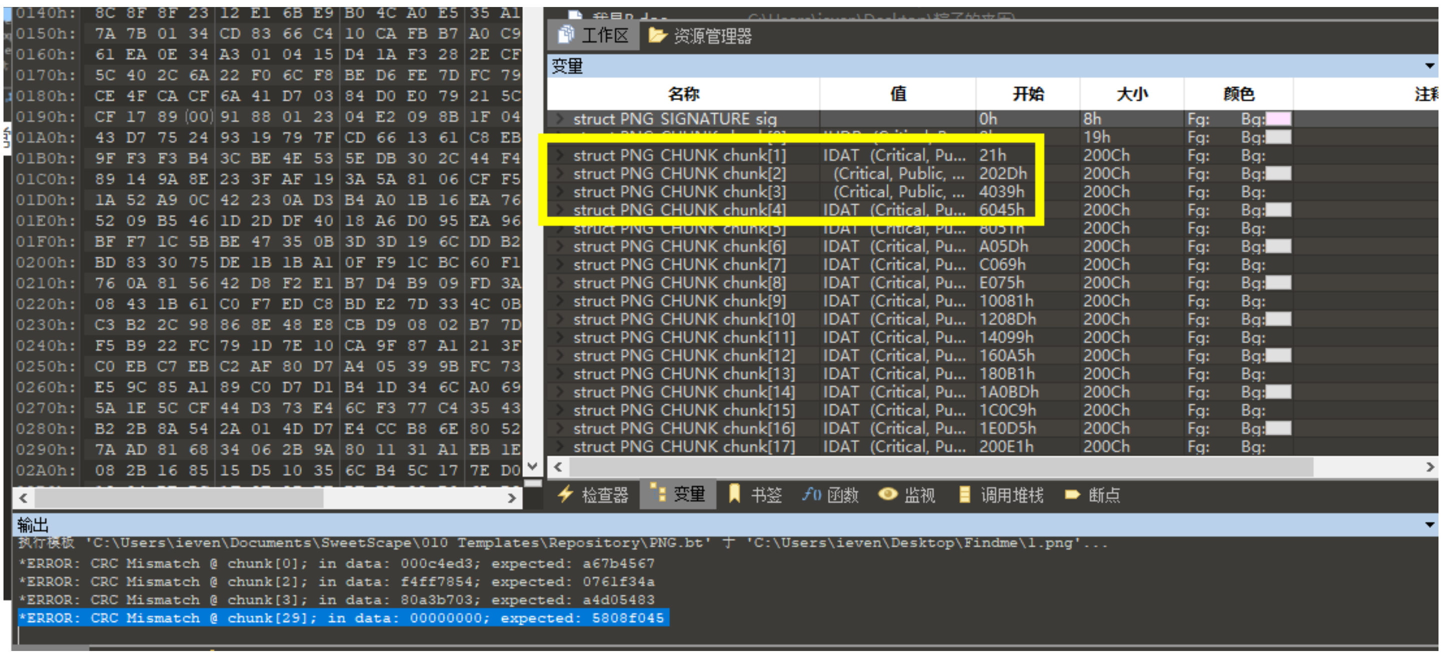
```python
import zlib
import struct

filename = '1.png'
with open(filename, 'rb') as f:
    all_b = f.read()
    crc32key = int(all_b[29:33].hex(),16)
    data = bytearray(all_b[12:29])
    n = 4095
    for w in range(n):
        width = bytearray(struct.pack('>i', w))
        for h in range(n):
            height = bytearray(struct.pack('>i', h))
            for x in range(4):
                data[x+4] = width[x]
                data[x+8] = height[x]
            crc32result = zlib.crc32(data)
            if crc32result == crc32key:
                print("width: ",end="")
                print(width)
                print("height: ",end="")
                print(height)
```

得到结果

```
width: bytearray(b'\x00\x00\x00\xe3')
height: bytearray(b'\x00\x00\x01\xc5')
```

修改打开之后，是这样的

丢进010分析，发现图片缺少chunk[2]、chunk[3]的`IDAT`标识：`49 44 41 54`

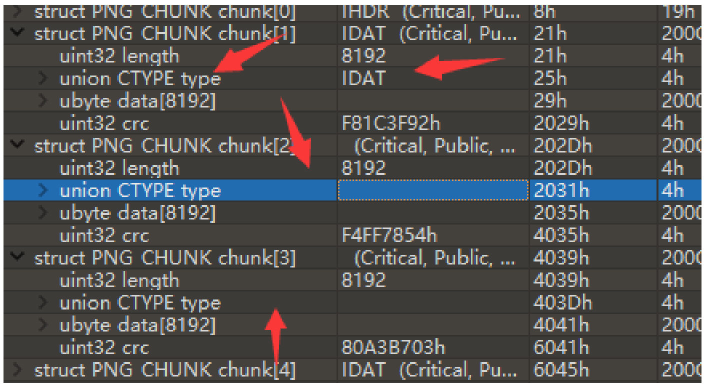


```
0140h: 8C 8F 8F 23 12 E1 6B E9 B0 4C A0 E5 35 A1
0150h: 7A 7B 01 34 CD 83 66 C4 10 CA FB B7 A0 C9
0160h: 61 EA 0E 34 A3 01 04 15 D4 1A F3 28 2E CF
0170h: 5C 40 2C 6A 22 F0 6C F8 BE D6 FE 7D FC 79
0180h: CE 4F CA CF 6A 41 D7 03 84 D0 E0 79 21 5C
0190h: CF 17 89 00 91 88 01 23 04 E2 09 8B 1F 04
01A0h: 43 D7 75 24 93 19 79 7F CD 66 13 61 C8 EB
01B0h: 9F F3 F3 B4 3C BE 4E 53 5E DB 30 2C 44 F4
01C0h: 89 14 9A 8E 23 3F AF 19 3A 5A 81 06 CF F5
01D0h: 1A 52 A9 0C 42 23 0A D3 B4 A0 1B 16 EA 76
01E0h: 52 09 B5 46 1D 2D DF 40 18 A6 D0 95 EA 96
01F0h: BF F7 1C 5B BE 47 35 0B 3D 3D 19 6C DD B2
0200h: BD 83 30 75 DE 1B 1B A1 0F F9 1C BC 60 F1
0210h: 76 0A 81 56 42 D8 F2 E1 B7 D4 B9 09 FD 3A
0220h: 08 43 1B 61 C0 F7 ED C8 BD E2 7D 33 4C 0B
0230h: C3 B2 2C 98 86 8E 48 E8 CB D9 08 02 B7 7D
0240h: F5 B9 22 FC 79 1D 7E 10 CA 9F 87 A1 21 3F
0250h: C0 EB C7 EB C2 AF 80 D7 A4 05 39 9B FC 73
0260h: E5 9C 85 A1 89 C0 D7 D1 B4 1D 34 6C A0 69
0270h: 5A 1E 5C CF 44 D3 73 E4 6C F3 77 C4 35 43
0280h: B2 2B 8A 54 2A 01 4D D7 E4 CC B8 6E 80 52
0290h: 7A AD 81 68 34 06 2B 9A 80 11 31 A1 EB 1E
02A0h: 08 2B 16 85 15 D5 10 35 6C B4 5C 17 7E D0
```

名称	值	开始	大小	颜色	
struct PNG SIGNATURE sig		0h	8h	Fg:	Bg:
			19h	Fg:	Bg:
struct PNG CHUNK chunk[1]	IDAT (Critical, Pu...	21h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[2]	(Critical, Public, ...	202Dh	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[3]	(Critical, Public, ...	4039h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[4]	IDAT (Critical, Pu...	6045h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[5]	IDAT (Critical, Pu...	8051h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[6]	IDAT (Critical, Pu...	A05Dh	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[7]	IDAT (Critical, Pu...	C069h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[8]	IDAT (Critical, Pu...	E075h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[9]	IDAT (Critical, Pu...	10081h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[10]	IDAT (Critical, Pu...	1208Dh	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[11]	IDAT (Critical, Pu...	14099h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[12]	IDAT (Critical, Pu...	160A5h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[13]	IDAT (Critical, Pu...	180B1h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[14]	IDAT (Critical, Pu...	1A0BDh	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[15]	IDAT (Critical, Pu...	1C0C9h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[16]	IDAT (Critical, Pu...	1E0D5h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[17]	IDAT (Critical, Pu...	200E1h	200Ch	Fg:	Bg:

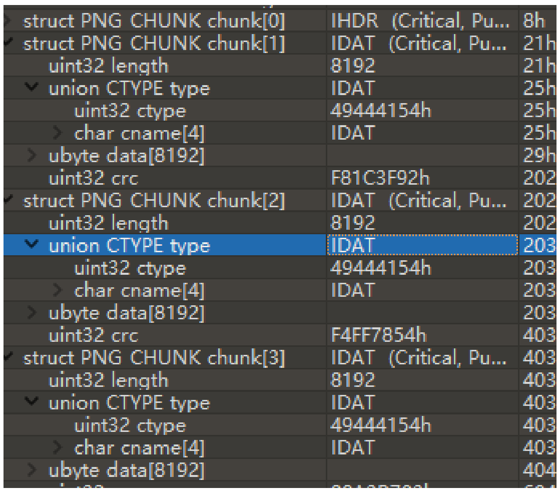
```
执行模板 'C:\Users\ieven\Documents\SweetScape\010 Templates\Repository\PNG.bt' 于 'C:\Users\ieven\Desktop\Findme\1.png'...
*ERROR: CRC Mismatch @ chunk[0]; in data: 000c4ed3; expected: a67b4567
*ERROR: CRC Mismatch @ chunk[2]; in data: f4ff7854; expected: 0761f34a
*ERROR: CRC Mismatch @ chunk[3]; in data: 80a3b703; expected: a4d05483
*ERROR: CRC Mismatch @ chunk[29]; in data: 00000000; expected: 5808f045
```

那我们给他加上去，修复就好了

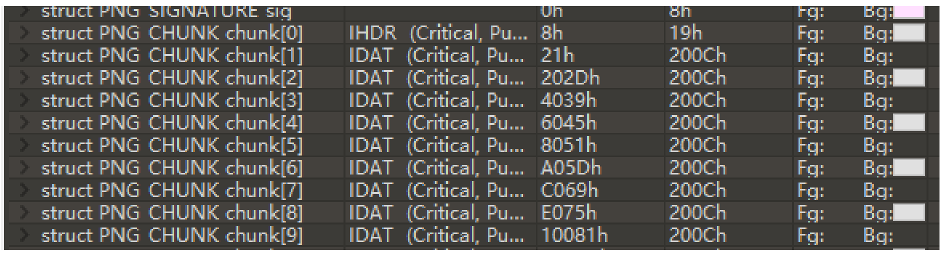


struct PNG CHUNK chunk[0]	IHDR (Critical, Pu...	8h	19h
struct PNG CHUNK chunk[1]	IDAT (Critical, Pu...	21h	200
uint32 length	8192	21h	4h
union CTYPE type	IDAT	25h	4h
ubyte data[8192]		29h	200
uint32 crc	F81C3F92h	2029h	4h
struct PNG CHUNK chunk[2]	(Critical, Public, ...	202Dh	200
uint32 length	8192	202Dh	4h
union CTYPE type		2031h	4h
ubyte data[8192]		2035h	200
uint32 crc	F4FF7854h	4035h	4h
struct PNG CHUNK chunk[3]	(Critical, Public, ...	4039h	200
uint32 length	8192	4039h	4h
union CTYPE type		403Dh	4h
ubyte data[8192]		4041h	200
uint32 crc	80A3B703h	6041h	4h
struct PNG CHUNK chunk[4]	IDAT (Critical, Pu...	6045h	200

修复完毕



struct PNG CHUNK chunk[0]	IHDR (Critical, Pu...	8h
struct PNG CHUNK chunk[1]	IDAT (Critical, Pu...	21h
uint32 length	8192	21h
union CTYPE type	IDAT	25h
uint32 ctype	49444154h	25h
char cname[4]	IDAT	25h
ubyte data[8192]		29h
uint32 crc	F81C3F92h	202
struct PNG CHUNK chunk[2]	IDAT (Critical, Pu...	202
uint32 length	8192	202
union CTYPE type	IDAT	203
uint32 ctype	49444154h	203
char cname[4]	IDAT	203
ubyte data[8192]		203
uint32 crc	F4FF7854h	403
struct PNG CHUNK chunk[3]	IDAT (Critical, Pu...	403
uint32 length	8192	403
union CTYPE type	IDAT	403
uint32 ctype	49444154h	403
char cname[4]	IDAT	403
ubyte data[8192]		404

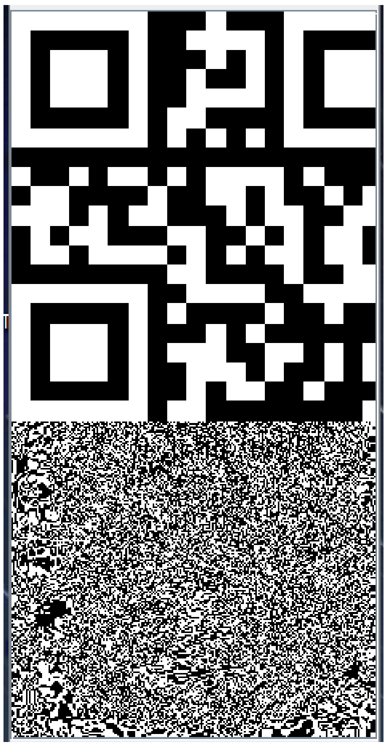


struct PNG SIGNATURE sig		0h	8h	Fg:	Bg:
struct PNG CHUNK chunk[0]	IHDR (Critical, Pu...	8h	19h	Fg:	Bg:
struct PNG CHUNK chunk[1]	IDAT (Critical, Pu...	21h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[2]	IDAT (Critical, Pu...	202Dh	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[3]	IDAT (Critical, Pu...	4039h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[4]	IDAT (Critical, Pu...	6045h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[5]	IDAT (Critical, Pu...	8051h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[6]	IDAT (Critical, Pu...	A05Dh	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[7]	IDAT (Critical, Pu...	C069h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[8]	IDAT (Critical, Pu...	E075h	200Ch	Fg:	Bg:
struct PNG CHUNK chunk[9]	IDAT (Critical, Pu...	10081h	200Ch	Fg:	Bg:

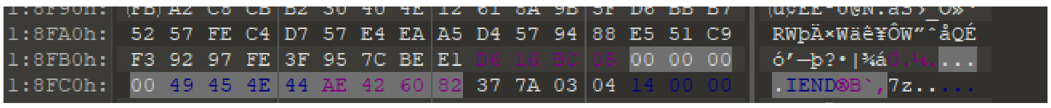
```
struct PNG_CHUNK chunk[10]    IDAT  (Critical, Pu    1208Dh    200Ch    Fg:    Bg:
```

终于是修复好啦~

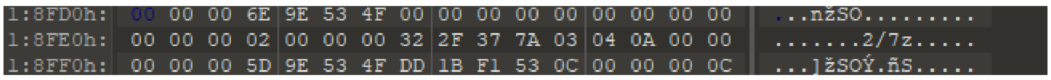
然后下一步就是找信息了，这里一共有5张图片，所以应该是有五段信息或者是什么连在一起的东西，先从第一张看起。
在某个通道里面发现了一张二维码



扫码得到`ZmxhZ3s0X3`，很熟悉的一个flag头，后面就是去找接下来的四段了
第二张图片的尾部发现了7z的压缩包

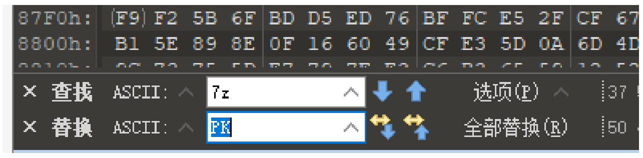


```
1:8FA0h:  52 57 FE C4 D7 57 E4 EA A5 D4 57 94 88 E5 51 C9   RWþÄ×Wäê¥ÔW"ˆåQÉ
1:8FB0h:  F3 92 97 FE 3F 95 7C BE E1 D6 16 BC 05 00 00 00   ó'—þ?•|¾áÖ.¼....
1:8FC0h:  00 49 45 4E 44 AE 42 60 82 37 7A 03 04 14 00 00   .IEND®B`‚7z.....
```



```
1:8FD0h:  00 00 00 6E 9E 53 4F 00 00 00 00 00 00 00 00 00  ...nžSO.........
1:8FE0h:  00 00 00 02 00 00 00 32 2F 37 7A 03 04 0A 00 00  .......2/7z.....
1:8FF0h:  00 00 00 5D 9E 53 4F DD 1B F1 53 0C 00 00 00 0C  ...]žSOÝ.ñS.....
```

将其手动分离出来，但是解压出了问题，再来细看一下，发现原来是PK被换成了7z，那我们再给换回去



```
87F0h:  F9 F2 5B 6F BD D5 ED 76 BF FC E5 2F CF 67
8800h:  B1 5E 89 8E 0F 16 60 49 CF E3 5D 0A 6D 4D
```

× 查找 ASCII: 7z 选项(P)

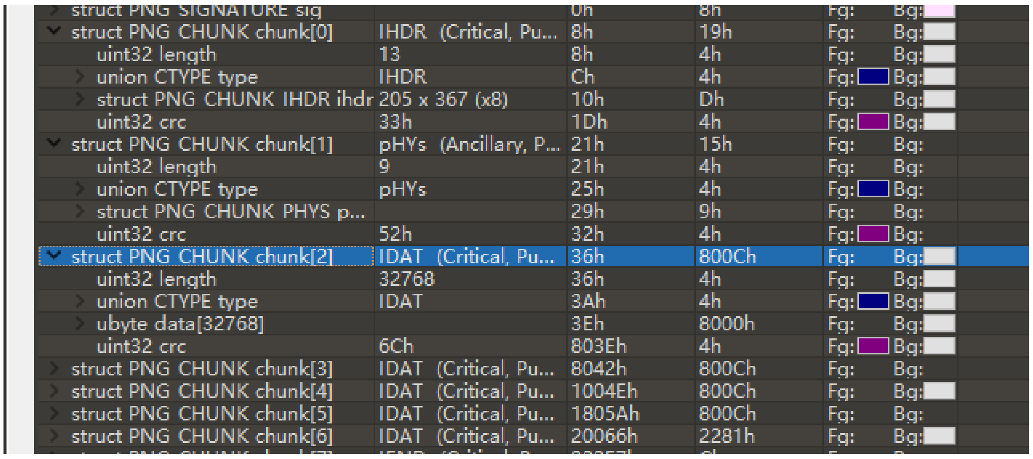
× 替换 ASCII: PK 全部替换(R)

成功解压，有一大堆的文件，我们直接按照修改日期排序，找到了`618.txt`这个藏有内容的文件

`You find it: 1RVcmVfc`

然后来看第三张图片

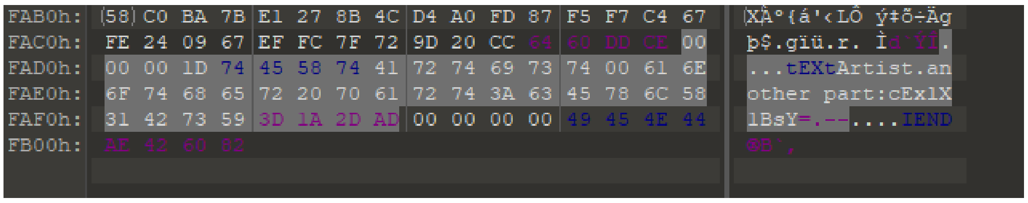
chunk[0]-chunk[6]的每一个数据块的crc值都是可打印的ascii字符



```
  struct PNG SIGNATURE sig                        0h      8h      Fg:  Bg:
v struct PNG CHUNK chunk[0]     IHDR  (Critical, Pu...  8h      19h     Fg:  Bg:
    uint32 length               13                      8h      4h      Fg:  Bg:
  > union CTYPE type            IHDR                    Ch      4h      Fg:  Bg:
  > struct PNG CHUNK IHDR ihdr  205 x 367 (x8)          10h     Dh      Fg:  Bg:
    uint32 crc                  33h                     1Dh     4h      Fg:  Bg:
v struct PNG CHUNK chunk[1]     pHYs  (Ancillary, P...  21h     15h     Fg:  Bg:
    uint32 length               9                       21h     4h      Fg:  Bg:
  > union CTYPE type            pHYs                    25h     4h      Fg:  Bg:
  > struct PNG CHUNK PHYS p...                          29h     9h      Fg:  Bg:
    uint32 crc                  52h                     32h     4h      Fg:  Bg:
v struct PNG CHUNK chunk[2]     IDAT  (Critical, Pu...  36h     800Ch   Fg:  Bg:
    uint32 length               32768                   36h     4h      Fg:  Bg:
  > union CTYPE type            IDAT                    3Ah     4h      Fg:  Bg:
  > ubyte data[32768]                                   3Eh     8000h   Fg:  Bg:
    uint32 crc                  6Ch                     803Eh   4h      Fg:  Bg:
> struct PNG CHUNK chunk[3]     IDAT  (Critical, Pu...  8042h   800Ch   Fg:  Bg:
> struct PNG CHUNK chunk[4]     IDAT  (Critical, Pu...  1004Eh  800Ch   Fg:  Bg:
> struct PNG CHUNK chunk[5]     IDAT  (Critical, Pu...  1805Ah  800Ch   Fg:  Bg:
> struct PNG CHUNK chunk[6]     IDAT  (Critical, Pu...  20066h  2281h   Fg:  Bg:
```

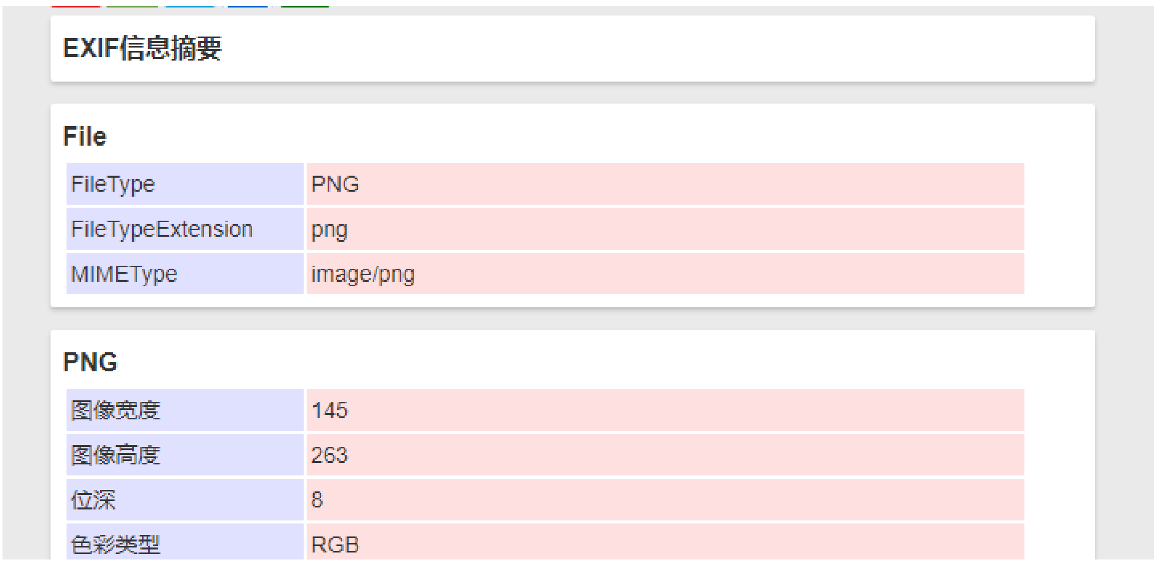
提取出来输出为ascii码也就是`3RlZ30=`

然后来看第四张图片，在文件末尾发现内容`cExlX1BsY`



```
FAB0h:  58 C0 BA 7B E1 27 8B 4C D4 A0 FD 87 F5 F7 C4 67  XÀº{á'‹LÔ ý‡õ÷Äg
FAC0h:  FE 24 09 67 EF FC 7F 72 9D 20 CC 64 60 DD CE 00  þ$.gïü.r. Ìd`ÝÎ.
FAD0h:  00 00 1D 74 45 58 74 41 72 74 69 73 74 00 61 6E  ...tEXtArtist.an
FAE0h:  6F 74 68 65 72 20 70 61 72 74 3A 63 45 78 6C 58  other part:cExlX
FAF0h:  31 42 73 59 3D 1A 2D AD 00 00 00 00 49 45 4E 44  1BsY=.-....IEND
FB00h:  AE 42 60 82                                      ®B`‚
```

其实也就是图片的EXIF信息，可以在这个网站上面看得到



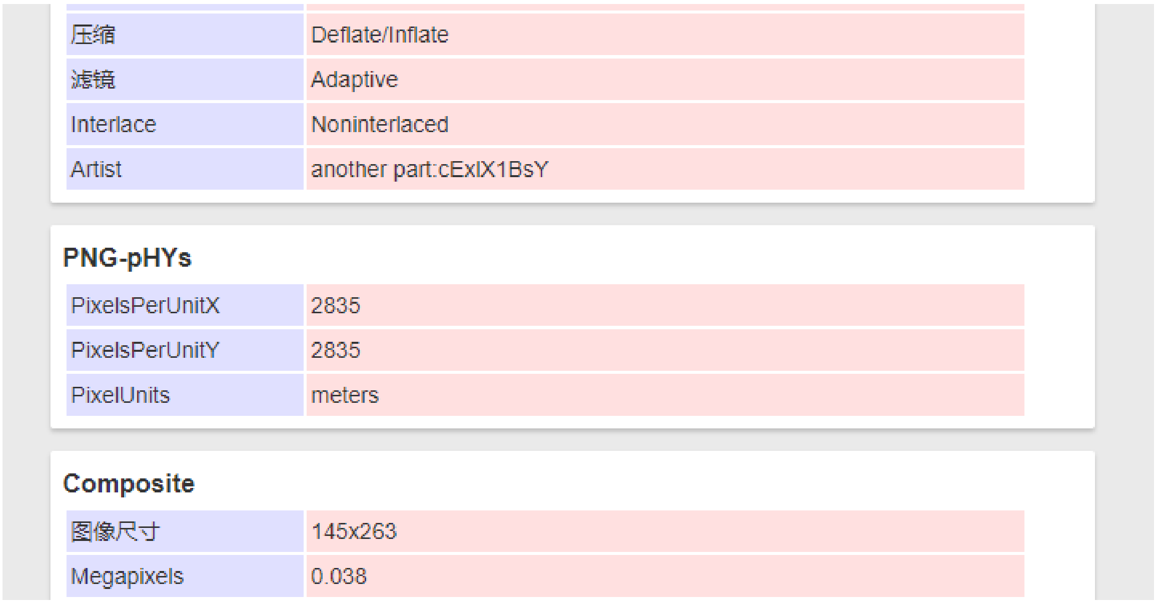
EXIF信息摘要

File

FileType	PNG
FileTypeExtension	png
MIMEType	image/png

PNG

图像宽度	145
图像高度	263
位深	8
色彩类型	RGB



压缩	Deflate/Inflate
滤镜	Adaptive
Interlace	Noninterlaced
Artist	another part:cExlX1BsY

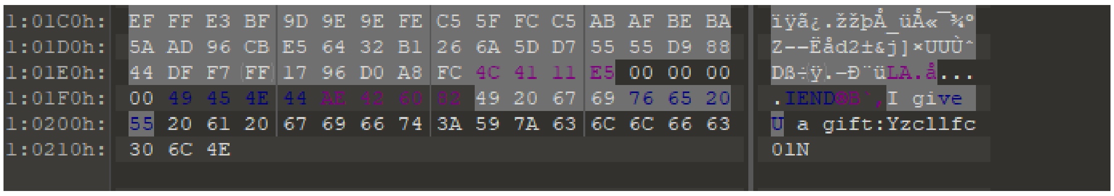
PNG-pHYs

PixelsPerUnitX	2835
PixelsPerUnitY	2835
PixelUnits	meters

Composite

图像尺寸	145x263
Megapixels	0.038

第五张图片也简单了，就在文件末尾`Yzcllfc0lN`



```
1:01C0h: EF FF E3 BF 9D 9E 9E FE C5 5F FC C5 AB AF BE BA  ïÿã¿.žžþÅ_üÅ«¯¾º
1:01D0h: 5A AD 96 CB E5 64 32 B1 26 6A 5D D7 55 55 D9 88  Z–Ëåd2±&j]×UUÙˆ
1:01E0h: 44 DF F7 FF 17 96 D0 A8 FC 4C 41 11 E5 00 00 00  Dß÷ÿ.–Ð¨üLA.å...
1:01F0h: 00 49 45 4E 44 AE 42 60 82 49 20 67 69 76 65 20  .IEND®B`‚I give 
1:0200h: 55 20 61 20 67 69 66 74 3A 59 7A 63 6C 6C 66 63  U a gift:Yzcllfc
1:0210h: 30 6C 4E                                         0lN
```

最后得到的结果按照顺序来，也就是这样子

```
ZmxhZ3s0X3
1RVcmVfc
3RlZ30=
cExlX1BsY
Yzcllfc0lN
```

最后稍微排列组合一下得到最后的结果

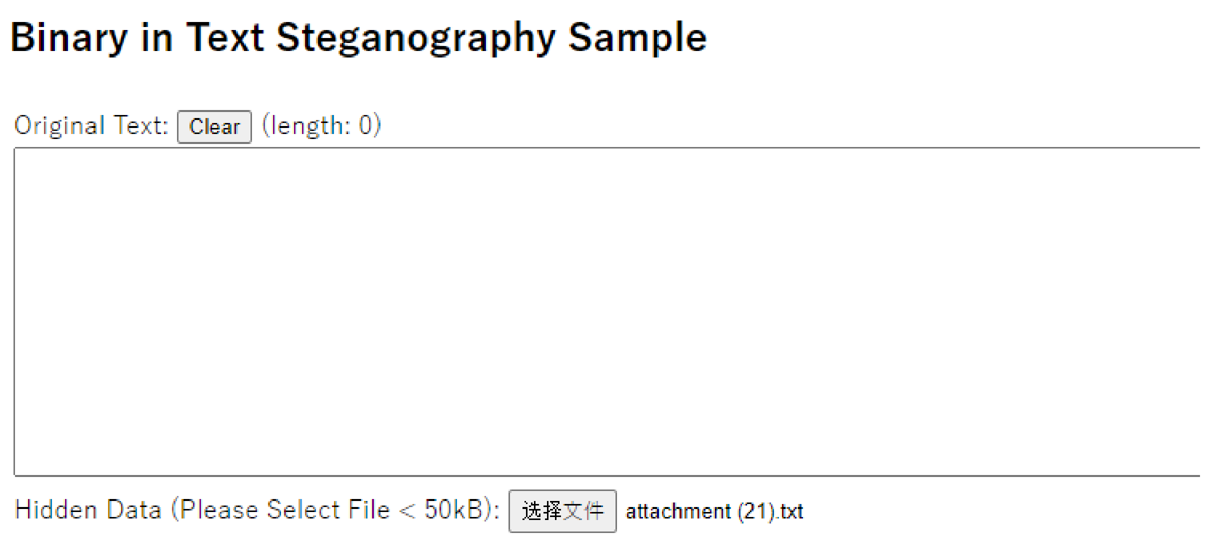
```
ZmxhZ3s0X3Yzcllfc0lNcExlX1BsY1RVcmVfc3RlZ30=
```

进行base64解码得到flag

```
flag{4_v3rY_sIMpLe_PlcTUre_steg}
```

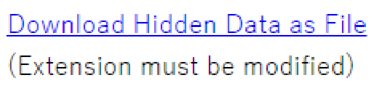
[UTCTF2020]zero

开始还以为是词频分析，但是脚本跑了保存，里面存在一些特殊字符，百度知道这个是叫做`零宽度字符加密`，在线分析网址。
关于这个网站的用法

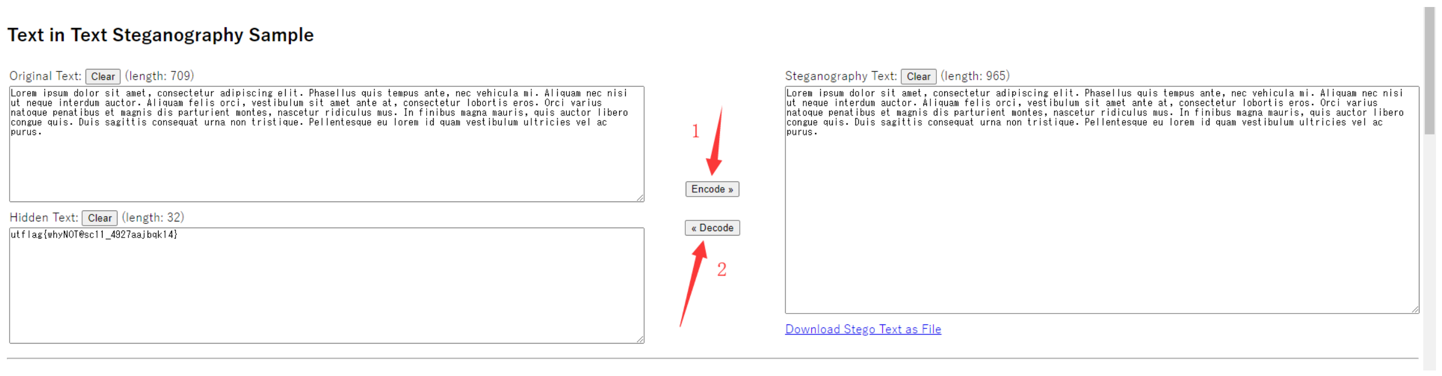


Binary in Text Steganography Sample
Original Text: Clear (length: 0)
Hidden Data (Please Select File < 50kB): 选择文件 attachment (21).txt

我们直接复制字符是不行的，得现在这个地方上传一下，然后点击下方的



再把下载了的这个文件内容复制到上方进行解密，按照步骤来即可



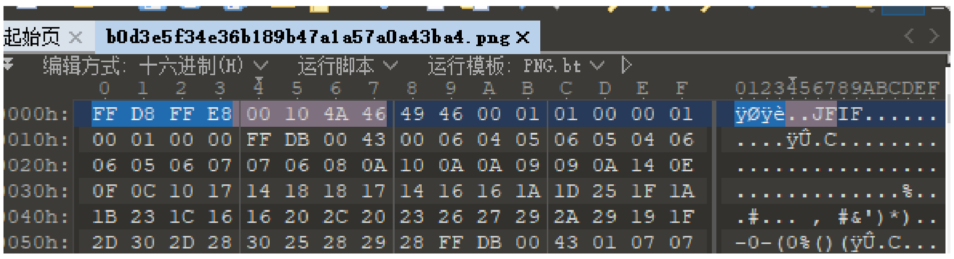
Text in Text Steganography Sample
Original Text: Clear (length: 709)
Hidden Text: Clear (length: 32)
1
Encode »
« Decode
2
Steganography Text: Clear (length: 965)
Download Stego Text as File

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis tempus ante, nec vehicula mi. Aliquam nec nisi
ut neque interdum auctor. Aliquam felis orci, vestibulum sit amet ante at, consectetur lobortis eros. Orci varius
natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In finibus magna mauris, quis auctor libero
congue quis. Duis sagittis consequat urna non tristique. Pellentesque eu lorem id quam vestibulum ultricies vel ac
purus.
```

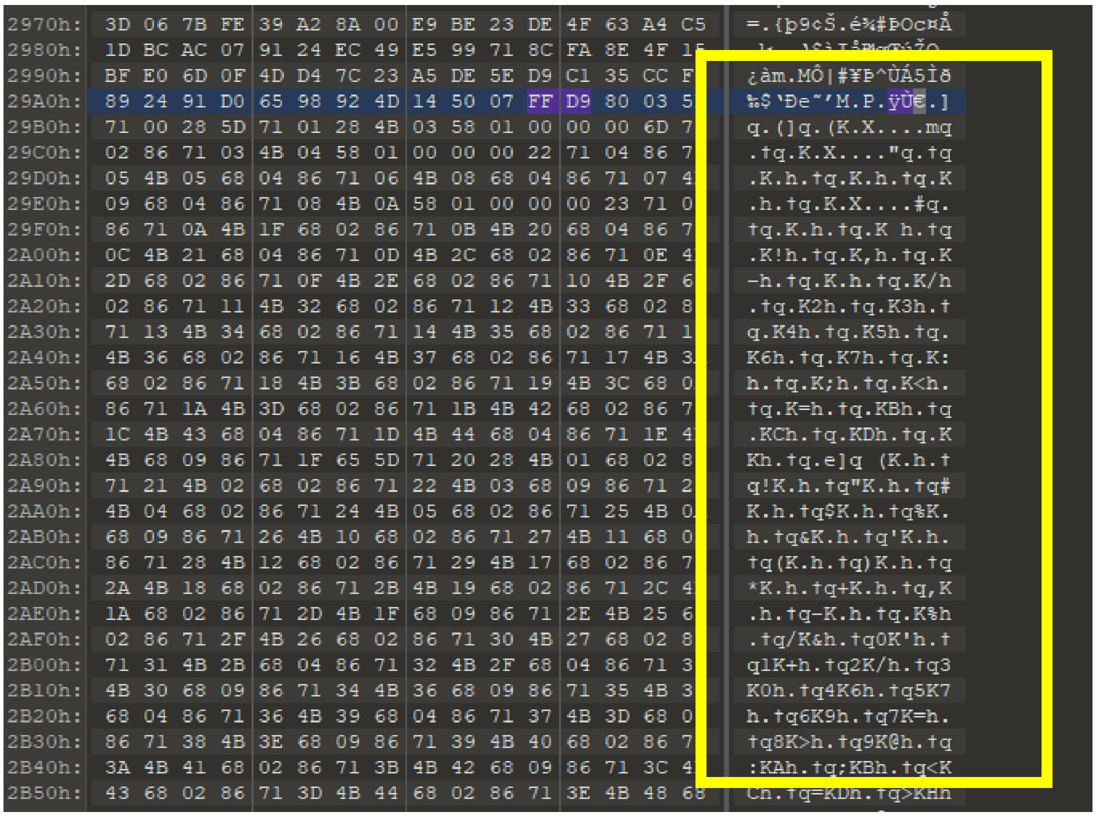
```
utflag{whyNOT@sc11_4927aajbqk14}
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quis tempus ante, nec vehicula mi. Aliquam nec nisi
ut neque interdum auctor. Aliquam felis orci, vestibulum sit amet ante at, consectetur lobortis eros. Orci varius
natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In finibus magna mauris, quis auctor libero
congue quis. Duis sagittis consequat urna non tristique. Pellentesque eu lorem id quam vestibulum ultricies vel ac
purus.
```

我爱linux

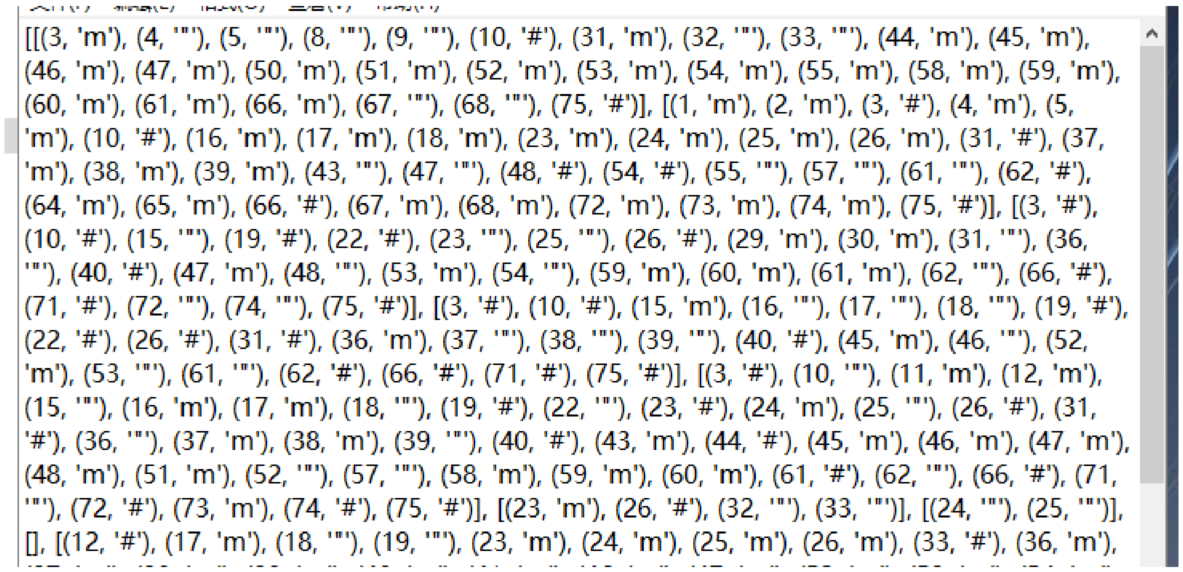

```
起始页 × | b0d3e5f34e36b189b47a1a57a0a43ba4.png ×
编辑方式: 十六进制(H) ∨   运行脚本 ∨   运行模板: PNG.bt ∨  ▷
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: FF D8 FF E8 00 10 4A 46 49 46 00 01 01 00 00 01  ÿØÿè..JFIF......
0010h: 00 01 00 00 FF DB 00 43 00 06 04 05 06 05 04 06  ....ÿÛ.C........
0020h: 06 05 06 07 07 06 08 0A 10 0A 0A 09 09 0A 14 0E  ................
0030h: 0F 0C 10 17 14 18 18 17 14 16 16 1A 1D 25 1F 1A  .............%..
0040h: 1B 23 1C 16 16 20 2C 20 23 26 27 29 2A 29 19 1F  .#... , #&')*)..
0050h: 2D 30 2D 28 30 25 28 29 28 FF DB 00 43 01 07 07  -0-(0%()(ÿÛ.C...
```

修改一下文件头，png->jpg，而且发现在文件末尾，多出了一大段的数据


```
2970h: 3D 06 7B FE 39 A2 8A 00 E9 BE 23 DE 4F 63 A4 C5  =.{þ9¢Š.é¾#ÞOc¤Å
2980h: 1D BC AC 07 91 24 EC 49 E5 99 71 8C FA 8E 4F 15  [illegible]
2990h: BF E0 6D 0F 4D D4 7C 23 A5 DE 5E D9 C1 35 CC F   ¿àm.MÔ|#¥Þ^ÙÁ5Ìð
29A0h: 89 24 91 D0 65 98 92 4D 14 50 07 FF D9 80 03 5   ‰$‘Ðe˜’M.P.ÿÙ€.]
29B0h: 71 00 28 5D 71 01 28 4B 03 58 01 00 00 00 6D 7   q.(]q.(K.X....mq
29C0h: 02 86 71 03 4B 04 58 01 00 00 00 22 71 04 86 7   .†q.K.X...."q.†q
29D0h: 05 4B 05 68 04 86 71 06 4B 08 68 04 86 71 07 4   .K.h.†q.K.h.†q.K
29E0h: 09 68 04 86 71 08 4B 0A 58 01 00 00 00 23 71 0   .h.†q.K.X....#q.
29F0h: 86 71 0A 4B 1F 68 02 86 71 0B 4B 20 68 04 86 7   †q.K.h.†q.K h.†q
2A00h: 0C 4B 21 68 04 86 71 0D 4B 2C 68 02 86 71 0E 4   .K!h.†q.K,h.†q.K
2A10h: 2D 68 02 86 71 0F 4B 2E 68 02 86 71 10 4B 2F 6   -h.†q.K.h.†q.K/h
2A20h: 02 86 71 11 4B 32 68 02 86 71 12 4B 33 68 02 8   .†q.K2h.†q.K3h.†
2A30h: 71 13 4B 34 68 02 86 71 14 4B 35 68 02 86 71 1   q.K4h.†q.K5h.†q.
2A40h: 4B 36 68 02 86 71 16 4B 37 68 02 86 71 17 4B 3   K6h.†q.K7h.†q.K:
2A50h: 68 02 86 71 18 4B 3B 68 02 86 71 19 4B 3C 68 0   h.†q.K;h.†q.K<h.
2A60h: 86 71 1A 4B 3D 68 02 86 71 1B 4B 42 68 02 86 7   †q.K=h.†q.KBh.†q
2A70h: 1C 4B 43 68 04 86 71 1D 4B 44 68 04 86 71 1E 4   .KCh.†q.KDh.†q.K
2A80h: 4B 68 09 86 71 1F 65 5D 71 20 28 4B 01 68 02 8   Kh.†q.e]q (K.h.†
2A90h: 71 21 4B 02 68 02 86 71 22 4B 03 68 09 86 71 2   q!K.h.†q"K.h.†q#
2AA0h: 4B 04 68 02 86 71 24 4B 05 68 02 86 71 25 4B 0   K.h.†q$K.h.†q%K.
2AB0h: 68 09 86 71 26 4B 10 68 02 86 71 27 4B 11 68 0   h.†q&K.h.†q'K.h.
2AC0h: 86 71 28 4B 12 68 02 86 71 29 4B 17 68 02 86 7   †q(K.h.†q)K.h.†q
2AD0h: 2A 4B 18 68 02 86 71 2B 4B 19 68 02 86 71 2C 4   *K.h.†q+K.h.†q,K
2AE0h: 1A 68 02 86 71 2D 4B 1F 68 09 86 71 2E 4B 25 6   .h.†q-K.h.†q.K%h
2AF0h: 02 86 71 2F 4B 26 68 02 86 71 30 4B 27 68 02 8   .†q/K&h.†q0K'h.†
2B00h: 71 31 4B 2B 68 04 86 71 32 4B 2F 68 04 86 71 3   q1K+h.†q2K/h.†q3
2B10h: 4B 30 68 09 86 71 34 4B 36 68 09 86 71 35 4B 3   K0h.†q4K6h.†q5K7
2B20h: 68 04 86 71 36 4B 39 68 04 86 71 37 4B 3D 68 0   h.†q6K9h.†q7K=h.
2B30h: 86 71 38 4B 3E 68 09 86 71 39 4B 40 68 02 86 7   †q8K>h.†q9K@h.†q
2B40h: 3A 4B 41 68 02 86 71 3B 4B 42 68 09 86 71 3C 4   :KAh.†q;KBh.†q<K
2B50h: 43 68 02 86 71 3D 4B 44 68 02 86 71 3E 4B 48 68  Ch.†q=KDh.†q>KHh
```

提取出来，但是看不懂，搜了下发现图片内容是《巨蟒与飞行马戏团》的图片，而py也是由于这个来的，猜测是pickle的序列化之后的数据，逆回去看看

```python
import pickle

f1 = open('1.txt','rb')
f2 = open('2.txt','a')
f2.write(str(pickle.load(f1)))
f1.close()
f2.close()
```



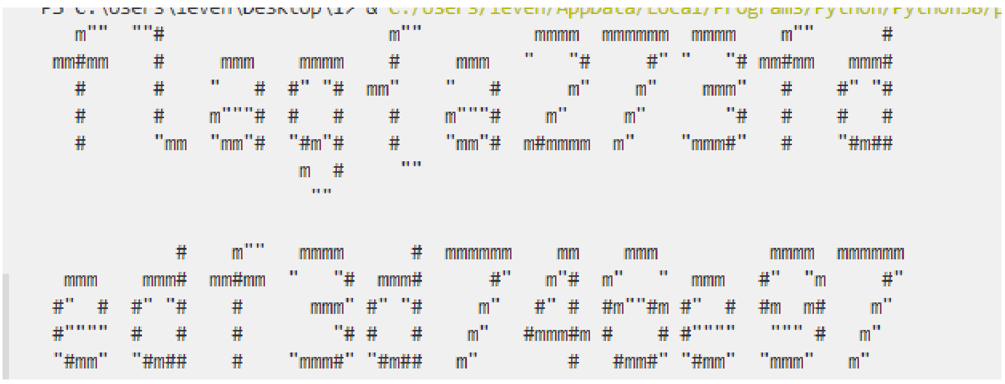
```
[[(3, 'm'), (4, '"'), (5, '"'), (8, '"'), (9, '"'), (10, '#'), (31, 'm'), (32, '"'), (33, '"'), (44, 'm'), (45, 'm'),
(46, 'm'), (47, 'm'), (50, 'm'), (51, 'm'), (52, 'm'), (53, 'm'), (54, 'm'), (55, 'm'), (58, 'm'), (59, 'm'),
(60, 'm'), (61, 'm'), (66, 'm'), (67, '"'), (68, '"'), (75, '#')], [(1, 'm'), (2, 'm'), (3, '#'), (4, 'm'), (5,
'm'), (10, '#'), (16, 'm'), (17, 'm'), (18, 'm'), (23, 'm'), (24, 'm'), (25, 'm'), (26, 'm'), (31, '#'), (37,
'm'), (38, 'm'), (39, 'm'), (43, '"'), (47, '"'), (48, '#'), (54, '#'), (55, '"'), (57, '"'), (61, '"'), (62, '#'),
(64, 'm'), (65, 'm'), (66, '#'), (67, 'm'), (68, 'm'), (72, 'm'), (73, 'm'), (74, 'm'), (75, '#')], [(3, '#'),
(10, '#'), (15, '"'), (19, '#'), (22, '#'), (23, '"'), (25, '"'), (26, '#'), (29, 'm'), (30, 'm'), (31, '"'), (36,
'"'), (40, '#'), (47, 'm'), (48, '"'), (53, 'm'), (54, '"'), (59, 'm'), (60, 'm'), (61, 'm'), (62, '"'), (66, '#'),
(71, '#'), (72, '"'), (74, '"'), (75, '#')], [(3, '#'), (10, '#'), (15, 'm'), (16, '"'), (17, '"'), (18, '"'), (19, '#'),
(22, '#'), (26, '#'), (31, '#'), (36, 'm'), (37, '"'), (38, '"'), (39, '"'), (40, '#'), (45, 'm'), (46, '"'), (52,
'm'), (53, '"'), (61, '"'), (62, '#'), (66, '#'), (71, '#'), (75, '#')], [(3, '#'), (10, '"'), (11, 'm'), (12, 'm'),
(15, '"'), (16, 'm'), (17, 'm'), (18, '"'), (19, '#'), (22, '"'), (23, '#'), (24, 'm'), (25, '"'), (26, '#'), (31,
'#'), (36, '"'), (37, 'm'), (38, 'm'), (39, '"'), (40, '#'), (43, 'm'), (44, '#'), (45, 'm'), (46, 'm'), (47, 'm'),
(48, 'm'), (51, 'm'), (52, '"'), (57, '"'), (58, 'm'), (59, 'm'), (60, 'm'), (61, '#'), (62, '"'), (66, '#'), (71,
'"'), (72, '#'), (73, 'm'), (74, '#'), (75, '#')], [(23, 'm'), (26, '#'), (32, '"'), (33, '"')], [(24, '"'), (25, '"')],
[], [(12, '#'), (17, 'm'), (18, '"'), (19, '"'), (23, 'm'), (24, 'm'), (25, 'm'), (26, 'm'), (33, '#'), (36, 'm'),
```

得到一大串元组坐标

可以发现，元组的第一个元素是位置，第二个元素是字符，于是画一下，脚本如下

```python
import pickle

with open('1.txt', 'rb') as f:
    data = pickle.load(f)
new_data = list()
for i in range(len(data)):
    tmp = [' ']*100
    new_data.append(tmp)
for i,d in enumerate(data):
    for m in d:
        new_data[i][m[0]] = m[1]
for i in new_data:
    print(''.join(i))
```

得到flag

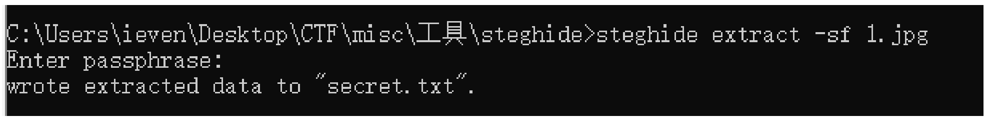


```
PS C:\Users\leven\Desktop\1> & C:/Users/leven/AppData/Local/Programs/Python/Python38/p
  m""  ""#                   m""          mmmm  mmmmmm  mmmm    m""      #
mm#mm    #     mmm    mmmm   #     mmm   "   "#     #" "   "# mm#mm   mmm#
  #      #    "   #  #" "# mm"    "   #      m"    m"    mmm"   #    #" "#
  #      #    m"""#  #   #   #    m"""#    m"     m"       "#   #    #   #
  #      "mm  "mm"#  "#m"#   #    "mm"#  m#mmmm  m"    "mmm#"   #    "#m##
                      m  #    ""
                       ""


           #    m""   mmmm      # mmmmmm    mm    mmm          mmmm  mmmmmm
 mmm    mmm#  mm#mm  "   "#  mmm#     #"   m"#  m"   "  mmm   #"  "m     #"
#"  #  #" "#    #      mmm" #" "#    m"   #" #  #m""#m #"  #  #m  m#    m"
#""""  #   #    #        "# #   #   m"   #mmm#m #    # #""""   """ #   m"
"#mm"  "#m##    #    "mmm#" "#m##  m"        #   #mm#" "#mm"  "mmm"   m"
```

这个其实也就是通过linux下面的`toilet`命令输出的

[ACTF新生赛2020]剑龙

先把js解密，得到`welcom3!`

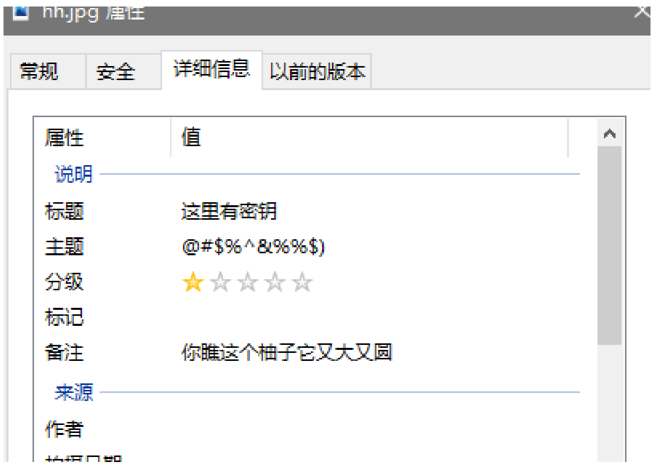


```
C:\Users\ieven\Desktop\CTF\misc\工具\steghide>steghide extract -sf 1.jpg
Enter passphrase:
wrote extracted data to "secret.txt".
```

得到

```
想要flag吗？解出我的密文吧~
U2FsdGVkX1/7KeHVl5984OsGUVSanPfPednHpK9lKvp0kdrxO4Tj/Q==
```

一看就知道是DES了，现在去找密码，然后在图片的详细信息里面发现了密码`@#$%^&%%$)`

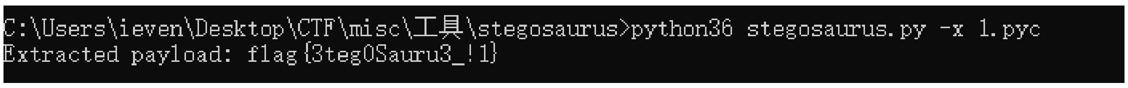


hh.jpg 属性

常规 | 安全 | 详细信息 | 以前的版本

属性	值
说明	
标题	这里有密钥
主题	@#$%^&%%$)
分级	★☆☆☆☆
标记	
备注	你瞧这个柚子它又大又圆
来源	
作者	

解密得到`think about stegosaurus`，在github上面找到对应的东西，网址。
跑一下得到flag，注意需要使用python36



```
C:\Users\ieven\Desktop\CTF\misc\工具\stegosaurus>python36 stegosaurus.py -x 1.pyc
Extracted payload: flag{3teg0Sauru3_!1}
```