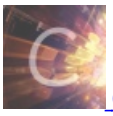
CTF 隐写 关于zlib的解压 2018全国大学生信息安全竞赛 picture

原创

never疯 于 2018-05-02 12:51:24 发布 26791 收藏 59
分类专栏： CTF 隐写 文章标签： CTF 网络安全 隐写术



CTF 隐写 关于zlib的解压 2018全国大学生信息安全竞赛 picture

相信大家在做隐写时 都会遇到.png图片里有.zlib
那么如果遇到这种情况怎么处理 ， 我将会用2018全国信息安全竞赛 picture 这道题来给大家举例讲解

首先我们需要先认识Python的几个函数

binascii.hexlify 和 binascii.unhexlify
binascii.hexlify() 将单个字符转化成对应码值的十六进制格式（\xxx)格式
binascii.hexlify('hello')
————————>输出 '68656c6c6f'

binascii.unhexlify() 两个字符为单位，看作十六进制值，转化成对应码值的字符格式
binascii.unhexlify('68656c6c6f')
————————>输出 hello

encode和decode
"hello".encode("hex")
————————>输出'68656c6c6f'

'68656c6c6f'.decode("hex")
————————>输出'hello'

zlib.decompress和zlib.compress
zlib.compress(str)用于压缩流数据
zlib.decompress(str)用于解压数据。

将这个三个函数仔细看了后,我们来做题

全国大学生信息安全大赛 picture

这是一道隐写术的题 我把照片放上来

先用linux下binwalk分析图片

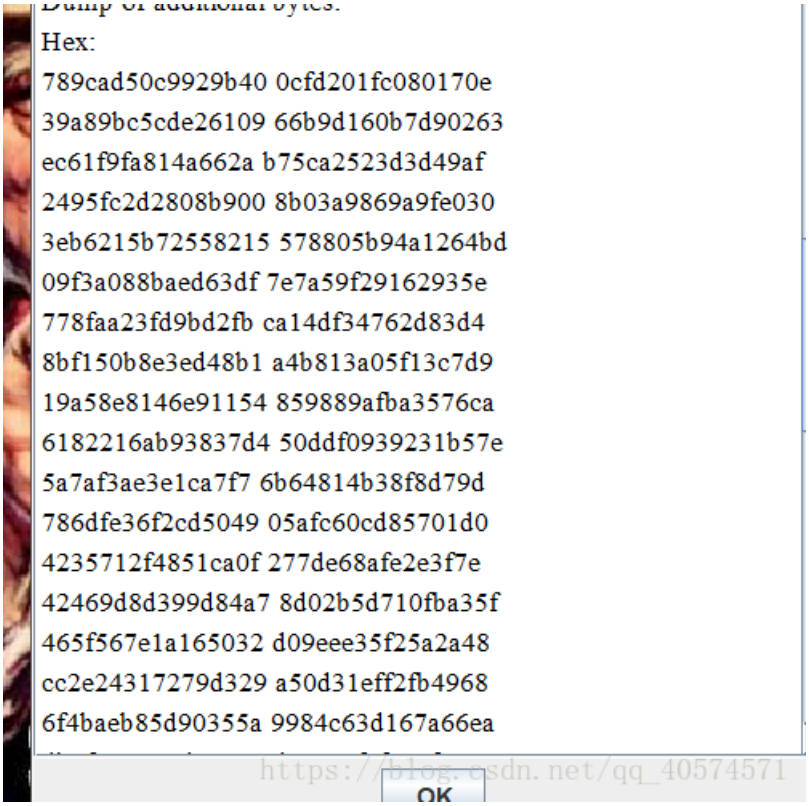
```
root@kali:~# binwalk task-ctf_06_tgF28nL.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
38884         0x97E4          Zlib compressed data, default compression
```

beef xss framework

发现文件里有zlib格式的文件
下面我们需要进行分离,分离之前我们必须知道.jpg格式文件开头结尾 .zlib文件开头结尾 我的另一篇文章里有介绍

那么接下里 我们打开winhex 或者 setgolve 查看

```
097C0  40 05 14 51 40 05 14 51  40 05 14 51 40 05 14 51  @  Q@  Q@  Q@  Q
097D0  40 05 14 51 40 05 14 51  40 05 14 51 40 05 14 51  @  Q@  Q@  Q@  Q
097E0  40 1F FF D9 78 9C AD 50  C9 92 9B 40 0C FD 20 1F  @ ýÙxœ-PÉ' >@ ý
097F0  C0 80 17 0E 39 A8 9B C5  CD E2 61 09 66 B9 D1 60  À€  9¨›ÅÍâa f¹Ñ`
09800  B7 D9 02 63 EC 61 F9 FA  81 4A 66 2A B7 5C A2 52  ·Ù cìaùú Jf*·\¢R
09810  3D 3D 49 AF 24 95 FC 2D  28 08 B9 00 8B 03 A9 86  ==I¯$•ü-( ¹ ‹ ©†
09820  9A 9F E0 30 3E B6 21 5B  72 55 82 15 57 88 05 B9  šŸà0>¶![rU, W^ ¹
09830  4A 12 64 BD 09 F3 A0 88  BA ED 63 DF 7E 7A 59 F2  J d½ ó ^ºícß~zYò
09840  91 62 93 5E 77 8F AA 23  FD 9B D2 FB CA 14 DF 34  'b"^w ª#ý›ÒûÊ ß4
09850  76 2D 83 D4 8B F1 50 B8  E3 ED 48 B1 A4 B8 13 A0  v-ƒÔ‹ñP¸ãíH±¤¸
09860  5F 13 C7 D9 19 A5 8E 81  46 E9 11 54 85 98 89 AF  _ ÇÙ ¥Ž Fé T…˜‰¯
09870  BA 35 76 CA 61 82 21 6A  B9 38 37 D4 50 DD F0 93  º5vÊa‚!j¹87ÔPÝð"
09880  92 31 B5 7E 5A 7A F3 AE  3E 1C A7 F7 6B 64 81 4B  '1µ~Zzó®> §÷kd K
09890  38 F8 D7 9D 78 6D FE 36  F2 CD 50 49 05 AF C6 0C  8ø× xmþ6òÍPI ¯Æ
098A0  D8 57 01 D0 42 35 71 2F  48 51 CA 0F 27 7D E6 8A  ØW ÐB5q/HQÊ '}æŠ
098B0  FE 2E 3F 7E 42 46 9D 8D  39 9D 84 A7 8D 02 B5 D7  þ.?~BF  9 „§  µ×
098C0  10 FB A3 5F 46 5F 56 7E  1A 16 50 32 D0 9E EE 35  ûf_F_V~  P2ÐžÎ5
098D0  F2 5A 2A 48 CC 2E 24 31  72 79 D3 29 A5 0D 31 EF  òZ*HÌ.$1ryÓ)¥ 1ï
098E0  F2 FB 49 68 6F 4B AE B8  5D 90 35 5A 99 84 C6 3D  òûIhoK®¸] 5Z™„Æ=
098F0  16 7A 66 EA DB 8F 4C 74  59 D6 5C 96 DA 8E 27 FA  zfêÛ LtYÖ\-ÚŽ'ú
09900  F9 4E F5 81 51 5D 9B 73  5C ED 15 B7 65 44 4D 3A  ùNõ Q]›s\í ·eDM:
09910  AA 07 8C 94 E3 70 8D CE  6D 12 8E 3D 99 61 43 8A  ª Œ”ãp Îm Ž=™aCŠ
09920  55 57 BF 12 1F 46 0B A3  8E 16 70 A4 A1 5C E5 E1  UW¿  F £Ž p¤¡\åá
09930  58 3B 8C 37 09 06 F6 F7  EE B0 B9 4C 94 F7 BA BC  X;Œ7  ö÷î°¹L”÷º¼
09940  A9 E7 34 94 5F 5E 64 4C  54 24 FB 6F 4D FA 9F A2  ©ç4”_^dLT$ûoMúŸ¢
09950  0F 47 CB 47 43 1C 9D E7  5C 90 A7 04 C3 C1 FC FA  GËGC  ç\ § ÃÁüú
09960  05 C0 8F 4F 15 A7 C2 CE                          À O §ÂÎ
```

两种方法都能找到zlib的位置 那么接下来我用三种方法进行解压

1. 1.

```
from zlib import *
data = open('task-ctf_06_tgF28nL.png', 'rb').read()[0x97E4:]
print data
data = decompress(data)
print data
```

这是解压代码,第一种是直接对图片进行偏移读取,直接读到zlib,其实正常步骤是分理出来改成.zlib查看里面是否有文件,但是我在试过后发现没有，那我就再对zlib文件进行解压查看,了解下是什么

2.

```
import zlib
import binascii
import base64
id='789cad50c9929b400cfd201fc080170e39a89bc5cde2610966b9d160b7d90263ec61f9fa814a662ab75ca2523d3d49af249
result =binascii.unhexlify(id)
print result
result = zlib.decompress(result)
print result
```

这是第二种用的是binascii.unhexlify()函数

3.

```
import zlib
import binascii

IDAT = "789cad50c9929b400cfd201fc080170e39a89bc5cde2610966b9d160b7d90263ec61f9fa814a662ab75ca2523d3d49a
print IDAT
result = zlib.decompress(IDAT)
print result
```

这是第三种方法用的是str.decode（‘hex’）

以上三种方法输出结果是一样的

```
1���IhoK��]�5Z���=zf□LtY�\□'��N��Q]�s\��eDM:����p��m=�aC�UW�F��p��\��X;�7
S1ADBBQAAQAAAIkwl0yA7xr1WgAAAE4AAAAEAAAAY29kZZBlO2zwD3GMSCSMtRcZvaCKbe5rkpIsODsSDyYfFgejUaRYCwiQxf8bC4D
```

我只是想告诉大家这三种方法的作用一样，因为网上的关于这个打代码各种各样，刚开始比较难理解
大家可能说这输出的是什么玩意 也看不懂
这道题第二个输出就是我们想看到，这道题第二个输出是一个base64的字符，这个怎么看出来的我没法告诉你，大概是你看多了慢慢就明白了

网上还有一些题接出来是3031这些

所以实际的数据是:

```
789C5D91011280400802BF04FFFF5C75294B5537738A21A27D1E49CFD17DB3937A92E7E603880A6D485100901FB0410153350DE83112EA2D51C54
CE2E585B15A2FC78E8872F51C6FC1881882F93D372DEF78E665B0C36C529622A0A45588138833A170A2071DDCD18219DB8C0D465D8B6989719645
ED9C11C36AE3ABDAEFCFC0ACF023E77C17C7897667
```

然后用python来写zlib解压

```
#!  /usr/bin/env  python
import  zlib
import  binascii
IDAT  =  "789C5D91011280400802BF04FFFF5C75294B5537738A21A27D1E49CFD17DB3937A92E7E603880A6D485100901FB0410153350DE83112
EA2D51C54CE2E585B15A2FC78E8872F51C6FC1881882F93D372DEF78E665B0C36C529622A0A45588138833A170A2071DDCD18219DB8C0D465D8B6
989719645ED9C11C36AE3ABDAEFCFC0ACF023E77C17C7897667".decode('hex')
#print  IDAT
result  =  binascii.hexlify(zlib.decompress(IDAT))
print  result

#print  result.decode('hex')
```

发现解出来了一些3031的字符串，30和31是hex的 0和1的编码，再解一次hex得到一串625长度的01字符串。

```
1111111000100001101111111100000101110010110100000110111010100000000010111011011101001000000001011101101110101110110101010
0101110110000010101011011010000011111111010101010101111111000000001011101110000000011010011000001010011101101111101010
1001000011100000000000101000000000100100110100010011100111101110011110000111011111100011001010001100111000010101000110
1000111101011000001010001011000001101110110010000111001110010000101111111010000000011010100100011110111111101110000110
1011011100000100001100110001111010111010001101001111100001011101011000111010011100101110100100111011011000110000010110
00011010001100011111110110101101110111011
```

得到的01 串的长度是625，除以8 除以7 都无法整除，也就是说没法直接转换成ascii码。

```
>>> 625.0/8.0
78.125
>>> 625.0/7.0
89.28571428571429
```

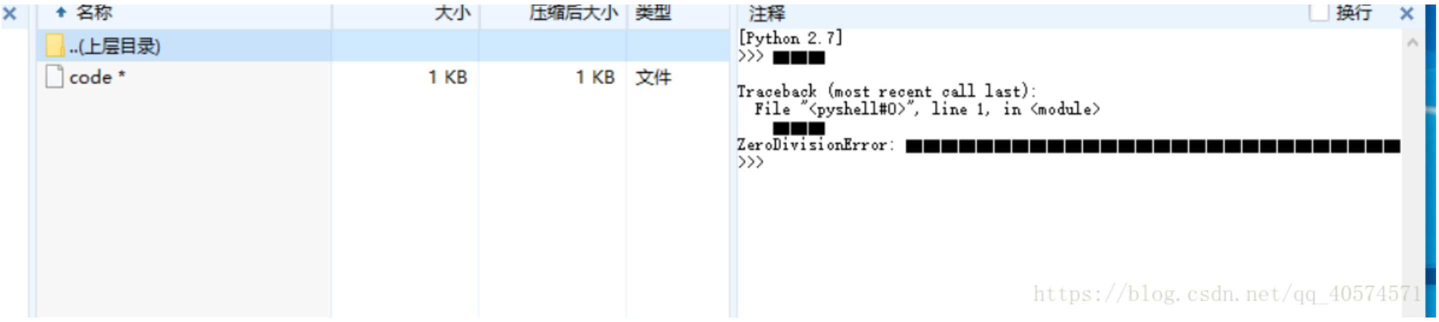
然后发现625 = 25*25，刚好是个正方形的形状，那么尝试一下 把这些01 组成一个正方形 看看是什么，可以用python的PIL编程可以很方便的画图，在kali自带就可以有，win的环境需要安装PIL的第三方库。

像这种题就是解出来是什么样再进行什么操作,这种30313031
他又再次进行解编码 最后形成一个二维码

咱们这个是base64,那么当然再用base64解下

```
import zlib
import binascii
import base64
id='789cad50c9929b400cfd201fc080170e39a89bc5cde2610966b9d160b7d90263ec61f9fa814a662ab75ca2523d3d49af249
result =binascii.unhexlify(id)
print result
result = zlib.decompress(result)
print result
result = base64.b64decode(result)
print result
fount = open(r"55.zlib","wb")
fount.write(result)
fount.close()
```

解压成一个zip文件后发现解压问题

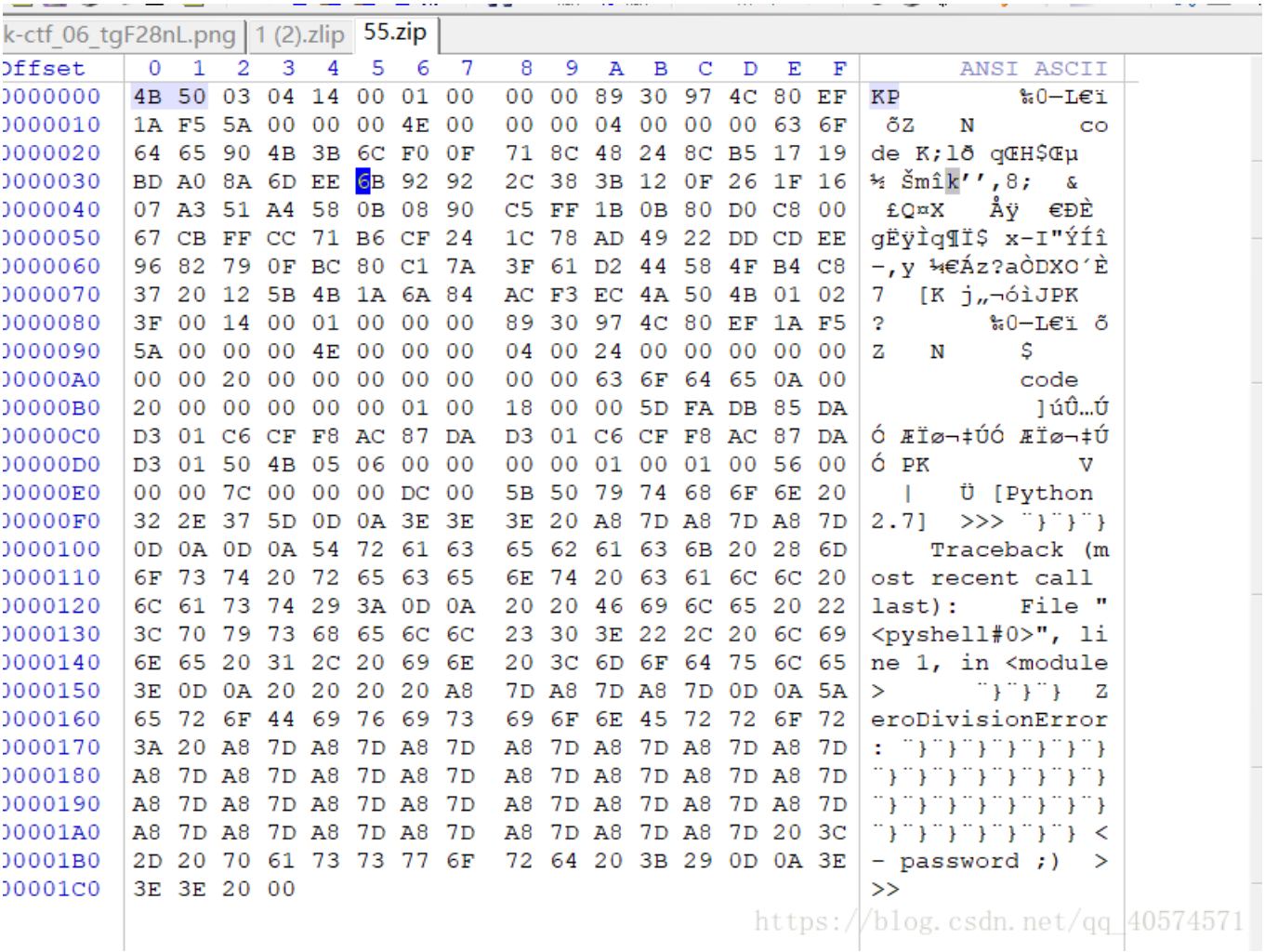


那个问题原因password已经给你了
他想说的是ZeroDivisionError对应的是什么
百度是分母为零
那么自己操作一遍

```
root@kali:~# python
Python 2.7.14 (default, Sep 17 2017, 18:50:44)
[GCC 7.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>>
```


拿到password

然后既然zip打不开,那就去找原因,怀疑对文件头尾做了手脚

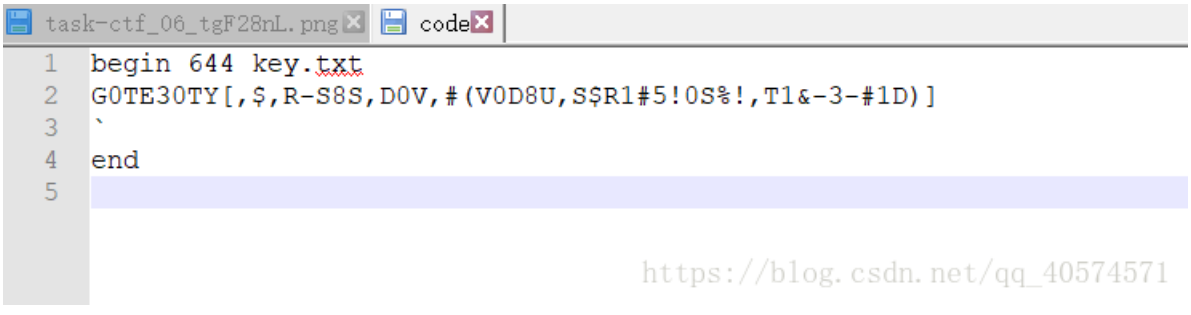


k-ctf_06_tgF28nL.png | 1 (2).zip | 55.zip

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
0000000	4B	50	03	04	14	00	01	00	00	00	89	30	97	4C	80	EF	KP %0—L€ï
0000010	1A	F5	5A	00	00	00	4E	00	00	00	04	00	00	00	63	6F	õZ N co
0000020	64	65	90	4B	3B	6C	F0	0F	71	8C	48	24	8C	B5	17	19	de K;lð qŒH$Œµ
0000030	BD	A0	8A	6D	EE	6B	92	92	2C	38	3B	12	0F	26	1F	16	½ Šmîk’’,8; &
0000040	07	A3	51	A4	58	0B	08	90	C5	FF	1B	0B	80	D0	C8	00	£Q¤X Åÿ €ÐÈ
0000050	67	CB	FF	CC	71	B6	CF	24	1C	78	AD	49	22	DD	CD	EE	gËÿÌq¶Ï$ x-I"ÝÍî
0000060	96	82	79	0F	BC	80	C1	7A	3F	61	D2	44	58	4F	B4	C8	-‚y ¼€Áz?aÒDXO´È
0000070	37	20	12	5B	4B	1A	6A	84	AC	F3	EC	4A	50	4B	01	02	7 [K j„¬óìJPK
0000080	3F	00	14	00	01	00	00	00	89	30	97	4C	80	EF	1A	F5	? %0—L€ï õ
0000090	5A	00	00	00	4E	00	00	00	04	00	24	00	00	00	00	00	Z N $
00000A0	00	00	20	00	00	00	00	00	00	00	63	6F	64	65	0A	00	code
00000B0	20	00	00	00	00	00	01	00	18	00	00	5D	FA	DB	85	DA	]úÛ…Ú
00000C0	D3	01	C6	CF	F8	AC	87	DA	D3	01	C6	CF	F8	AC	87	DA	Ó ÆÏø¬‡ÚÓ ÆÏø¬‡Ú
00000D0	D3	01	50	4B	05	06	00	00	00	00	01	00	01	00	56	00	Ó PK V
00000E0	00	00	7C	00	00	00	DC	00	5B	50	79	74	68	6F	6E	20	| Ü [Python
00000F0	32	2E	37	5D	0D	0A	3E	3E	3E	20	A8	7D	A8	7D	A8	7D	2.7] >>> ¨}¨}¨}
0000100	0D	0A	0D	0A	54	72	61	63	65	62	61	63	6B	20	28	6D	Traceback (m
0000110	6F	73	74	20	72	65	63	65	6E	74	20	63	61	6C	6C	20	ost recent call
0000120	6C	61	73	74	29	3A	0D	0A	20	20	46	69	6C	65	20	22	last): File "
0000130	3C	70	79	73	68	65	6C	6C	23	30	3E	22	2C	20	6C	69	<pyshell#0>", li
0000140	6E	65	20	31	2C	20	69	6E	20	3C	6D	6F	64	75	6C	65	ne 1, in <module
0000150	3E	0D	0A	20	20	20	20	A8	7D	A8	7D	A8	7D	0D	0A	5A	> ¨}¨}¨} Z
0000160	65	72	6F	44	69	76	69	73	69	6F	6E	45	72	72	6F	72	eroDivisionError
0000170	3A	20	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	: ¨}¨}¨}¨}¨}¨}
0000180	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	¨}¨}¨}¨}¨}¨}¨}
0000190	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	¨}¨}¨}¨}¨}¨}¨}
00001A0	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	A8	7D	20	3C	¨}¨}¨}¨}¨}¨} <
00001B0	2D	20	70	61	73	73	77	6F	72	64	20	3B	29	0D	0A	3E	- password ;) >
00001C0	3E	3E	20	00													>>

果然没错,更改成50 4B

可以打开了 将password输入 得到key.txt



task-ctf_06_tgF28nL.png | code

```
1  begin 644 key.txt
2  G0TE30TY[,$,R-S8S,D0V,#(V0D8U,S$R1#5!0S%!,T1&-3-#1D)]
3  `
4  end
5
```


能看出来又是一个编码 用 UUEncode 解出flag

你问我为啥是UUEncode 我也不知道,看多了就知道了吧,我也是队友告诉的