

xctf re新手题

原创

Team  于 2021-02-28 15:10:20 发布  71  收藏 3

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 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/weixin_53409153/article/details/114218904

版权

Hello, CTF

首先，下载附件后，拖入IDA32中，然后shift+f12，会发现

	.rdata:0...	00000013	C	GetLastActivePopup	
	.rdata:0...	00000010	C	GetActiveWindow	
	.rdata:0...	0000000C	C	MessageBoxA	
	.rdata:0...	0000000B	C	user32.dll	
	.rdata:0...	0000000D	C	KERNEL32.dll	
	.data:00...	00000008	C	wrong!\n	
	.data:00...	0000000A	C	success!\n	
	.data:00...	0000001A	C	please input your serial:	
	.data:00...	00000023	C	437261636b4d654a757374466f7246756e	
	.data:00...	00000006	C	\t-\r]	https://blog.csdn.net/weixin_53409153
	.data:00...	00000006	C	粒冢	

我们看见这串字符，应该就会有所感觉，他就是flag，然后：

跳转——交叉列表——F5查看伪代码，然后就有：

```
signed int v3; // ebx
char v4; // al
int result; // eax
int v6; // [esp+0h] [ebp-70h]
int v7; // [esp+0h] [ebp-70h]
char v8; // [esp+12h] [ebp-5Eh]
char v9[20]; // [esp+14h] [ebp-5Ch]
char v10; // [esp+28h] [ebp-48h]
__int16 v11; // [esp+48h] [ebp-28h]
char v12; // [esp+4Ah] [ebp-26h]
char v13; // [esp+4Ch] [ebp-24h]

strcpy(&v13, "437261636b4d654a757374466f7246756e");
while ( 1 )
{
    memset(&v10, 0, 0x20u);
    v11 = 0;
    v12 = 0;
    sub_40134B(aPleaseInputYou, v6);
    scanf(aS, v9);
    if ( strlen(v9) > 0x11 )
        break;
    v3 = 0;
    do
    {
        v4 = v9[v3];
        if ( !v4 )
            break;
        sprintf(&v8, asc_408044, v4);
        strcat(&v10, &v8);
        ++v3;
    }
    while ( v3 < 17 );
```

strcpy, 即string copy (字符串复制) 的缩写。strcpy是一种C语言的标准库函数, strcpy把含有'\0'结束符的字符串复制到另一个地址空间, 返回值的类型为char*。

我们观察发现, 他有数字和字母组成, 切最大的字母是f, 所以优先考虑16进制转换:

16进制到ASCII字符串在线转换工具

```
1 437261636b4d654a757374466f7246756e
```

```
2
```

🗑️ 清空

↕ 交换位置

📄 示例

🔄 转换

💾 保存结果

📄 复制结果

```
1 CrackMeJustForFun
```

https://blog.csdn.net/weixin_53409153

我们得到的就是flag。

CrackMeJustForFun

[open-sourc](#)

re逆向新手题中的第一道题目对于我们这些新手而言还是有一点难度的，主要就是得看得懂代码的意思。

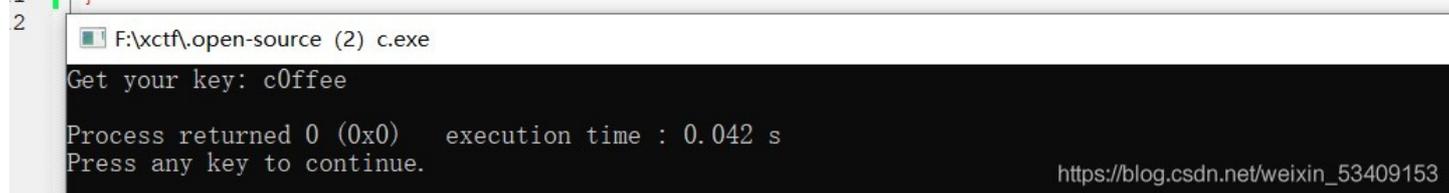
下载后得到代码，然后打开，进行分析注释有

```
1 #include <stdio.h>
2 #include <string.h>
3 int main(int argc, char *argv[])
4 {
5     if (argc != 4)
6     {
7         printf("what?\n");
8         exit(1);
9     }
10    //循环得到argc=4
11    unsigned int first = atoi(argv[1]);
12    if (first != 0xcafe)
13    {
14        printf("you are wrong, sorry.\n");
15        exit(2);
16    }
17    //如果不满足条件就退出,则first=0xcafe
18    unsigned int second = atoi(argv[2]);
19    if (second % 5 == 3 || second % 17 != 8)
20    {
21        printf("ha, you won't get it!\n");
22        exit(3);
23    }
24    //如果满足条件就退出,则令second=25
25    if (strcmp("h4cky0u", argv[3]))
26    {
27        printf("so close, dude!\n");
28        exit(4);
29    }
30    //strcmp与argv[3]相等,则突出循环argv[3]="h4cky0u"
31    printf("Brr wxxx gxx\n");
32
33    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;
34
35    printf("Get your key: ");
36    printf("%x\n", hash);
37    return 0;
38 }
```

https://blog.csdn.net/weixin_53409153

在得到自己的理解以后，对代码进行重新编辑，则有：

```
1 #include <stdio.h>
2 #include <string.h>
3 int main()
4 {
5     /*unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;*/
6     unsigned int hash = 0xcafe * 31337 + (25 % 17) * 11 + strlen("h4cky0u") - 1615810207;
7
8     printf("Get your key: ");
9     printf("%x\n", hash);
10    return 0;
11 }
```



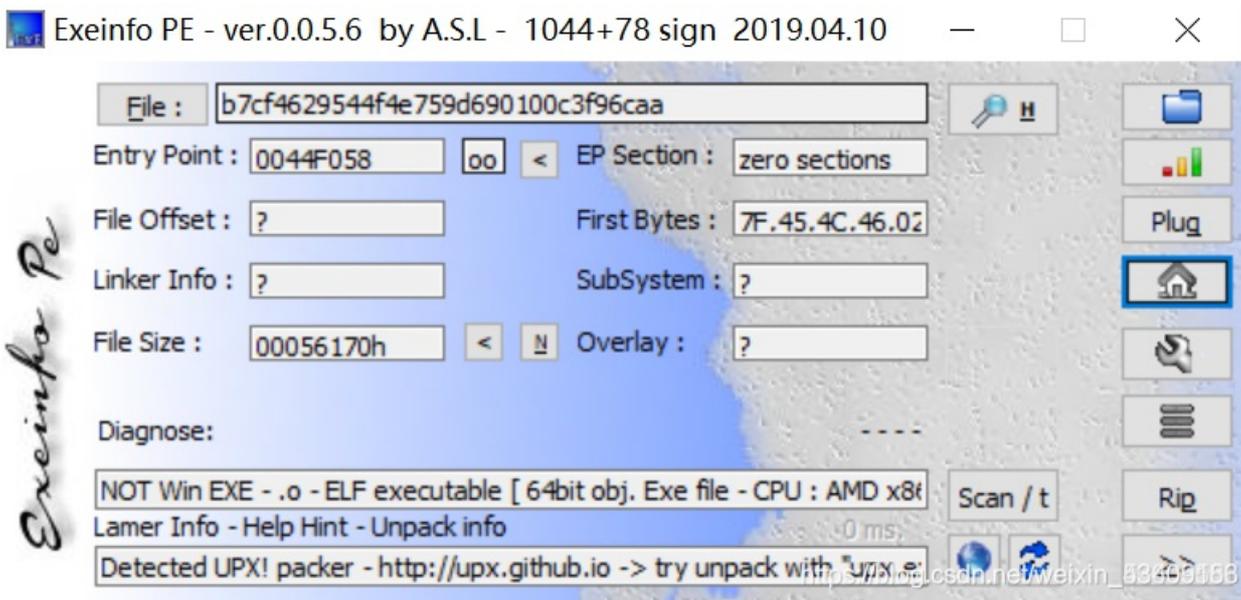
https://blog.csdn.net/weixin_53409153

将得到的flag为c0ffee输入即可。

(其中不知道的C语言知识可以在网上查找一下，我也是找了好久才搞明白的。)

simple-unpack

首先，题目的介绍，我们可以知道，我们最开始还是需要进行检查的：



发现文件是64位的，然后使用IDA64，shift+f12

`.- b7cf4629544f4e759d690100c3f96caa C:\Users\Administrator\Downloads\b7cf4629544f4e759d690100c3f96caa`

编辑(E) 跳转(J) 搜索(H) 视图(V) 调试器 选项(O) 窗口(W) 帮助



flag{Upx_1s_n0t_a_d3liv3r_c0mp4n}

logmein

首先，下载文件，改为exe后缀，查壳，发现是64位，然后用IDA64打开，shift+f12有：

地址	长度	类型	字符串
LOAD:000...	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:000...	0000000A	C	libc.so.6
LOAD:000...	00000005	C	exit
LOAD:000...	0000000F	C	__isoc99_scanf
LOAD:000...	00000007	C	printf
LOAD:000...	00000007	C	strlen
LOAD:000...	00000012	C	__libc_start_main
LOAD:000...	0000000F	C	__gmon_start__
LOAD:000...	0000000A	C	GLIBC_2.7
LOAD:000...	0000000C	C	GLIBC_2.2.5
.rodata:...	0000002D	C	Welcome to the RC3 secure password guesser.\n
.rodata:...	00000033	C	To continue, you must enter the correct password.\n
.rodata:...	00000013	C	Enter your guess:
.rodata:...	00000005	C	%32s
.rodata:...	00000015	C	Incorrect password!\n
.rodata:...	0000002E	C	You entered the correct password!\nGreat job!\n
.eh_frame...	00000006	C	.*3\$\n

https://blog.csdn.net/weixin_53409153

发现没有想要的flag，然后找到main函数，f5查看伪代码，有：

函数名称	段
__init_proc	.init
sub_4004C0	.plt
__strlen	.plt
__printf	.plt
__libc_start_main	.plt
__isoc99_scanf	.plt
__exit	.plt
__gmon_start__	.plt
start	.text
sub_400560	.text
sub_4005E0	.text
sub_400600	.text
main	.text
sub_4007C0	.text
sub_4007F0	.text
init	.text
fini	.text
__term_proc	.fini
__strlen	.exte
__printf	.exte
__libc_start_main	.exte
__isoc99_scanf	.exte
__exit	.exte

```

1 void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
2 {
3     size_t v3; // rsi
4     int i; // [rsp+3Ch] [rbp-54h]
5     char s[36]; // [rsp+40h] [rbp-50h]
6     int v6; // [rsp+64h] [rbp-2Ch]
7     __int64 v7; // [rsp+68h] [rbp-28h]
8     char v8[8]; // [rsp+70h] [rbp-20h]
9     int v9; // [rsp+8Ch] [rbp-4h]
10
11     v9 = 0;
12     strcpy(v8, ":\\"AL_RT^L*.?+6/46");
13     v7 = 28537194573619560LL;
14     v6 = 7;
15     printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
16     printf("To continue, you must enter the correct password.\n");
17     printf("Enter your guess: ");
18     __isoc99_scanf("%32s", s);
19     v3 = strlen(s);
20     if ( v3 < strlen(v8) )
21         sub_4007C0(v8);
22     for ( i = 0; i < strlen(s); ++i )
23     {
24         if ( i >= strlen(v8) )
25             ((void (*)(void))sub_4007C0)();
26         if ( s[i] != (char)((_BYTE *)&v7 + i % v6) ^ v8[i] )
27             ((void (*)(void))sub_4007C0)();
28     }
29     sub_4007F0();
30 }

```

https://blog.csdn.net/weixin_53409153

然后，我们可以再

```
if ( s[i] != (char)((_BYTE *)&v7 + i % v6) ^ v8[i] )
sub_4007C0();
```

找flag在这里我们可以使用Python来写，脚本

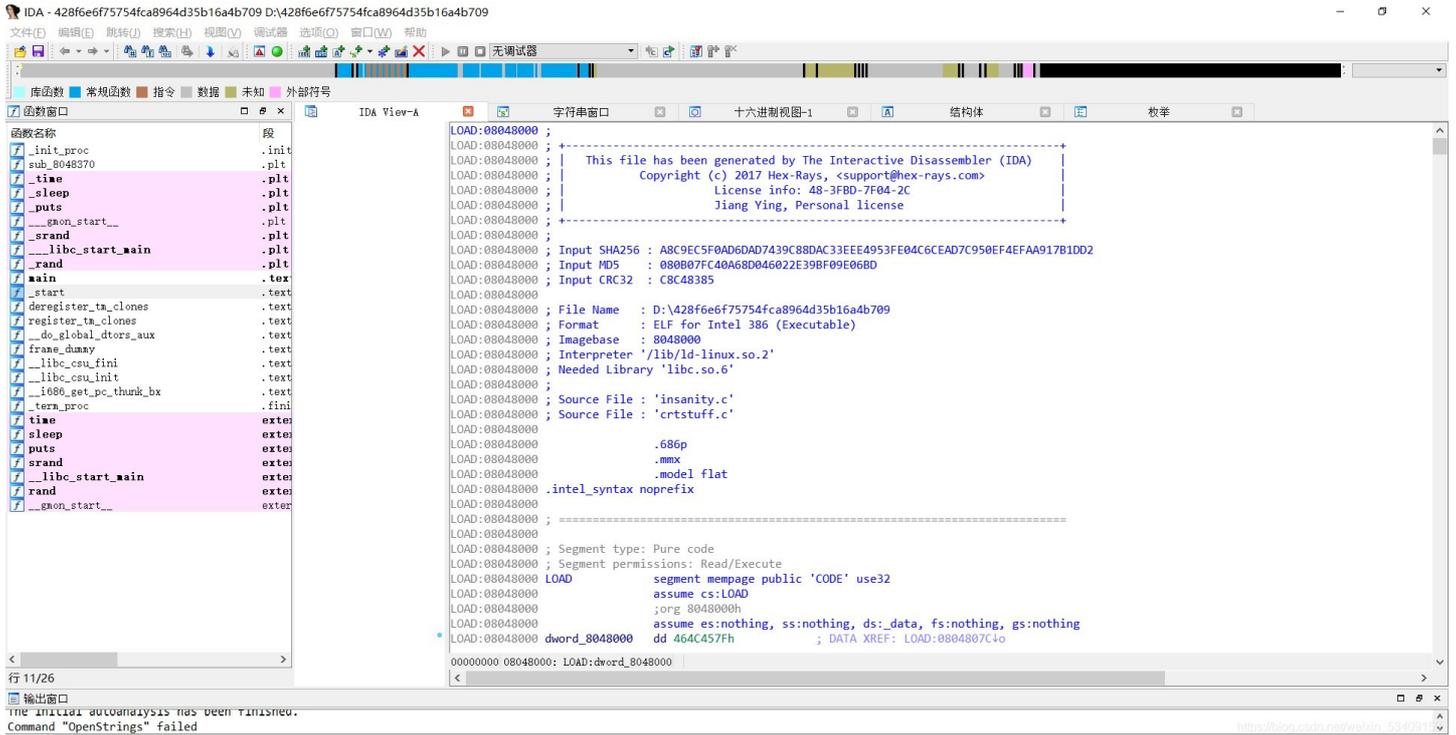
```

a = ":\\"AL_RT^L*.?+6/46"
b = "harambe"
c = 7
flag = ""
for i in range(0,len(a)):
flag += chr(ord(b[i%c]) ^ ord(a[i]))
print(flag)

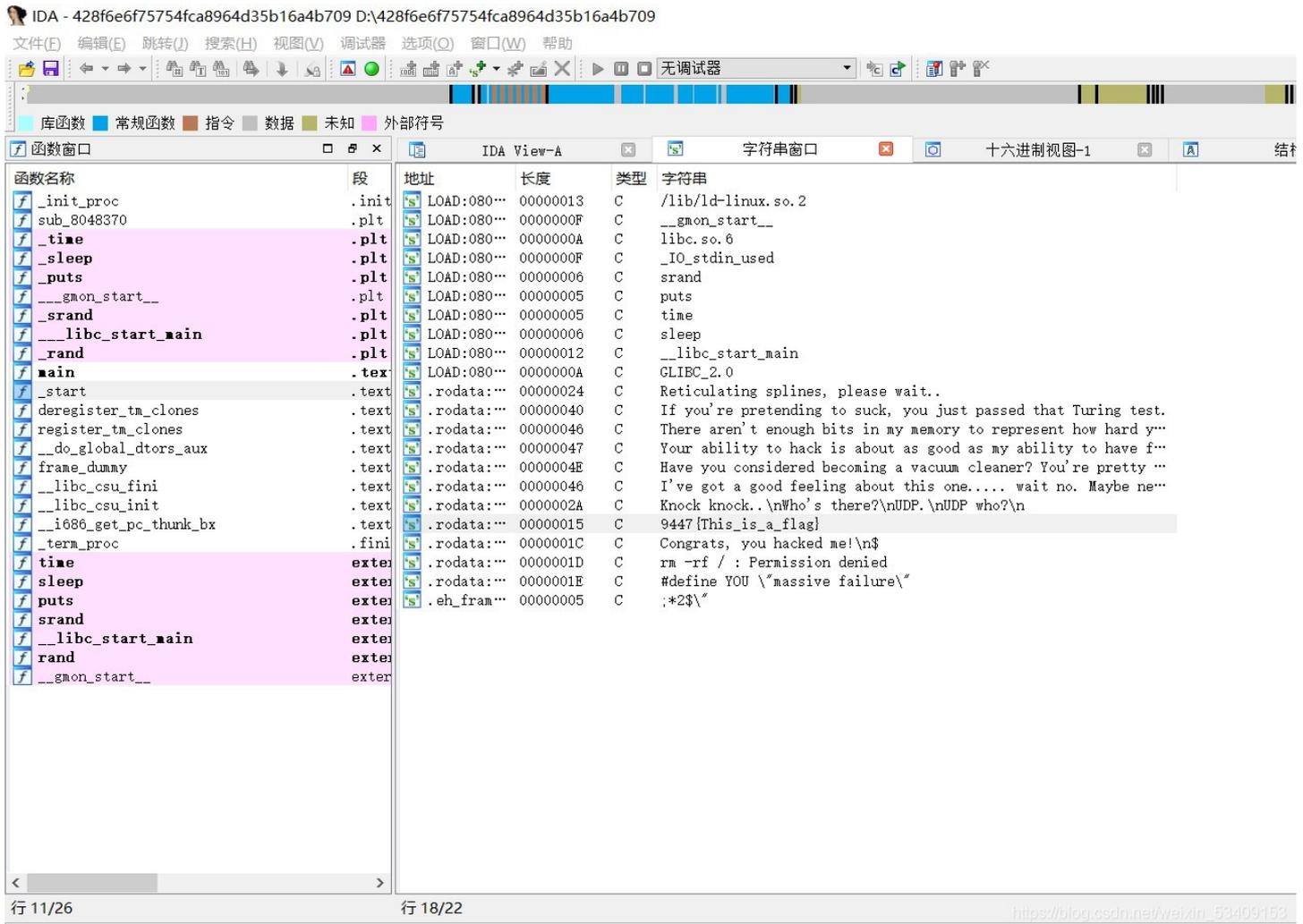
```

得到：RC3-2016-XORISGUD

下载附件后发现，这个附件我们无法打开，这个时候我们使用exeinfo打开该附件，查看它的外壳，我们会发现它是32位的，我们再用IDA32打开，得到：



这个时候我们还是无法发现flag，然后我们在进行shift+f12，来进行操作（shift+f12：可以打开string窗口，一键找出所有的字符串，右击setup，还能对窗口的属性进行设置）这个时候得到：



这个时候我们仔细观察就会发现想要的flag

9447{This_is_a_flag}

getit

首先下载附件后，用IDA64打开，然后shift+f12得到：

```
LOAD:00000000 00000000 C __smulsi@4.1.0
LOAD:00000000 0000000A C GLIBC_2.4
LOAD:00000000 0000000C C GLIBC_2.2.5
.eh_frame00000006 C ;*3$`
.data:00000000 00000021 C c61b68366edeb7bdce3c6820314b7498
.data:00000000 0000002B C harifCTF{????????????????????????????????}
.data:00000000 0000002C C *****
```

然后，点击进入，在s处进入交叉引用：

```
.data:000000000000010A0 s char s[]
.data:000000000000010A0 s db 'c61b68366edeb7bdce3c6820314b7498',0
.data:000000000000010A0 ; DATA XREF: main+25fo
.data:000000000000010A0 ; main+25fo
```

进入以后会有伪代码：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v3; // a1
    __int64 v5; // [rsp+0h] [rbp-40h]
    int i; // [rsp+4h] [rbp-3Ch]
    FILE *stream; // [rsp+8h] [rbp-38h]
    char filename[8]; // [rsp+10h] [rbp-30h]
    unsigned __int64 v9; // [rsp+28h] [rbp-18h]

    v9 = __readfsqword(0x28u);
    LODWORD(v5) = 0;
    while ( (signed int)v5 < strlen(s) )
    {
        if ( v5 & 1 )
            v3 = 1;
        else
            v3 = -1;
        *(&t + (signed int)v5 + 10) = s[(signed int)v5] + v3;
        LODWORD(v5) = v5 + 1;
    }
    strcpy(filename, "/tmp/flag.txt");
    stream = fopen(filename, "w");
    fprintf(stream, "%s\n", u, v5);
    for ( i = 0; i < strlen(&t); ++i )
    {
        fseek(stream, p[i], 0);
        fputc(*(&t + p[i]), stream);
        fseek(stream, 0LL, 0);
        fprintf(stream, "%s\n", u);
    }
    fclose(stream);
    remove(filename);
    return 0;
}
```

https://blog.csdn.net/weixin_53409153

然后用Python写一段脚本有：

```
#!/usr/bin/env python
2 t = 'sharifCTF{????????????????????????????????}'
3 t1 = list(t)
```

```
4 s1 = 'c61b68366eдеб7bdce3c6820314b7498'  
5 v5 = 0  
6  
7 for i in range(0,32):  
8     if(v5 & 1):  
9         v3 = 1  
10    else:  
11        v3 = -1  
12  
13    t1[(v5+10)] = chr(ord(s1[v5]) + v3)  
14    v5 += 1  
15 flag = ''.join(t1)  
16 print(flag)  
17
```

https://blog.csdn.net/weixin_53409153

得到SharifCTF{b70c59275fcfa8aebf2d5911223c6589}

上面就是flag

标题

```
#!/usr/bin/env python
# encoding: utf-8
import base64

def encode(message):
    s = ''
    for i in message:
        x = ord(i) ^ 32
        x = x + 16
        s += chr(x)

    return base64.b64encode(s)

correct = 'XlnkVmtUI1MgXWBZXCFeKY+AaXNt'
flag = ''
print 'Input flag:'
flag = raw_input()
if encode(flag) == correct:
    print 'correct'
else:
    print 'wrong'
import base64
```

https://blog.csdn.net/weixin_53409153

这是生成的py文件

然后，对这个文件的运算逻辑进行逆向

写EXP

```
1  #-*- coding:utf-8 -*-
2  import base64
3
4  last_flag = 'XlnkVmtUI1MgXWBZXCFeKY+AaXNt'
5  first_flag = base64.b64decode(last_flag)
6  print "This is base64-decode message!!:"
7  print first_flag
8
9  right_flag = ''
10 for i in first_flag:
11     right_flag += chr((ord(i) - 16)^32)
12
13 print "This is the last flag:"
14 print right_flag
```

先对字符串进行base64,然后，再进行异或运算得到最后的flag:nctf{d3c0mpil1n9_PyC}