HGAME Week1 WriteUp

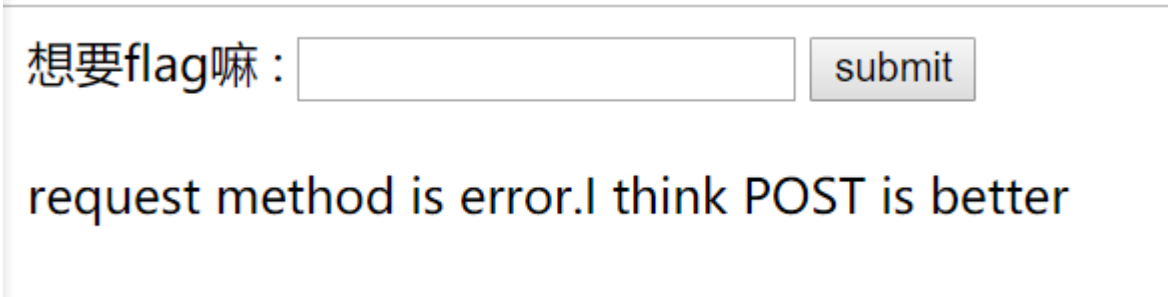
WEB

谁吃了我的flag

根据hint，Mki使用vim编写题目并且非正常关闭，根据vim的特性，vim编辑器，在terminal被意外关闭时，会产生.swp文件 。于是将地址栏的 **index.html**改为**.index.html.swp**,下载得到swp文件，打开获得flag:hgame{3eek_diScl0Sure_fRom+wEbsit@}

换头大作战

这个头应该是HTTP的请求头了，打开网站，输入want



按f12，

发现method是“get"，于是改成“post”，输入”want“然后submit，

出现了学习资料链接和一句：only localhost can get flag

查了下资料，然后再问问百度，于是后退，用burpsuite抓包

结合localhost，输入X-Forwarded-For:127.0.0.1，然后Go

```
POST /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
Content-Length: 5
Cache-Control: max-age=0
Origin: http://120.78.184.111:8080
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.110 Safari/537.36
X-Forwarded-For:127.0.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://120.78.184.111:8080/week1/how/index.php?want=
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: admin=0
Connection: close

want=
```

```
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 583

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>换头大作战</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
  <script src="main.js"></script>
</head>
<body>
<form action="index.php" method="get">
  想要flag嘛：<input name="want" type="text">
  <input type="submit" value="submit">
</form>
</body>
</html>

<br/>https://www.wikiwand.com/en/User_agent<br/>please use Waterfox/50.0
```

拉到最下面，发现please use Waterfox/50.0

于是在左边的User-Agent:加上/Waterfox/50.0，然后Go

接着出现the requests should referer from www.bilibili.com

于是将左边Referer:内容替换成www.bilibili.com，然后Go

又出现了you are not admin

看看左边，admin=0，于是把0换成1，拿到flag：hgame{hTTp_HeaDeR_iS_Ez}

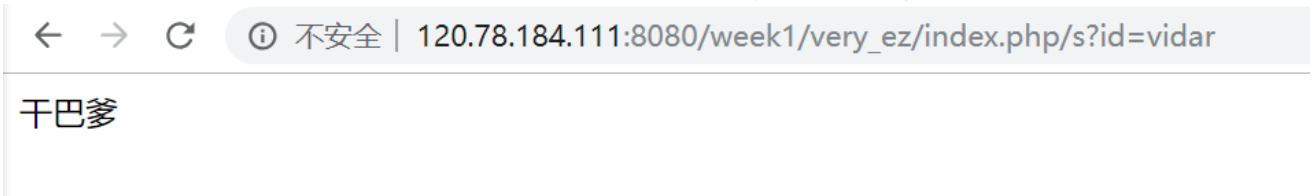
（感觉有点不知所以的写完这一题，很想知道这背后的原理的是啥，出题人是怎么操作的？）

very easy web

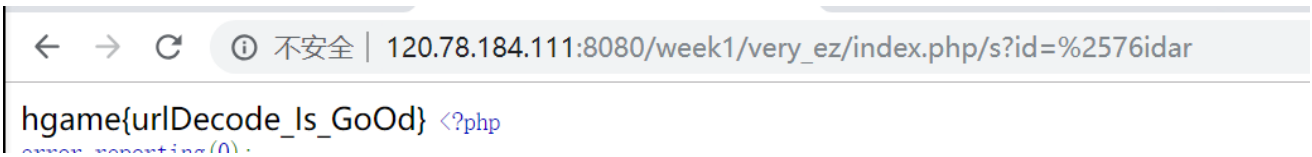
打开网页，一眼看到干巴爹。嗯？！

然后看到了get，那么可以在地址栏解决了。看程序，首先一个判断语句，id不能是vidar，然后，将id解码一次，id是vidar，那么拿到flag。

那就，那就把v编码呗，在地址栏输入/s?id=%76idar,回车（真easy啊，等着flag跳出来）。干巴爹？！什么鬼。



一看地址栏发现，id变成了vidar，大概是浏览器会将我的输入自动解码一次。那么，76没办法编码了，编码%吧，在地址栏输入/s?id=%2576idar,回车



拿到flag：hgame{urlDecode_Is_GoOd}

（看地址栏，怎么还是%2576呢，不该是%76么，是因为页面没跳转么？）

can u find me?

打开网页，

the gate has been hidden

can you find it? xixixi

f12，发现有个

```
<a href="f12.php"></a>
```

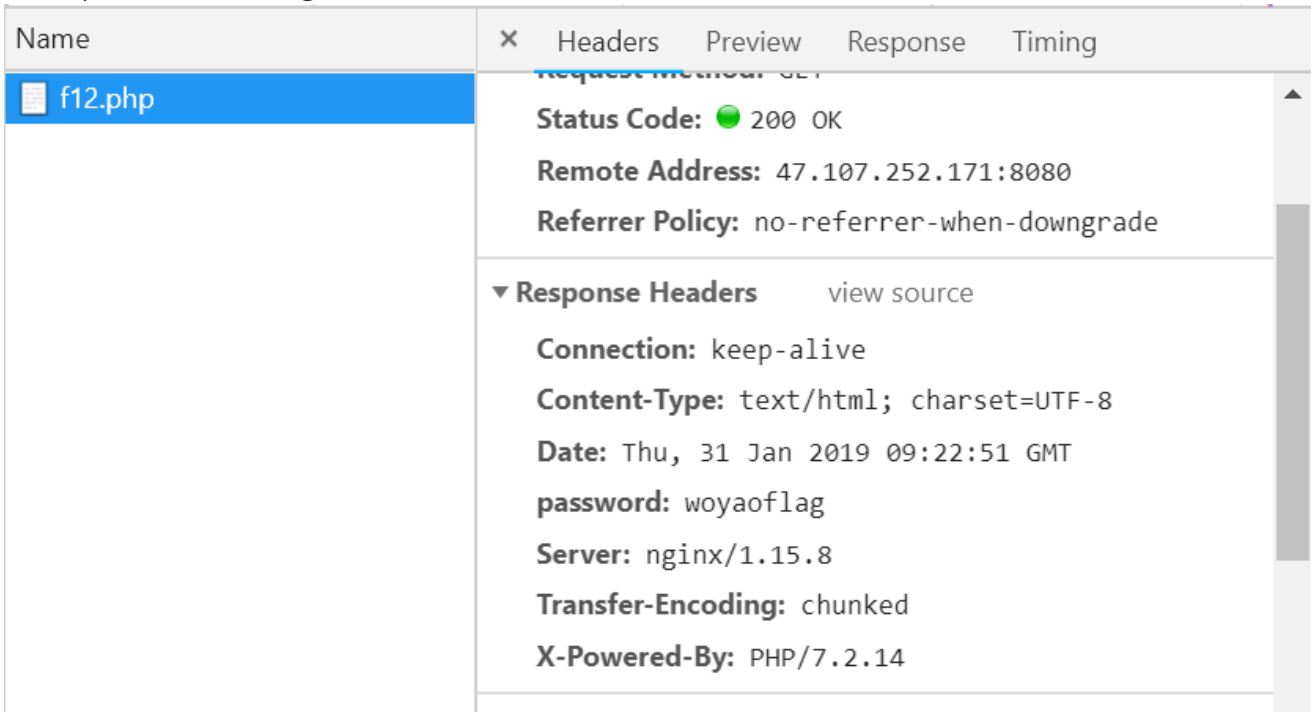
于是在地址栏键入/f12.php

yeah!you find the gate

but can you find the password?

please post password to me! I will open the gate for you!

首先找password，用Google自带的抓包，很容易就找到了



然后写一个post

```
<form action="http://47.107.252.171:8080/f12.php"method="post">
    <input type="submit"name="password"value="woyaoflag">
</form>
```

然后post过去：

yeah!you find the gate

but can you find the password?

please post password to me! I will open the gate for you!

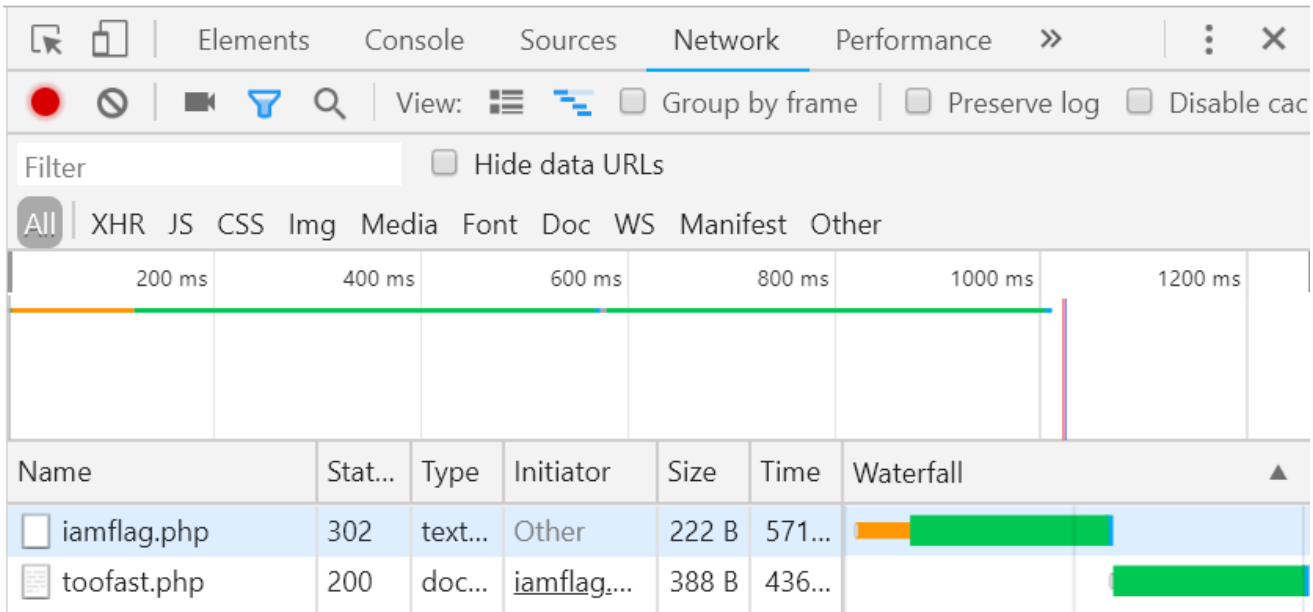
right!

click me to get flag

那么单击（有这么简单？要给直接给啊）

aoh,your speed is sososo fast,the flag must have been left in somewhere

（果然）sososo fast，flag被落下了？！



从Google的抓包，可以看到有个iamflag.php，Type是text，那么肯定是真flag了，status是302，被跳过了，所以不被看见，那么，后退，用burpsuite抓吧。

Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 302 Found
Server: nginx/1.15.8
Date: Thu, 31 Jan 2019 09:40:10 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.2.14
location: toofast.php
Content-Length: 132

<html>
    <head>
        <title>can you find me?</title>
    </head>
    <body>
        <p>flag:hgame{f12_1s_aMazIng111}</p>
    </body>
</html>
```

拿到flag：hgame{f12_1s_aMazIng111}

RE

brainfxxker

（好名字）逻辑题了，

代码

```
",>++++++++++[<---------->-]<++[+.],>+++++++++[<--------->-]<-[+.],>+++++++[<------->-]<---
[+.],>++++++[<------>-]<+++[+.],>++++++++[<---------->-]<++[+.],>++++++++++[<---------->-]<--
[+.],>++++++++++[<--------->-]<-----[+.],>++++++++++[<---------->-]<+[+.],>+++++++++[<--------->-]
<---[+.]"
```

说明：不执行[+.]这部分

查了一下brainfxxker，比较特别的，就是：**]：如果指针指向的单元值不为零，向前跳转到对应的 [指令的次一指令处**

以及 **[： 如果指针指向的单元值为零，向后跳转到对应的] 指令的次一指令处**

那么这题代码的逻辑就是，【以第一个[+.]及之前代码为例】输入一个数（应该是一个字母的ascii码，学长ascii码的链接都给了，，，）然后到下一位，加10（整个数组被初始化过，全为0，那么此时此处为10），然后回到原来的数，减10，再过去，10减1，不是零，那么回到第一个 [后的内容，回到那个被减10的数，再减10，，，，以此类推，总共减了10次，减了100，然后跳出循环，再回到那个（被减的很惨的）数，给它加2，此时如果要跳过[+.]那么这个数应该为0，所以，这个最初被输入的数（字母的ascii码）为10*10-2=98，查表，是b（一共九个逗号，代表输入了9次，所以这个词大概是brainfuck的b吧）之后以此类推，一个一个查，最终拿到flag：hgame{bR4!NfUcK}

HelloRe

emmm，难得的签到题，拿到文件，放入IDA，看到

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  __int64 result; // rax@4
  __int64 v4; // rcx@4
  char s[8]; // [sp+0h] [bp-30h]@1
  __int64 v6; // [sp+8h] [bp-28h]@1
  __int64 v7; // [sp+10h] [bp-20h]@1
  __int64 v8; // [sp+18h] [bp-18h]@1
  __int64 v9; // [sp+28h] [bp-8h]@1

  v9 = *MK_FP(__FS__, 40LL);
  *(_QWORD *)s = 0LL;
  v6 = 0LL;
  v7 = 0LL;
  v8 = 0LL;
  puts("Please input your key:");
  fgets(s, 32, stdin);
  s[strlen(s) - 1] = 0;
  if ( !strcmp(s, "hgame{Welc0m3_t0_R3_World!}") )
    puts("success");
  else
    puts("failed..");
  result = 0LL;
  v4 = *MK_FP(__FS__, 40LL) ^ v9;
  return result;
}
```

好叭，拿到flag：hgame{Welc0m3_t0_R3_World!}

r & xor

拿到文件放入IDA，

```
__int64 v37; // [sp+138h] [bp-8h]@1

v37 = *MK_FP(__FS__, 40LL);
v31 = 3483951462304802664LL;
v32 = 6859934930880520053LL;
v33 = 3560223458491458926LL;
v34 = 2387225997007150963LL;
v35 = 8200481;
memset(v6, 0, 0x90uLL);
v7 = 1;
v8 = 7;
```

R很重要？那就按R！

```
v37 = *MK_FP(__FS__, 40LL);
v31 = '0Y{emagh';
v32 = '_3byam_u';
v33 = '1ht_deen';
v34 = '!!!en0_s';
v35 = '}!!';
```

flag就这么出来了？（提交后，呃，假的！咦~~）

看一下IDA的伪代码main函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax@2
  __int64 v4; // rsi@9
  signed int i; // [sp+8h] [bp-138h]@3
  int v6[6]; // [sp+10h] [bp-130h]@1
  int v7; // [sp+28h] [bp-118h]@1
  int v8; // [sp+30h] [bp-110h]@1
  int v9; // [sp+38h] [bp-108h]@1
  int v10; // [sp+3Ch] [bp-104h]@1
  int v11; // [sp+40h] [bp-100h]@1
  int v12; // [sp+44h] [bp-FCh]@1
  int v13; // [sp+48h] [bp-F8h]@1
  int v14; // [sp+4Ch] [bp-F4h]@1
  int v15; // [sp+50h] [bp-F0h]@1
  int v16; // [sp+54h] [bp-ECh]@1
  int v17; // [sp+5Ch] [bp-E4h]@1
  int v18; // [sp+60h] [bp-E0h]@1
  int v19; // [sp+64h] [bp-DCh]@1
  int v20; // [sp+68h] [bp-D8h]@1
  int v21; // [sp+6Ch] [bp-D4h]@1
  int v22; // [sp+70h] [bp-D0h]@1
  int v23; // [sp+74h] [bp-CCh]@1
  int v24; // [sp+78h] [bp-C8h]@1
  int v25; // [sp+80h] [bp-C0h]@1
  int v26; // [sp+84h] [bp-BCh]@1
  int v27; // [sp+88h] [bp-B8h]@1
```

```
  int v28; // [sp+8Ch] [bp-B4h]@1
  int v29; // [sp+90h] [bp-B0h]@1
  int v30; // [sp+94h] [bp-ACh]@1
  __int64 v31; // [sp+A0h] [bp-A0h]@1
  __int64 v32; // [sp+A8h] [bp-98h]@1
  __int64 v33; // [sp+B0h] [bp-90h]@1
  __int64 v34; // [sp+B8h] [bp-88h]@1
  int v35; // [sp+C0h] [bp-80h]@1
  char s[104]; // [sp+D0h] [bp-70h]@1
  __int64 v37; // [sp+138h] [bp-8h]@1

  v37 = *MK_FP(__FS__, 40LL);
  v31 = '0Y{emagh';
  v32 = '_3byam_u';
  v33 = '1ht_deen';
  v34 = '!!!en0_s';
  v35 = '}!!';
  memset(v6, 0, 0x90uLL);
  v7 = 1;
  v8 = 7;
  v9 = 92;
  v10 = 18;
  v11 = 38;
  v12 = 11;
  v13 = 93;
  v14 = 43;
  v15 = 11;
  v16 = 23;
  v17 = 23;
  v18 = 43;
  v19 = 69;
  v20 = 6;
  v21 = 86;
  v22 = 44;
  v23 = 54;
  v24 = 67;
  v25 = 66;
  v26 = 85;
  v27 = 126;
  v28 = 72;
  v29 = 85;
  v30 = 30;
  puts("Input the flag:");
  __isoc99_scanf("%s", s);
  if ( strlen(s) == 35 )
  {
    for ( i = 0; i < 35; ++i )
    {
      if ( s[i] != (v6[i] ^ *((_BYTE *)&v31 + i)) )
      {
        puts("Wrong flag , try again later!");
        result = 0;

        goto LABEL_9;
```

```
      }
    }
    puts("You are right! Congratulations!!");
    result = 0;
  }
  else
  {
    puts("Wrong flag , try again later!");
    result = 0;
  }
LABEL_9:
  v4 = *MK_FP(__FS__, 40LL) ^ v37;
  return result;
}
```

根据代码的意思，不难读出，输入的字符串（大概就是真flag了）长度为35，用它的每一个字符和v6数组的每一个整数做异或运算，结果为这个假flag，伪代码来看，v6数组只存了6个0，然后应该就"溢出"了，"溢出"了怎么办？

```
-0000000000000130 var_130         dd 6 dup(?)
-0000000000000118 var_118         dd ?
-0000000000000114                 db ? ; undefined
-0000000000000113                 db ? ; undefined
-0000000000000112                 db ? ; undefined
-0000000000000111                 db ? ; undefined
-0000000000000110 var_110         dd ?
-000000000000010C                 db ? ; undefined
-000000000000010B                 db ? ; undefined
-000000000000010A                 db ? ; undefined
-0000000000000109                 db ? ; undefined
-0000000000000108 var_108         dd ?
-0000000000000104 var_104         dd ?
-0000000000000100 var_100         dd ?
-00000000000000FC var_FC          dd ?
-00000000000000F8 var_F8          dd ?
-00000000000000F4 var_F4          dd ?
-00000000000000F0 var_F0          dd ?
-00000000000000EC var_EC          dd ?
-00000000000000E8                 db ? ; undefined
-00000000000000E7                 db ? ; undefined
-00000000000000E6                 db ? ; undefined
-00000000000000E5                 db ? ; undefined
-00000000000000E4 var_E4          dd ?
-00000000000000E0 var_E0          dd ?
-00000000000000DC var_DC          dd ?
-00000000000000D8 var_D8          dd ?
-00000000000000D4 var_D4          dd ?
-00000000000000D0 var_D0          dd ?
-00000000000000CC var_CC          dd ?
-00000000000000C8 var_C8          dd ?
-00000000000000C4                 db ? ; undefined
-00000000000000C3                 db ? ; undefined
-00000000000000C2                 db ? ; undefined
```

```
35  int v35; // [sp+C0h] [bp-80h]@1
36  char s[104]; // [sp+D0h] [bp-70h]
37  __int64 v37; // [sp+138h] [bp-8h]
38
39  v37 = *MK_FP(__FS__, 40LL);
40  v31 = '0Y{emagh';
41  v32 = '_3byam_u';
42  v33 = 'lht_deen';
43  v34 = '!!!en0_s';
44  v35 = '}!!';
45  memset(v6, 0, 0x90uLL);
46  v7 = 1;
47  v8 = 7;
48  v9 = 92;
49  v10 = 18;
50  v11 = 38;
51  v12 = 11;
52  v13 = 93;
53  v14 = 43;
54  v15 = 11;
55  v16 = 23;
56  v17 = 23;
57  v18 = 43;
58  v19 = 69;
59  v20 = 6;
60  v21 = 86;
61  v22 = 44;
62  v23 = 54;
63  v24 = 67;
64  v25 = 66;
65  v26 = 85;
66  v27 = 126;
67  v28 = 72;
000006C6 main:45
```

```
SP+0000000000000010
```

从各个数组的地址可以看出，v6之后是v7，然后接下来4个地址是undefined，四个字节，刚好一个int大小，试着数了一下，加上undefined的地址，从v6[0]到v30，刚好可以放35个int，猜想undefined的地址的值为0，然后手动查表，在线异或。

最终拿到flag：hgame{X0r_ 1s_interest1ng_isn't_it?}

Pro的Python教室（一）

进入网页

代码如下

```
import base64
import hashlib

enc1 = 'hgame{'
enc2 = 'SGVyZV8xc18zYXN5Xw=='
enc3 = 'Pyth0n}'

print 'Welcome to Processor\'s Python Classroom!\n'
print 'Here is Problem One.'
print 'There\'re three parts of the flag.'

print '-----------------------------------------------'

print 'Plz input the first part:'
first = raw_input()
if first == enc1:
    pass
else:
    print 'Sorry , You\'re so vegatable!'
    exit()

print 'Plz input the secend part:'
secend = raw_input()
secend = base64.b64encode(secend)
if secend == enc2:
    pass
else:
    print 'Sorry , You\'re so vegatable!'
    exit()

print 'Plz input the third part:'
third = raw_input()
third = base64.b32decode(third)
if third == enc3:
    pass
else:
    print 'Sorry , You\'re so vegatable!'
    exit()

print 'Oh, You got it !'
```

根据代码，那么flag的第一段就是**hgame{**了

第二段被base64编码之后是SGVyZV8xc18zYXN5Xw==，那么把这段解码，为：**Here_1s_3asy_**

第三段，根据代码，是被解码后，为**Pyth0n}**，但我直接

```
flag3=base64.b32encode ('Pyth0n}')
```

发现程序运行不了，又因为其实这一段挺好看的，我就直接把它当第三段了，提交，成功

所以flag：hgame{Here_1s_3asy_Pyth0n}（但其实不明白为什么，赶紧学python去了，不然下周要凉）

PWN

aaaaaaaaaa

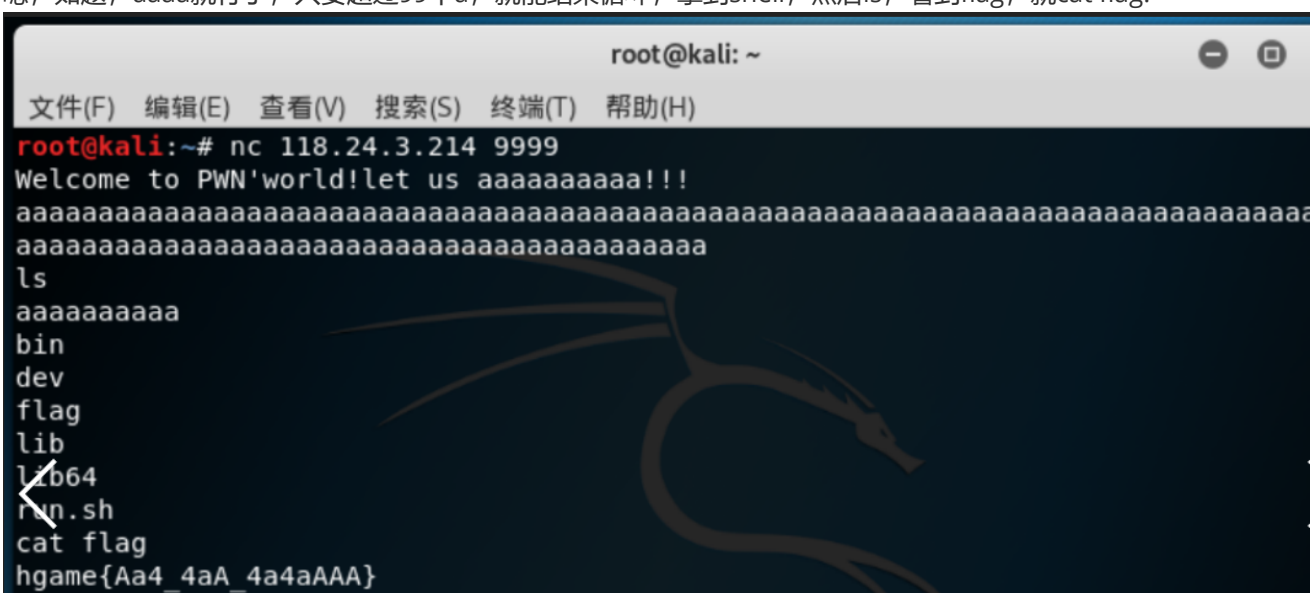
呵呵，签到题，（对于我这个没用过Linux系统的人来说，我还是做了半天）

先打开文件，放到IDA，看到伪代码

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int v3; // eax@4
  signed int v5; // [sp+Ch] [bp-4h]@1

  setbuf(_bss_start, 0LL);
  signal(14, handle);
  alarm(0xAu);
  puts("Welcome to PWN'world!let us aaaaaaaaaa!!!");
  v5 = 0;
  while ( 1 )
  {
    v3 = v5++;
    if ( v3 > 99 )
      break;
    if ( getchar() != 97 )
      exit(0);
  }
  system("/bin/sh");
  return 0;
}
```

嗯，如题，aaaa就行了，只要超过99个a，就能结束循环，拿到shell，然后ls，看到flag，就cat flag!



```
root@kali:~# nc 118.24.3.214 9999
Welcome to PWN'world!let us aaaaaaaaaa!!!
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
ls
aaaaaaaaaa
bin
dev
flag
lib
lib64
run.sh
cat flag
hgame{Aa4_4aA_4a4aAAA}
```

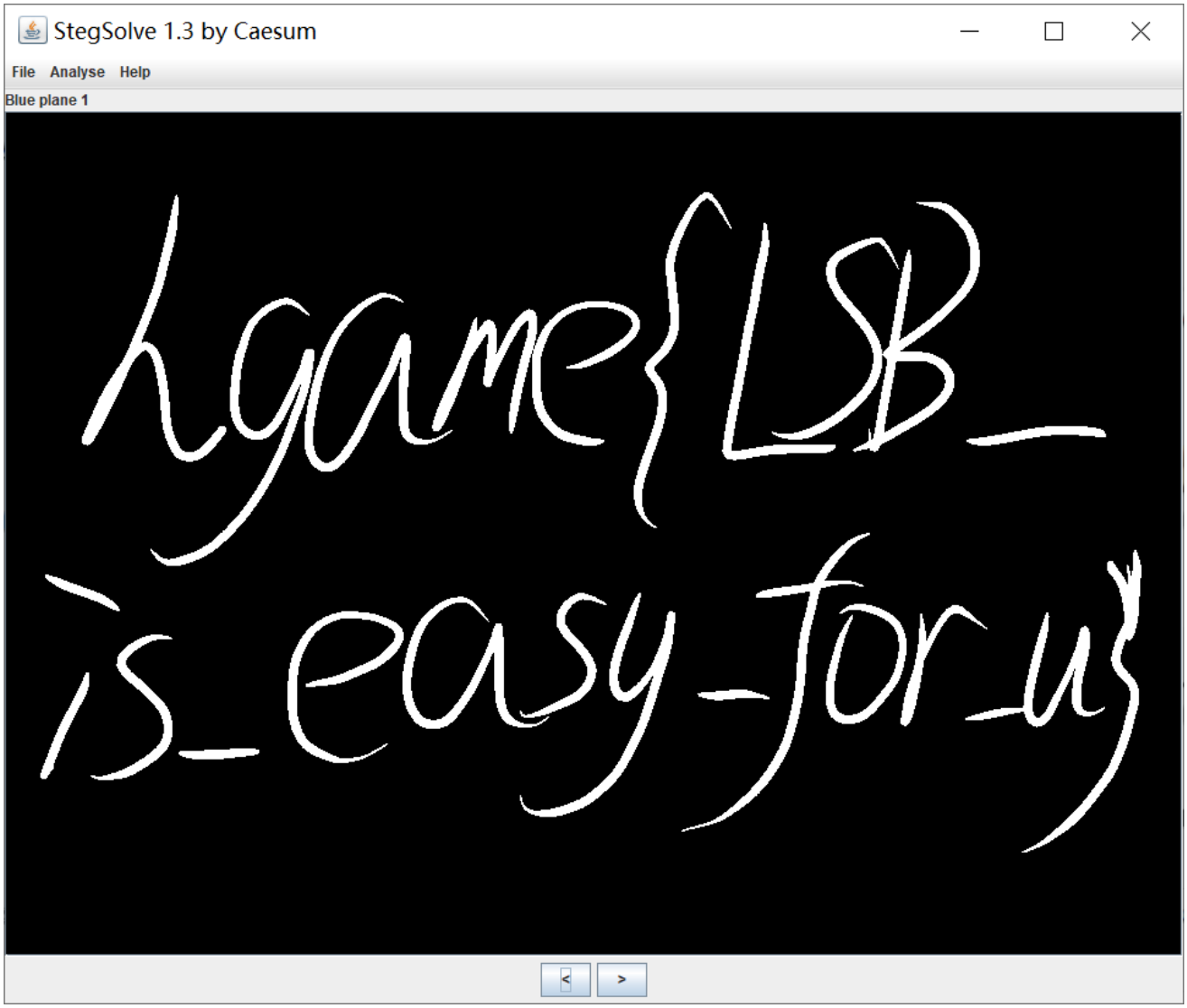
拿到flag：hgame{Aa4_4aA_4a4aAAA} (大概是本次hgame我拿得到得唯一一个pwn的flag了吧)

MISC

Hidden Image in LSB

（原来还要让我们写脚本的么，50分要做出500分的感觉?）

既然学长都说了，那就用stegsolve 打开图片，下面两个按钮，一顿乱按

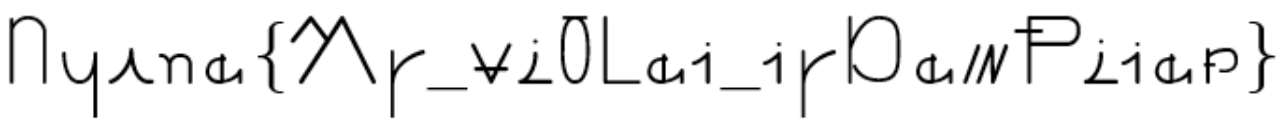


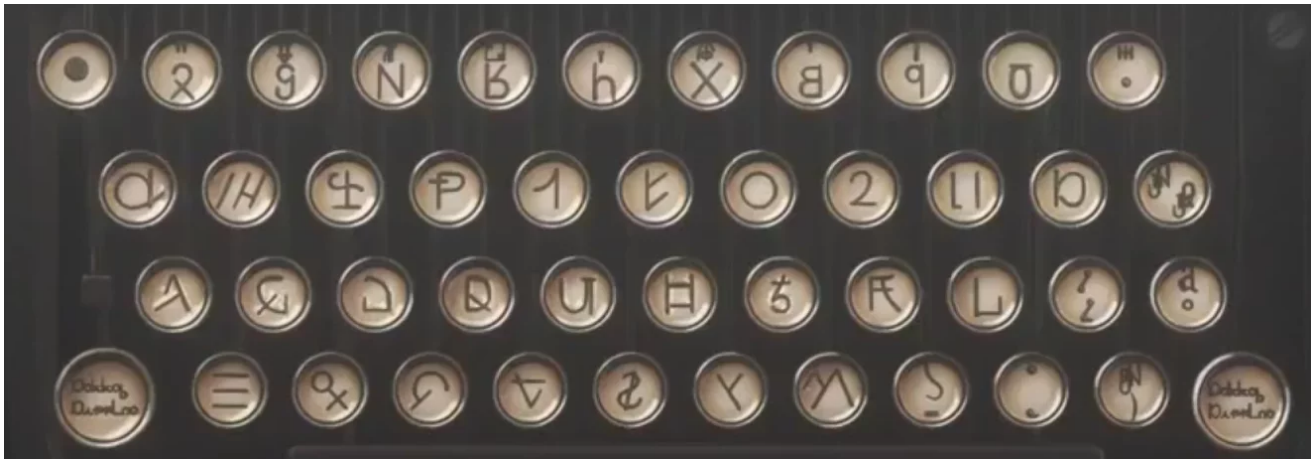
拿到flag：hgame{LSB_is_easy_for_u}（不不不，easy for stegsolve）

打字机

打开文件，两张图，以为又是隐写，后来呃，，，，原来就是单纯的对照啊

加上hint，紫罗兰（violent），就容易确定答案了。





键盘上的都是大写，所以和键盘上稍微不一样的就是小写，键位还是按照正常键盘排序。在加上flag的格式：hgame，确定了e的模样。

最后得到flag：hgame{My_vi0Let_tyPewRiter}

Broken Chest

（箱子坏掉了，那就修啊），打开压缩包，文件拖不出来？

把这个zip放进winhex

Broken-Chest (2).zip

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	4F	4B	03	04	14	00	09	00	08	00	55	BB	35	4E	CE	7C	OK U»5NÎ|
00000016	B3	B0	22	00	00	00	14	00	00	00	08	00	00	00	66	6C	³°" fl
00000032	61	67	2E	74	78	74	67	49	3F	48	A0	BE	53	8B	38	E4	ag.txtgI?H ¾S‹8ä
00000048	5A	42	49	02	08	5D	55	A6	4A	67	B2	B3	CE	B0	6E	C1	ZBI]U¦Jg²³Î°nÁ
00000064	0B	85	DC	EB	4F	91	4D	BF	50	4B	07	08	CE	7C	B3	B0	…ÜëO'M¿PK Î|³°
00000080	22	00	00	00	14	00	00	00	50	4B	01	02	1F	00	14	00	" PK
00000096	09	00	08	00	55	BB	35	4E	CE	7C	B3	B0	22	00	00	00	U»5NÎ|³°"
00000112	14	00	00	00	08	00	24	00	00	00	00	00	00	00	20	00	$
00000128	00	00	00	00	00	00	66	6C	61	67	2E	74	78	74	0A	00	flag.txt
00000144	20	00	00	00	00	00	01	00	18	00	3E	2C	76	B6	9D	B1	>,v¶ ±
00000160	D4	01	3E	2C	76	B6	9D	B1	D4	01	1D	F1	7E	C5	9C	B1	Ô >,v¶ ±Ô ñ~Åœ±
00000176	D4	01	50	4B	05	06	00	00	00	00	01	00	01	00	5A	00	Ô PK Z
00000192	00	00	58	00	00	00	10	00	53	30	6D	45	54	68	31	6E	X S0mETh1n
00000208	67	5F	55	35	65	66	75	4C									g_U5efuL

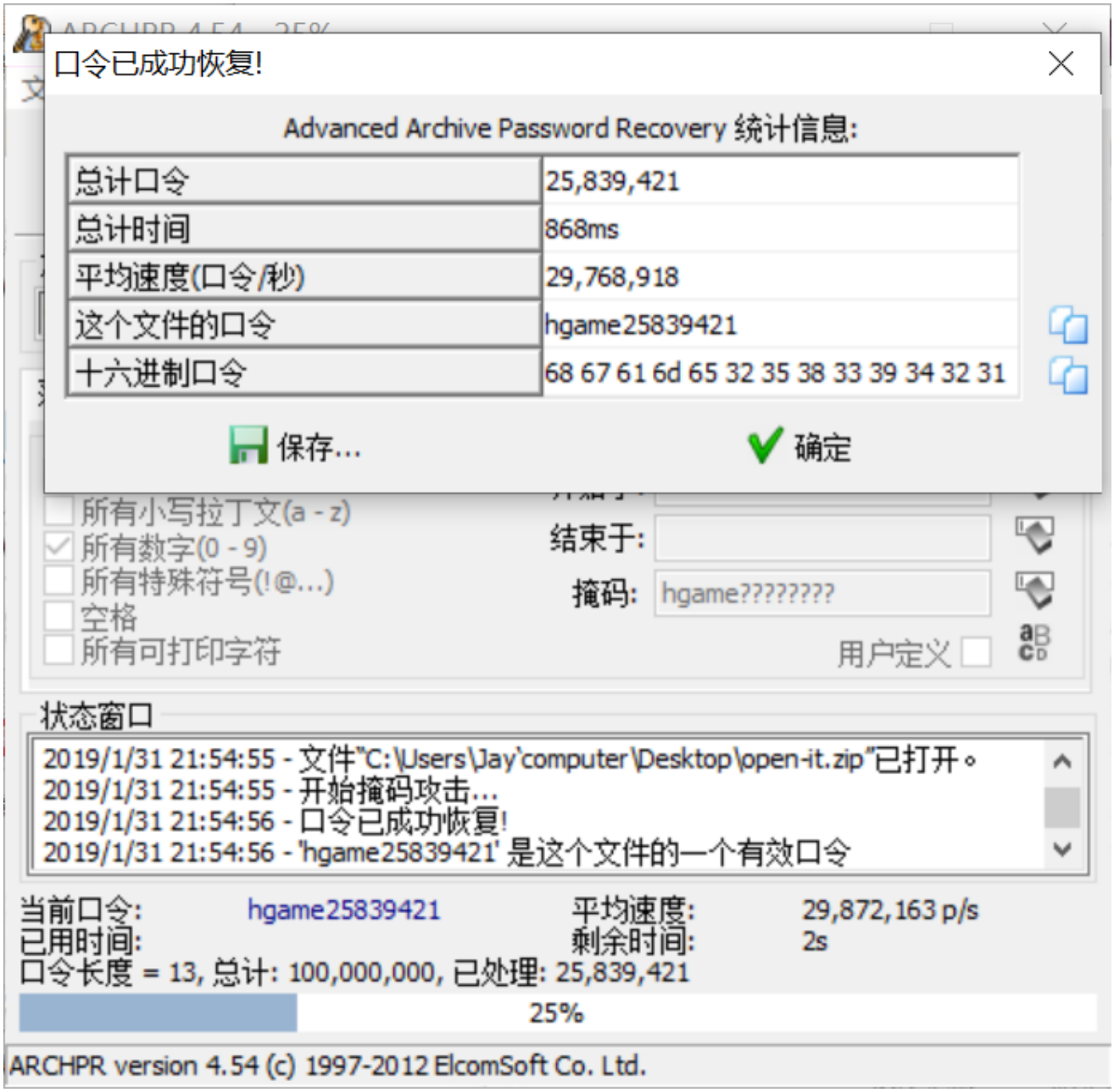
压缩包开头不是PK么，怎么OK了？看来是这里坏掉了，改4F为50后，打开压缩包的flag，有密码？这不就在旁边嘛，**S0mETh1ng_U5efuL**，这么明显的给你了。

最后拿到flaghgame{Cra2y_D1aM0nd}

Try

（100分，算这里的压轴嘛？）

pcapng文件，用Wireshark打开，->导出对象-> HTTP, 发现一个zip，导出，zip里是一个password.txt，内容是**hgame********，还有个zip，里面是加密的1.jpg，于是用ARCHPR掩码攻击，先从最简单试起，猜它8个都是数字，竟然成功拿到密码



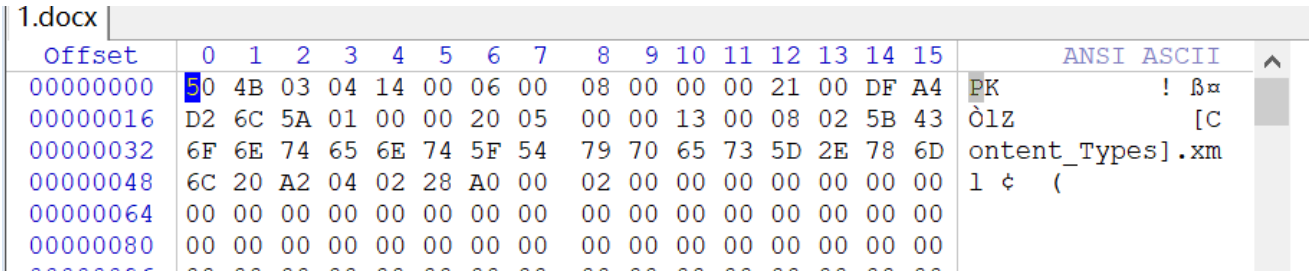
打开1.jpg，哇，这是哪个漂亮小姐姐，可是不给flag，那就放进winhex，

1.jpg.zip

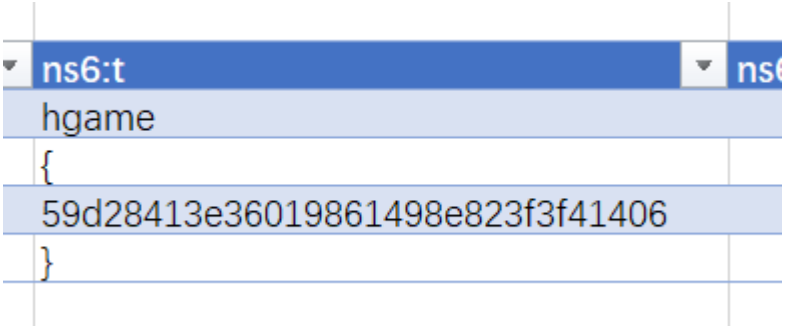
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00089264	E7	F6	FA	23	46	F0	00	80	3F	3A	C1	7E	B1	7F	E7	8C	çöú#Fð €?:Á~± çŒ
00089280	F8	23	9C	08	FF	CA	FE	E5	9A	F8	45	FF	FD	B9	F0	17	ø#œ ÿÊþåšøEÿý¹ð
00089296	A8	49	FE	E5	94	F8	37	32	04	E4	CF	2C	B8	D7	5F	C7	¨Iþå”ø72 äÏ,¸×_Ç
00089312	6B	6F	9B	E8	7E	A6	FE	02	50	4B	01	02	3F	00	14	00	ko›è~¦þ PK ?
00089328	09	00	08	00	F8	74	32	4E	DD	C4	CD	94	E7	24	00	00	øt2NÝÄÍ”ç$
00089344	92	2F	00	00	06	00	24	00	00	00	00	00	00	00	20	00	’/ $
00089360	00	00	00	00	00	00	31	2E	64	6F	63	78	0A	00	20	00	1.docx
00089376	00	00	00	00	01	00	18	00	C6	82	9F	9A	F8	AE	D4	01	Æ‚Ÿšø®Ô
00089392	B7	3B	22	83	9C	B3	D4	01	4A	37	DD	06	F8	AE	D4	01	·;"ƒœ³Ô J7Ý ø®Ô
00089408	50	4B	05	06	00	00	00	00	01	00	01	00	58	00	00	00	PK X
00089424	0B	25	00	00	00	00											%

拉到最下面发现1.docx和PK，（这个小姐姐果然不单纯）先暴力一点，直接改后缀。（希望小姐姐只藏了一个zip包）嗯，没问题。又是加密文件？拖进ARCHPR，发现不让破，说是文件未被口令保护，那应该是伪加密了。再次拖进winhex，如上图，改文件的全局方式位标记 09 00为00 00，打开1.docx文件，嗯？啥都没有？无字天书！（终于对上题目了）

再次拖进winhex（拖上瘾了），



开头就是PK，再次暴力改后缀（希望不会出事），呃，没事是没事，可是.xml是哪路神仙啊，无奈百度，在Excel表格中试了好几个.xml文件，直到试到document.xml



拿到flag：hgame{59d28413e36019861498e823f3f41406 }

(这个flag，不走寻常路啊，我以为里面还要解码啥的，用了各种工具，最后无奈上交，竟然，ok fine)

CRYPTO

Mix

（那个URL什么鬼）

--.../....-/....-/-.../--.../...--/...../..-./-..../-../-..../..-./--.../----./....-/....-/--.../.----/-..../-.../--.../-.../-..../..---/...../.----/-..../-..../...--/....-/...--/-----/-..../...../--.../-..

嗯哼，在线解摩斯密码，得：744B735F6D6F7944716B7B6251663430657D

一眼过去，十六进制ASCII码，在线转，得：tKs_moyDqk{bQf40e}

两个花括号，有点意思了，栅栏加密，手动解，得：tsmyq{Q4eK_oDkbf0}

有flag的格式了，凯撒加密！还是在线解（我好水哦。。）找到hgame开头得那一个，就决定是你了，flag！hgame{E4sY_cRypt0}

Base全家

打开文件，下了一跳，密集恐惧症orz。

大概是被base全家加密了不知道多少次吧。然后，对于一个python小白来讲，当然是网上找脚本了，（赶紧学python，赶紧学python）

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-


from base64 import *

result = {
     '16':lambda x :b16decode(x),
     '32':lambda x :b32decode(x),
     '64':lambda x :b64decode(x),
}

flag = open('code.txt','r')
flag_content = flag.read()
# print flag_content

num = ('16','32','64')

for i in range(20):
    for k in num:
        try:
            flag_content = result[k](flag_content)
            if flag_content:
                break
            else:
                continue
        except:
            pass

with open("final_flag.txt","wb") as final_flag:
    final_flag.write(flag_content)
```

尝试了几个循环次数，发现循环二十次后能得到：

base58 : 2BAja2VqXoHi9Lo5kfQZBPjq1EmZHGEudM5JyDPREPmS3CxrpB8BnC

base58是啥？ctf在线工具！

解码**2BAja2VqXoHi9Lo5kfQZBPjq1EmZHGEudM5JyDPREPmS3CxrpB8BnC**

得到flag：hgame{40ca78cde14458da697066eb4cc7daf6}

（因为有Try的flag的经历，这个flag，我没有对它产生丝毫的怀疑，真的!）